

Internet Architecture and Innovation

Barbara van Schewick



Internet Architecture and Innovation

Internet Architecture and Innovation

Barbara van Schewick

**The MIT Press
Cambridge, Massachusetts
London, England**

© 2010 Barbara van Schewick

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

For information about special quantity discounts, email special_sales@mitpress.mit.edu.

Set in Stone Sans and Stone Serif by Toppan Best-set Premedia Limited. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Van Schewick, Barbara.

Internet architecture and innovation / Barbara van Schewick.

p. cm.

Includes bibliographical references and index.

ISBN 978-0-262-01397-0 (hardcover : alk. paper) 1. Internet. 2. Computer network architectures. 3. Technological innovations. 4. Business—Data processing. I. Title.

TK5105.875.157V378 2010

004.6'5—dc22

2009037130

10 9 8 7 6 5 4 3 2 1

for Steffen, Lukas, and Daniel

Contents

Acknowledgments ix

Introduction 1

I Foundations

1 Architecture and Innovation 19

Architecture 19

The Relationship between Architecture and the Economic System 23

The Relationship between Architecture and Innovation 28

II The End-to-End Arguments and the Original Architecture of the Internet

2 Internet Design Principles 37

Modularity 38

Integrated Design 44

Layering 46

Modularity and Layering in Network Architectures 50

The End-to-End Arguments 57

3 The Original Architecture of the Internet 83

Introduction to the Original Architecture of the Internet 83

The Internet and the Layering Principle 88

The Internet and the End-to-End Arguments 90

Some Misconceptions about the End-to-End Arguments and the Architecture of the Internet 103

III Architectural Constraints on Innovation

4 Architecture and the Cost of Innovation 115

Costs of Change in Modular and Integrated Architectures 118

Costs of Change in the Original Architecture of the Internet 137

5	Architecture and the Organization of Innovation	165
	Architecture and Organization in Modular and Integrated Architectures	166
	Implications of the Link between Architecture and Feasible Governance Structures	195
	Architecture and Organization in the Original Architecture of the Internet	201
6	Architecture and Competition among Makers of Complementary Components	215
	Architectural Differences—Application Awareness and Application Control	217
	Effect of Ability to Discriminate or Exclude	218
	Effect on Pricing Strategies	273
	Conclusion	280
IV The End-to-End Arguments and Application Innovation		
7	Network Architectures and the Economic Environment for Application Innovation	285
	Potential Deviations from the End-to-End Arguments	286
	The Four Network Architectures	287
	Analysis	289
8	Decentralized versus Centralized Environments for Application Innovation	297
	Effects of Differences in Innovator Diversity and in Control over Innovation	298
	Effect of Differences in Control over Deployment	349
	Overall Effect of Architectural Differences on Application Innovation	351
9	Public and Private Interests in Network Architectures	355
	Public Interests in Network Architectures	355
	Network Providers' Private Interests in Network Architectures	371
	Conclusion	377
	Notes	393
	References	495
	Index	551

Acknowledgments

Over the years, this work has been supported by many people. I am grateful to all of them. I'm immensely grateful to Dean Larry Kramer for his generous support of this project, and for helpful conversations and advice. Many colleagues kindly shared insights or advice. I am particularly grateful to Dick Craswell, Rob Daines, Paul Goldstein, Hank Greely, Joe Grundfest, Mark Kelman, Mike Klausner, Mark Lemley, Helen Stacey, Al Sykes, and Allen Weiner.

Mike Ananny, Bob Briscoe, Josh Cohen, Larry Lessig, Mark Lemley, and Phil Weiser carefully read the entire manuscript and spent hours discussing it with me. Their thoughts and comments were invaluable. Dick Craswell, Matthew Elliott, Brett Frischmann, Shane Greenstein, Joe Grundfest, Al Sykes, and Jonathan Zittrain read one or more chapters. Their thoughtful comments improved the manuscript greatly.

Over the years, I have had many conversations and discussions which helped shape the arguments in this book. I am particularly grateful to Marvin Ammori, Pio Baake, Jack Balkin, Stefan Bechtold, Scott Bradner, Susan Crawford, Gerry Faulhaber, Ed Felten, Terry Fisher, Mark Gaynor, Sharon Gillett, David Isenberg, Scott Hemphill, Louis Kaplow, Bill Lehr, Robert Pepper, Howard Shelanski, Karen Sollins, Larry Solum, Jim Speta, Kevin Werbach, Tim Wu, and Chris Yoo. I have also had many interesting conversations on Internet architecture and policy with my colleagues in the Trilogy project, in particular with Marcelo Bagnulo Braun, Olivier Bonaventure, Bob Briscoe, Louise Burness, Costas Courcoubetis, Phil Eardley, Alan Ford, Lars Eggert, Robert Hancock, Mark Handley, Toby Moncaster, Arnaud Jacquet, Iljitsch van Beijnum, and Damon Wischik.

My work on the book was supported by a number of students. Mike Ananny, Matthew Elliott, Ronan Flanagan, Shira Liu, Pedro Miranda, Samad Nasserian, Elisabeth Oppenheimer, and Hewan Teshome provided excellent research assistance. Mike Annany, Ingrid Erickson, Jordan Segall,

and Kathyne Tafolla Young helped edit earlier versions of some of the chapters. Alexander Harris, Stuart Loh, Jonathan Mayer, Michelle Munoz, David Rizk, Emily Roberts, Robert Sensenbrenner, Roman Swoopes, and Andrew Zahn checked the citations. I am very grateful to all of them.

Special thanks to the amazing librarians at Stanford Law School, who found even the most exotic books and got them on my desk in incredibly short time, and to Sonia Moss for excellent interlibrary loan assistance.

Mary Ann Rundell, my assistant, prepared various versions of the manuscript, maintained the bibliographic database, organized and managed the checking of citations, and worked through various versions of the bibliography to find and correct errors. Her dedication to this project was incredible. I am fortunate to work with her. Many thanks to Stephanie Basso for her help in checking the bibliography.

Dolfen Leung ably converted my drawings into professional figures.

I am grateful to everybody at the MIT Press who helped turn the manuscript into a book. Special thanks to Elizabeth Murry, who acquired the manuscript, to John Covell, my editor, for encouragement and advice, and to Paul Bethge for editing the manuscript.

David Reed generously granted permission to reprint his account of the split between TCP and IP in the early days of the Internet.

This book builds on some of my earlier work. An earlier version of the first two sections of chapter 6 was published as “Towards an Economic Framework for Network Neutrality Regulation,” *Journal on Telecommunications and High Technology Law* 5 (2007), no. 2: 329–391. I wrote that paper while I was a Senior Researcher at the Telecommunication Networks Group at the Technische Universität Berlin. I’m grateful to Adam Wolisz for supporting my work on this project and for providing an intellectually stimulating environment in which my thinking on these issues could evolve. During this time, I had many interesting discussions on Internet architecture, innovation, and regulation with Mathias Bohge, Marc Emmelmann, Michael Eyrich, James Gross, Vlado Handziski, Sven Hermann, Andreas Koepke, Martin Kubisch, Günter Schäfer, Lars Westerhoff, Sven Wiethölter, Daniel Willkomm, and Andreas Willig. Matthias Bärwolff, Dick Craswell, Einer Elhauge, Joseph Farrell, and Chris Yoo provided important feedback on the final version of this article that helped me improve the arguments in the book. I would also like to thank the participants of the faculty workshops at the law schools at Emory University, Fordham University, Harvard University, Michigan University, New York University, Stanford University, the University of California at Los Angeles, the University of Illinois, the University of Pennsylvania, the University of Southern

California, the University of Virginia, and the Wharton School's Legal Studies Department, and the participants in the 33rd Research Conference on Communication, Information, and Internet Policy (TPRC 2005), the Berkman Center for Internet and Society Luncheon Series at Harvard Law School, and the Computer Science and Artificial Intelligence Laboratory Speaker Series at the Massachusetts Institute of Technology for helpful questions and comments on the arguments of this article and the topic of this book.

Many of the ideas in this book were first explored in my doctoral dissertation at the Technische Universität Berlin. I'm particularly grateful to my advisors, Bernd Lutterbeck and Larry Lessig. I'm indebted to Bernd Lutterbeck for discussions, comments, and support throughout the years. I would also like to thank the other members of my dissertation committee, Ulrich Heiss and Adam Wolisz, for approaching this project with enthusiasm and an open mind. I did a substantial part of my doctoral research at Stanford Law School's Center for Internet and Society, during my time as a residential fellow there and during a later visit. I'm grateful to Stanford Law School and the Center for Internet and Society for their support during this time. My doctoral work was also financially supported by the German National Academic Foundation and the Gottlieb Daimler- and Karl Benz-Foundation. A number of people offered thoughtful comments on the finished dissertation that helped shape the arguments in the book. They include Mike Ananny, Matthias Bärwolff, Stefan Bechtold, Marjory Blumenthal, Ingrid Erickson, Gerry Faulhaber, Josh Lerner, and Doug Lichtman.

Finally, a few more personal thanks:

I owe a lot to Larry Lessig. I became interested in Internet architecture and innovation after reading his work on the role of code in cyberspace and on the impact of the end-to-end arguments on innovation. He invited me to Stanford when I started to work in this field, and volunteered to be my advisor. Our discussions and his comments have been invaluable to me. He continues to be a great colleague, counselor, and friend.

Over the years, I have had many inspiring conversations with Yochai Benkler, David Clark, Brett Frischmann, Arnold Picot, David Reed, and Phil Weiser that went far beyond the topic of this book. I have learned a lot from them. I am extremely grateful for their support and advice.

Jörg Riecken's advice, enthusiasm, and support were instrumental in the formative stages of this project. I am very grateful for everything he has done. I am indebted to my parents for their love, encouragement, and support over the years. They were always there when I needed them. My

children, Lukas and Daniel, have grown up with this project. Many thanks for their patience when I couldn't be around, and for being so much fun to be with when I could. Last, but not least, I would like to thank my husband, Steffen van Schewick, for his love, encouragement, and support throughout this project. A lot of arguments in this book have been developed, clarified, and refined in discussions with him. An environmental planner by training, he learned more about the impact of Internet architecture on innovation than he ever wanted to know. He invested an incredible amount of time and energy to support this work in so many ways that it is impossible to mention them all. Without him, this book would not have been possible.

Introduction

In the thirty years since its inception, the Internet has experienced remarkable growth. What started in the 1970s as an experimental network connecting research and military networks has become a global network linking more than 1.6 billion users worldwide.¹ The Internet's growth has been fueled by an unprecedented amount of innovation. Over the years, network engineers have developed numerous new physical networking technologies (including Wi-Fi and optical networking technologies) over which the Internet can run. The Internet now connects everything from sensors to supercomputers. A constant stream of new applications lets users do new things, or do them more efficiently, making the Internet more attractive and useful to diverse users. As more and more users adopt them, these applications, the content they help produce and make available, and the new economic, social, cultural, and political practices they enable are transforming all areas of society. They change, for example, how firms can organize themselves, make their products, and interact with customers and with other firms. They increase the opportunities available to us, helping us to be more productive in our professional and private lives; to interact with relatives, friends, and strangers; to get to know them, communicate, or work with them; to educate ourselves using a variety of sources; and to participate in social, cultural, and democratic discourse.

The Internet's growing size, its transition from a research network operated by public entities to a commercial network operated by commercial providers who need to make profits, and its transition from a network connecting a small community of users who trust one another to a global network with users who do not know one another and may even intend to harm one another put pressure on the Internet's technical foundations. To deal with the impending shortage of Internet addresses, users and network providers deploy network-address translators that let several devices share the same address. To protect their networks against attacks,

organizations put firewalls that block potentially harmful applications at the borders of their private networks. To increase their profits, network providers use technologies that enable them to identify and control the applications and the content passing through their networks.

These changes violate the design principles on which the Internet originally was based. They are hotly contested in technical forums such as the Internet Engineering Task Force, which develops and maintains the core Internet standards. The debate has also reached the political arena: arguing that these changes will destroy the Internet as a platform for innovation, free speech, and economic growth, scholars, public-interest organizations, industry organizations, and companies have asked regulatory agencies and legislative bodies all over the world to step in and restore the beneficial environment for innovation that the architecture of the Internet originally created.² In the United States, the debates over open access to broadband cable networks, over network neutrality, and over network management in broadband networks have all been shaped by such arguments.³

Independent of attempts to solve problems within the framework provided by the existing Internet, a new generation of networking research projects has begun to contemplate how the Internet of the future should look. Many of these projects start from a clean architectural slate, essentially creating a new Internet architecture from scratch.

In each of these cases, network engineers, legislators, and regulators face the same questions: How important are the design principles that shaped the Internet's original architecture? Are the (proposed) deviations much-needed improvements that reflect changed technical or business requirements, or are we losing the very features that were central to the Internet's success?

For regulators, the situation is even more complicated. Shouldn't a technical issue such as how to best structure the architecture of the Internet be left to network engineers? And if certain architectures foster innovation more than others, why won't the market create them?

This book aims to answer these questions. It examines how changes in the Internet's architecture (that is, its underlying technical structure) affect the economic environment for innovation and evaluates the impact of these changes from the perspective of public policy.

The answers are not obvious. After decades of research on innovation, we understand how changes in laws, in norms, or in prices affect the economic environment for innovation and how they affect innovators' decisions to innovate. We lack a similar understanding of how architecture affects innovation. Just as the architecture of a house describes its basic

inner structure, the architecture of a complex system describes the basic inner structure of the system—its components, what they do, and how they interact to provide the system’s functionality. That such a technical structure may have economic consequences at all is a relatively recent insight.⁴ Most people still think of architectures as technical artifacts that are relevant only to engineers.

Thus, understanding how the Internet’s architecture affects innovation requires us to think more generally about how architectures affect innovation. How can the architecture of a complex system influence the economic system in general, or an economic activity such as innovation in particular? Which features of an architecture affect the economic environment for innovation, and how do these features vary across different architectures or design principles?

The answers to these questions are relevant beyond the Internet. Since in most cases a system’s functional requirements determine the system’s architecture only in part, system architects have considerable latitude to choose different architectures, and this latitude allows them to consider other goals too. This freedom is particularly pronounced in software-intensive systems that are less constrained by the laws of nature than the design of physical products is. In particular, if architectures have economic consequences, system architects can create architectures that prioritize and realize particular economic goals.

These insights open new opportunities for businesses, for law, and for public policy. Businesses may want to engage in “strategic design” by creating architectures that shape the competitive environment in their favor. In the future, being able to design architectures that further a firm’s strategic interests or knowing how to evaluate other firms’ architectural strategies and react to them may be as important to a firm’s success in the marketplace as a firm’s ability to engage in more conventional forms of competition.⁵

For law and public policy, the economic impact of architecture seems to be empowering and challenging at the same time. Traditionally, policy makers have used the law to bring about desired economic effects. Architecture may provide an alternative way of influencing economic systems. Apart from using architecture to realize their own economic goals, policy makers may have to constrain the extent to which private actors can use architecture to further their private economic interests. This is particularly relevant to communications policy, a field in which certain architectures may seriously restrict regulators’ ability to regulate at a later stage. Moreover, as communications networks continue to permeate more and more

sectors of the economy, the negative effects of an architecture that strongly favors a few economic actors may be particularly long-term and severe.

To exploit the effect of architecture on the economic system in practice, to design architectures that further our interests, or to understand what other people's architectures may mean for us we need to understand exactly how an architecture influences the economic system and what features of an architecture we must tweak to create a specific economic effect. This book is a step toward that goal.

Overview of Chapters

Chapter 1 introduces a theoretical framework for understanding how architectures relate to economic systems and, more specifically, for understanding how architectures affect innovation. It explains architectures and design principles as concepts, and it highlights ways in which architectures influence economic systems and are influenced by them. In particular, the relationship between architectures and the economic system can be understood within a broader framework that economists use to explain the evolution of the economy as a whole. In this framework, the economic system evolves as economic actors pursue their own interests within a set of constraints, and as they act to change those constraints. Constraints delimit the options available to economic actors and influence the costs and benefits associated with these options. Well-known constraints include prices, laws, norms, and the natural and technical environments in which economic actors exist. Like these other constraints, an architecture can affect human behavior by imposing constraints on those who interact with the architecture or are exposed to it. Specifically, by imposing constraints on those who design, produce, and use a complex system, the architecture of the system (and the design principles that were used to create it) can influence the economic system in which the system is developed, produced, and used. Different architectures may impose different constraints, which may result in different decisions by economic actors, which in turn may result in different firm and market structures and different levels of economic activity. And by changing existing architectures or creating new ones, economic actors can change the constraints that architectures impose.

Before we can understand how specific architectures constrain economically, we need to understand how they constrain technically. Part II of the book introduces the architecture we will use to study the effect of architecture on innovation—the original architecture that governed the

Internet from its inception to the early 1990s⁶—and the design principles that were used to create that architecture: modularity, layering, and the end-to-end arguments.

While network engineers agree that the end-to-end arguments are among the few architectural principles underlying the architecture of the Internet,⁷ other scholars have offered widely differing and often contradicting views on what the end-to-end arguments are, what they say, and how they relate to the architecture of the Internet. For example, some suggest that the end-to-end principle is “an important architectural principle that has governed the Internet since its inception”⁸; others say that “the end-to-end principle simply does not dictate a robustly specified functional design for the network”⁹ but “follows from (and is an articulation of) the implicit design principle inherent to the layers model of the TCP/IP protocol [*sic*]”¹⁰; still others argue that “the Internet was never wholly end-to-end”¹¹ and that the end-to-end argument “is not an organizing principle; . . . if it is a principle, it is probably not true, and . . . even if it is true, it is probably not useful.”¹² In policy debates concerning the architecture of the Internet, opponents of regulation often argue that proponents of certain regulatory interventions (for example, of open-access rules or network-neutrality rules) have stretched the end-to-end principle beyond its original meaning.¹³ Even networking engineers often disagree about whether a certain technical solution violates the end-to-end arguments or not. In view of the high level of confusion, I discuss the end-to-end arguments and their relationship to the architecture of the Internet in detail.

My analysis yields an important insight: there is no single version of the end-to-end arguments, but two different ones that embody different rules for architectural design. The first version, which I call *the narrow version*, was presented by Jerome Saltzer, David Reed, and David Clark in the 1981 paper in which the end-to-end arguments were first named and identified as a design principle¹⁴; the second version, which I call *the broad version*, is the focus of later papers by these authors.¹⁵ The difference between the two versions is not immediately apparent, and Saltzer, Reed, and Clark never explicitly drew attention to the change in definition. There are, however, real differences in scope, content, and validity that make it necessary to distinguish between the two versions. At the same time, the silent coexistence of two different design principles under the same name explains some of the confusion that surrounds the end-to-end arguments.

Chapter 2 describes the design principles that were used to create the original architecture of the Internet and highlights the trade-offs involved in each of them. Chapter 3 describes how these design principles shaped

the original architecture of the Internet. Chapter 3 also discusses some of the misconceptions surrounding the end-to-end arguments and their relationship to the architecture of the Internet. For example, scholars have argued that the end-to-end arguments rule out architectures that store state in the network, prohibit the provision of Quality of Service in the network,¹⁶ require the network to be simple, or constrain the design of applications by preferring certain application-management structures over others. Others have argued that the original architecture of the Internet was not based on the end-to-end arguments on the ground that the end-to-end arguments are no different from the layering principle, or on the ground that the original paper describing the end-to-end arguments was published after the original architecture of the Internet was developed, or on the ground that e-mail had a certain management structure.

To evaluate how the Internet's architecture affects innovation, we must understand exactly how architecture constrains economic actors. How do design principles and the architectures they shape influence the costs and benefits associated with a given innovation? How do they affect who can design and build an innovation? Answering these questions is the goal of part III. Each chapter in that part focuses on a particular aspect of the relationship between architecture and innovation and explores it in detail. The resulting insights are then applied to assess how the Internet's original architecture affects innovation. Although both versions of the end-to-end arguments shaped the original architecture of the Internet, only the broad version affects the economic environment for innovation. Therefore, parts III and IV focus on modularity, layering, and the broad version.

Chapter 4 explores how architecture influences the cost of realizing an innovation. It first discusses the effect of architecture on the costs of innovation in the context of modular and integrated architectures. Using the theory of real options, it explores the relationship between the costs of innovation implied by different architectures and the value of experimentation under uncertainty. It shows how the different costs of change in modular and integrated architecture affect the option value of these architectures, and how these differences affect the willingness of component designers to take risks when designing their components.

Chapter 4 also explores how differences between alternative modular architectures influence the option value of comparable modules in these architectures. In particular, the analysis shows how differences in the costs of innovation with respect to a module, the nature of dependencies on that module, and the uncertainty surrounding a module may affect the level of investment and rate of change of that module in alternative archi-

tures. The chapter then uses these general insights about the effect of architecture on the costs of innovation and on the likelihood of innovation to determine how specific design choices in the Internet's original architecture affect these factors. It explores how the decision for or against the broad version of the end-to-end arguments influences the costs of developing applications, and how the decision for or against the use of relaxed layering with a portability layer affects the costs of developing the physical network technologies over which the Internet can run. It also highlights how these architectural choices influence the costs of producing, distributing or deploying innovations.

The final subsection of chapter 4 explores how the Internet's original architecture affects the rate of change at the Internet and transport layers. In recent years, the difficulties associated with evolving the core of the Internet have received much attention in the networking community. My analysis traces these difficulties to the deployment context of the commercial Internet, which has made it much more difficult to change certain parts of the Internet's architecture than a purely technical analysis of the structural dependencies within the architecture would suggest. On a more abstract level, the analysis shows how the deployment requirements implied by a network architecture interact with the economic system in which the network is used to constrain the evolution of the architecture beyond what a real-options analysis would predict.

Chapter 5 focuses on the implications of architectural choices for the organizational structures in which the development and production of the system—and subsequent innovation—can take place. The analysis shows how an architecture affects how a single firm that has designed an architecture can organize the subsequent development and production of components to this architecture, and how the organizational options enabled by an architecture may ultimately influence the structure of the industries surrounding the architecture. Ultimately, these mechanisms determine whether independent economic actors other than the system's architect can develop and produce new components for the system, both initially and later. The chapter then applies these insights to the original architecture of the Internet. In particular, it discusses how that architecture affects who can develop new applications. In a network architecture based on the broad version of the end-to-end arguments, anyone who knows (or is willing to learn) how to program and has access to a computer connected to the Internet can develop new applications. An innovator does not have to be an employee of a firm or get outside funding to realize his or her idea for an application. As a result, the set of potential innovators is much

larger and more diverse than in network architectures that deviate from the broad version.

Chapter 6 explores how an architecture may affect the competitive strategies available to makers of complementary components. In particular, it analyzes how differences in a component's ability to monitor and control the execution of neighboring components may affect the strategies available to the maker or the users of the controlling component, and how this influences what benefits makers of complementary components can expect to realize. Although the chapter explores the effect of a component's ability to control other components in the context of the Internet's original architecture, the underlying theoretical framework is general enough to apply to other architectures. The chapter shows how architectural differences between the Internet's original architecture and a hypothetical architecture deviating from the broad version of the end-to-end arguments alter network providers' strategic options. It explores the conditions under which a network provider might have an incentive to discriminate against some applications, and it discusses recent instances of discrimination. This chapter also examines whether differences in market structure or laws may moderate the impact of architectural differences. It shows that network providers may have an incentive to exclude applications even if they face competition from other providers. The final section explores the effect of architectural differences on network providers' pricing strategies, such as their ability to charge users application-specific prices or to charge the providers of applications or content for the right to gain access to the customers of an Internet service provider.

Applications are the transmission belts that transform the general functionality of the Internet—transporting data packets from one computer to another—into something that creates value for users (and, in the process, creates value for society). Part IV of the book analyzes how changes in the architecture of the Internet affect the economic environment for innovation in applications and evaluates the public-policy consequences.

Chapters 4–6 provide a variety of insights into the relationship between architectures and design principles, on the one hand, and the economic system, on the other hand, both on a general level and with respect to the original architecture of the Internet and the design principles that shaped it. Chapter 7 draws on those insights to summarize how increases in the amount of application-specific functionality in the network's core change the economic environment for innovation at the application level. Thus, the chapter provides a framework in which the effects of deviations from

the broad version of the end-to-end arguments on application-level innovation can be assessed. It shows that as the amount of application-specific functionality in the network increases, independent application developers' incentives to innovate are reduced. At the same time, the range of potential innovators decreases, while the locus of control over the development and deployment of innovations in applications gradually shifts to the network owner. In addition, the chapter evaluates how such architectural changes affect other characteristics of the economic environment for application-level innovation, including network owners' incentives to innovate at this level and the costs associated with adopting new or improved applications.

Chapter 8 explores the effect of these differences on the amount of application innovation that will occur under different network architectures. In particular, drawing on theories of innovation and the histories of specific applications, chapter 8 examines how differences between decentralized and centralized environments for innovation and the associated differences in the number and types of innovators may affect the amount, the quality, and the character of application-level innovation. In a network architecture based on the broad version of the end-to-end arguments, anyone with access to an end host and with the ability to learn a programming language is a potential innovator. Innovators independently decide on their approaches to innovation; users independently choose which applications they want to use. In contrast, in architectures deviating from the broad version, network providers control which applications can be realized and used. At the same time, the set of potential innovators is smaller and less diverse; in the extreme case, network providers are the only remaining innovators. If there is uncertainty or consumer heterogeneity, a larger and more diverse group of innovators under an end-to-end architecture will produce more and better applications than a few network providers who control which applications are realized and used. If users' needs are heterogeneous, innovators in the end-to-end architecture will also produce more diverse applications that better meet users' needs.

Chapter 9 assesses how the various network architectures relate to the public interest and to network providers' private interests. Whereas earlier chapters focused on how network architectures affect innovation, this chapter looks more broadly at the economic and non-economic consequences of complying with or deviating from the broad version of the end-to-end arguments. The analysis shows that the public's interests in network architecture diverge from the network providers' interests,

creating a market failure regarding the evolution of the Internet's architecture.

The concluding chapter summarizes the book's main arguments and discusses the implications of the book's results for public policy and network design.

Overall, the book shows that the Internet's original architecture was based on a design principle that I call *the broad version of the end-to-end arguments*. This design fostered innovation in applications. Today, the Internet's architecture changes in ways that deviate from this principle. These changes reduce the amount and the quality of application innovation at significant costs to society: As a general-purpose technology, the Internet does not create value through its existence alone. It creates value by helping users do what they want to do, or by letting them do so more efficiently. Applications are the tools that let users realize this value. By reducing innovation in applications and limiting users' ability to decide how to use the network, these changes significantly reduce the Internet's usefulness and value for users, and, ultimately, for society as a whole. In addition, reducing innovation in applications limits the Internet's ability to contribute to economic growth. Finally, the Internet's ability to enhance individual freedom, its ability to provide a platform for better democratic participation, and its ability to foster a more critical and self-reflective culture are tightly linked to features resulting from the broad version of the end-to-end arguments. By removing these features, the changes to the architecture also threaten the Internet's ability to realize its social, political, and cultural potential. Though the broad version of the end-to-end arguments also has social costs, they are not large enough to justify sacrificing the social benefits created by the broad version.

While public interests suffer, network providers benefit from the changes. They control the evolution of the network, and it is highly unlikely that they will change course without government intervention.

Navigating the Book

The author of an academic book usually assumes that her readers know the relevant literature, and that she only has to add to it. I do not have this luxury. My argument crosses a number of disciplines and speaks to readers with a wide range of backgrounds. The book is designed to be accessible to all of them. In particular, one does not have to have a background in networking or in economics to understand it. As a result, the book is longer than it would be if it were targeted at readers in a single discipline, and it contains explanations that will be familiar to some

Table I.1

The end-to-end arguments from a technical perspective.

Understanding the End-to-End Arguments	Pages
What are the two versions of the end-to-end arguments?	57–75, 377–379
What are the differences between the two versions, and why should we distinguish between the two?	75–81
What is the trade-off underlying the broad version?	68–75, 355–371
Does the broad version differ from the layering principle?	104–105
Does the broad version prohibit quality of service?	106–107
Does the broad version require the network to be stupid or simple?	107
Do the end-to-end arguments prevent state in the network?	105–106
Does the broad version make it more difficult (or even impossible) to make the network secure?	366–368
How can you deviate from the broad version? What does it mean to make the network more controllable or more opaque?	286–287
The End-to-End Arguments and the Architecture of the Internet	Pages
How have the two versions shaped the Internet’s original architecture?	90–103, 110–112, 379–381
Is the Internet Protocol general enough?	142, box 4.3
How do the two versions relate to current developments in the Internet, such as the evolution of applications toward a more distributed structure or the proliferation of middleboxes?	383–387, 371–372
Is the trade-off underlying the broad version still appropriate today?	368–371, 355–356
Should the end-to-end arguments continue to guide the Internet’s architecture as a technical design principle?	388–389

readers but not to others. For example, engineers will already know about modularity, the layering principle, and the architecture of the Internet, and lawyers, economists, and management scientists will already know how transaction costs influence the boundary choices of firms. I hope that my headings and introductions will help readers recognize things they already know and help them get to what they do not know.

The book can be read straight through, from cover to cover. But one may also follow some of the more specific conceptual threads that run through the book. The first thread explores the end-to-end arguments from a technical perspective (table I.1). The second thread analyzes the social value of architectures based on the broad version of the end-to-end arguments and the resulting policy implications (table I.2).¹⁷ The third thread explores aspects of the approach to “architecture and economics” (box I.1) advanced by the book. Within the framework described in

Table I.2

The social value of the broad version of the end-to-end arguments.

The End-to-End Arguments and Their Relationship to the Internet's Original Architecture	Pages
What are the two versions of the end-to-end arguments, and why should we distinguish between the two?	57–81, 377–379
How have they shaped the Internet's original architecture?	90–103, 110–112, 373–381
The Effect of the Broad Version on Application Innovation	Pages
How does the broad version affect the cost of developing new applications?	137–148, 383–387
How does the broad version affect who can develop new applications?	204–213
Can innovators with no or little outside funding really produce successful applications?	204–213, 300–308, 334–345
How does it affect independent application developers if network providers can control the applications on their network?	215–281
Do network providers have an incentive to discriminate against or exclude unaffiliated applications?	218–273
Does competition among network providers remove this incentive?	255–264
What pricing strategies can network providers follow in a network where they control the applications?	217–218, 273–280
How does the economic environment for application innovation change if networks deviate from the broad version?	285–295, 383–387
How do differences in the size and diversity of the innovator pool and differences in control over application innovation and deployment affect the overall amount, quality, and type of innovation?	297–351
How important is it if innovators other than the network provider can independently choose their approach to innovation?	345–348
How important is user choice for application innovation?	349–351
How do the architectural differences among network architectures that are based on the broad version or deviate from it, taken together, affect the overall amount, quality, and type of application innovation under these architectures?	351–353
The Social Value of Network Architectures Based on the Broad Version	Pages
What are the social benefits associated with the broad version?	355–365
How important is application innovation?	355–360
How important is user choice?	349–351, 361–363
How does the broad version affect the Internet's ability to realize its political, social and cultural potential?	364–365
What are the social costs associated with the broad version?	365–368
How should we trade off among the social benefits and social costs?	368–371
Do network providers' private interests diverge from the public interest?	371–375
Policy Implications	Pages
What do these insights mean for public policy?	218–221, 264, 272–273, 387–392
Should legislators mandate compliance with the broad version?	388

Box I.1

Architecture and Economics

This book's approach to the study of architecture and innovation is an example of a more general approach to studying the architecture of complex systems, an approach I call *architecture and economics*. The approach understands architecture as one of several constraints on human behavior and uses economic theory (broadly defined) to explore the effect of these constraints. In its narrowest meaning, "architecture and economics" denotes efforts to understand how the architectures of complex systems influence, and are influenced by, the economic systems in which the complex systems are designed, produced, and used. As we saw above, the links between architectures and economic systems have important implications for how businesses compete and how public policy is made. While this book focuses on the effect of architecture on a specific economic activity (innovation), the underlying framework is general and can be used to understand the effect of architecture on a much broader range of human behavior. Viewed from this perspective, "architecture and economics" describes a much broader field of research—efforts that use economic theory to understand how architectures affect specific forms of human behavior, and, more generally, how architectures influence, or are influenced by, economic, social, cultural, or political systems.^a I use the term in the broader sense.^b

a. I do not mean to imply that this type of research has not existed so far. Most research in this area, however, focuses on specific architectures or design principles. For example, a large body of literature in management science and engineering explores the economic effect of modular and integrated architectures in the design of physical products. (See note 4 to this chapter for references to this literature.) There is some research in software engineering that examines the economic effect of software architectures (e.g., Sullivan et al. 1999; Boehm and Sullivan 2000; Sullivan et al. 2001; Erdogmus et al. 2002). Representative examples of legal scholarship exploring the effect of the Internet's architecture are Lessig 1998, Lemley and Lessig 1999, Lessig 1999a, Lessig 2001, Wu 2003a, Benkler 2006, Balkin 2008, and Zittrain 2008. There is, however, no established field of research that connects work in this field under a common umbrella, nor is there an accepted framework or methodology for exploring these issues. Among others who have advocated for a more unified approach to the study of the architectures of complex systems are Baldwin and Clark (2006b) (who advocate a "science of design" that would cover questions very similar to the ones outlined in the text) and van Schewick (2004).

b. "Law and Economics" has a similar dual meaning. It is used to describe the study of the relationships between law and the economic system, but it also describes, more broadly, efforts to understand the effect of law on human behavior using economic theory. For an overview of the different schools of thought within this field, see Mercuro and Medema 2006.

Table I.3

Architecture and economics.

Factors Mediating the Effect of Architecture	Pages
<hr/>	
How do non-architectural constraints interact with architecture?	
Theory	26–28, 31
Examples	196–197, 218–221, 264, 272–273
How do characteristics of the actors exposed to the architecture affect the impact of architecture?	
Theory	30
Examples	204–213, 298–345
How do actors' relationships with other actors affect the impact of architecture?	
Theory	31
Examples	133, 218–221, 255–264, 212–213
<hr/>	
The Effect of the Economic System on Architecture	
<hr/>	
How is architecture influenced by economic systems?	
Theory	3, 23–26, 28, 32, 389
Examples	151–163, 371–372, 389–392
<hr/>	
The Effect of Architecture on Activities Other Than Innovation	
<hr/>	
How can the architecture of a complex system affect the economic, social or political systems in which the complex system is used?	
Theory	28, 361–362
Examples	359–361, 362–365
<hr/>	
Architecture and Economics: Implications	
<hr/>	
What are the implications of the economic impact of architecture for businesses?	
Theory	3
Examples	371–372, 389–392
What are the implications of the economic impact of architecture for law and policy?	
Theory	3–4
Examples	388
<hr/>	

chapter 1, the book focuses on the effect of one constraint (architecture) on one specific activity (innovation), and sets aside consideration of other factors (such as the effect of non-architectural constraints, or the mechanisms by which architectures are influenced by economic systems) that are relevant within the framework. This thread ties together the portions of the book that touch on these other factors (table I.3). In doing so, it complements the detailed analysis (in chapters 4–8) of how architectures—particularly that of the Internet—affect innovation.

I Foundations

1 Architecture and Innovation

Engineers focus on technology; managers, legislators, regulators, and the lawyers and economists who advise them focus on economic systems. This common separation of responsibilities may blind us to the links between the two: the architecture of a complex technical system—the description of its basic building blocks—fundamentally influences and is influenced by the economic system that drives the development, production, and use of the system. Failure to appreciate these links may result in policies or strategies that do not realize the economic potential of technical systems. To understand and exploit these links, engineers need to learn a bit about the economics that support their innovation, while managers, lawyers, economists, regulators, and legislators need to learn something about the technologies they aim to manage. This chapter introduces a theoretical framework for understanding how technical architectures relate to economic systems and, more specifically, for understanding how architectures affect innovation. It explains architectures and design principles as concepts, and it highlights the mechanisms by which architectures influence and are influenced by economic systems.

Architecture

Scholars of Internet policy may already feel familiar with the notion of architecture. Since the publication of Lawrence Lessig's book *Code and Other Laws of Cyberspace*, the idea that the 'architecture' of the Internet regulates behavior in cyberspace in a way that is similar to law has gained widespread popularity.¹ The potential effect of this regulation on civil liberties is often captured in the slogan "Architecture is politics."²

In the context of this book, this familiarity is misleading. When texts on Internet policy refer to architecture, they usually mean the software and hardware that make up the Internet.³ In this terminology, the word

Box 1.1

The Place of Architecture Design in the Product-Development Process

The development of a complex system proceeds from the requirements specification to the design to the implementation (for software systems) or production (for physical systems) of the system.^a The requirements specification defines the functional, business, and quality requirements the system must meet. The design creates a detailed specification—a blueprint—from which the system can be built: it completely describes the system, including instructions about how to produce it. The implementation or production of the system results in an executable program, or, in the case of physical products, a usable product. The architecture of a system is produced as part of the design phase in an intermediate step between requirements specification and detailed design.^b

a. These activities are part of every software-development project or (in the case of physical products) product-development project. Models of software-development or product-development processes differ in the amount of overlapping, interleaving, and iterating among these activities. For an overview of different models of the software-development process, see van Vliet 2000, chapter 3. For an overview of different approaches to product development, see Ulrich and Eppinger 2000, pp. 13–24, 235–272.

b. The text describes the sequence of activities in a development project that follows best practices. In reality, some software systems are developed without formal specification of an architecture. Sometimes an architecture emerges over time, as the people working on the system come to a better understanding as to what rationales or design principles have implicitly driven the evolution of the system and make these explicit in a formal description of the architecture. The original architecture of the Internet seems to have emerged in this way.

‘architecture’ refers to implemented technical systems—i.e., the networks, computers, and programs that constitute the operational Internet. ‘Architecture’ is synonymous with “the system.”

In computer science, and throughout this book, ‘architecture’ has a very different meaning: it denotes the fundamental structures of a complex system as defined during the early stages of product development (box 1.1). Similar to the way the architecture of a house is different from the house itself, the architecture of a system is not the final, working system; rather, it is a description of the system’s basic building blocks.

When designers create a complex system, they typically seek to reduce the problem’s complexity by decomposing the system into pieces that work together to provide the required functionality. The architecture

describes the early design decisions about the decomposition of the system. It contains all the information that is necessary to understand how the chosen collection of components will satisfy the functional, business, and quality requirements. In short, the architecture describes the components of the system, what they do, and how they interact.

More precisely, the architecture is a high-level system description that specifies the components of the system, the externally visible properties of the components, and the relationships among components. The externally visible properties of a component are the features that other components must know about if they are to interact with it. They include the services provided by the component, their performance characteristics, fault handling, use of shared resources and other characteristics.^{4,5} Architectures of distributed or parallel systems also describe how software components map to hardware elements,⁶ or, in the case of physical products, how components map to the physical elements of products.⁷ For example, the architecture of the complex system “personal computer” names the components (e.g., central processing unit, keyboard, monitor), describes what they do (e.g., the central processing unit processes software program instructions; the keyboard lets users give input to the system; the monitor displays the output) and describes how components interact (e.g., peripherals may connect to the rest of the system using the Universal Serial Bus standard). But the architecture does not describe how the keyboard operates internally to record and transfer input, or how the monitor displays information transferred to it by the rest of the system. In other words, the architecture provides an abstract view of a system as a collection of “black boxes,” describing how they behave and interact but not how they work. In particular, the architecture ignores questions of implementation such as algorithm design and data representation.

Architecture is a hierarchical concept.⁸ Components at the highest level of a system’s architecture can be decomposed into a set of interrelated subcomponents that may also have architectures.⁹ Decomposition can continue until the lowest level of elementary subsystem is reached. In the case of the personal computer, the monitor can be broken down further into interacting subcomponents (the screen, the case, and so on). Going up the hierarchy, a personal computer may be part of a larger complex system, such as a home network or the Internet.

Design Choices

Many different architectures can usually satisfy a complex system’s functional requirements.¹⁰ This is because changes to the structure or behavior of one component can often be offset by changing its interactions with

other system pieces so that the system's overall functionality stays the same. This interchangeability is especially important in software engineering, because software architectures are less limited by real-world constraints such as the laws of nature that constrain the design of physical products. Apart from a few technical constraints, software designers are free to shape the architecture according to the desired goals. Thus, although the capability of a system to meet the functional requirements depends on a correct architecture, it does not depend on specific architectural choices.

But a number of other qualities of a system *do* depend on architectural choices. Some of these qualities (e.g., performance, availability, reliability, usability, security) have to do with the operation of the system; others (e.g., modifiability, reusability, testability) have to do with the development or maintenance of the system.¹¹ For example, modifiability (being able to make changes quickly and cost-effectively) depends on how many components must be modified to implement a change.¹² A system is more modifiable if functions that are likely to change together are concentrated in one or a few components. Since the allocation of functionality to components is an architectural issue, the modifiability of a system is determined by its architecture.

Performance, on the other hand, is only partly affected by architectural decisions. The performance of a software system, for example, depends in part on the amount and the complexity of inter-component communication.¹³ The amount and the complexity of interaction among components are, in turn, determined by how functionality is distributed among components during the design of the architecture. Thus, the performance of a software system is attributable in part to architectural choices. At the same time, some non-architectural decisions (e.g., the choice of algorithms and their implementation) also affect the performance of the system.

Qualities that depend on specific architectural choices are very difficult, if not impossible, to realize at later stages of the development process if the architecture does not support them. Thus, choosing an architecture that can support the desired qualities is crucial for meeting the quality goals of a system.¹⁴ In this sense, "Architecture is quality."

Unfortunately, qualities do not exist in isolation. Aspects of the architecture that promote one quality may have a negative effect on another quality.¹⁵ For example, fault tolerance in software systems is usually achieved by replicating critical components. When one version of a component fails, an identical version takes over. Additional components, however, may compromise security by offering additional points of attack. Thus, an

architectural solution that increases fault tolerance may reduce security.¹⁶ Owing to the interactions among qualities, it is impossible to maximize all qualities simultaneously—choosing an architecture requires trade-offs, prioritizing some qualities at the expense of others. Deciding which attributes are more important than others is not an architectural choice—this decision depends on the requirements that the system has to meet and must be made by the organization who sets these requirements. Usually, this will be the organization that wants to use or sell the system.¹⁷

Design Principles

Owing to the complex interactions among architectural requirements of different qualities, it is difficult to translate a set of quality requirements into an appropriate architecture that realizes these requirements. Design principles help us move from qualities to architectures.

A design principle describes known connections between architectural choices¹⁸ and the characteristics of the resulting architecture.¹⁹ It describes the architectural means by which a specific set of trade-offs among conflicting qualities is realized. Like a recipe, a design principle describes how to design an architecture for a system with specific quality characteristics, and, like different versions of a dish, the resulting architectures will differ depending on the design principles that were used to create them.²⁰ Which design principle (or combination of principles) should be used depends on what qualities are desired.²¹ Thus, system architects select the design principles that will give them the combination and the prioritization of features they want to realize.

Design principles are still important after the initial architecture is created because they guide the detailed design that follows.²² Continuing to use the original design principles is especially critical if an architecture exists for a long time. When a system has to evolve in response to additional requirements or changes in technology, adding new parts to the system without considering the original design principles can disturb the balance among conflicting quality attributes that was struck in the original architecture. Adhering to the original design principles ensures that the system will continue to have the qualities and properties that the architecture originally prioritized.

The Relationship between Architecture and the Economic System

The importance of architecture transcends technical considerations: the architecture of a system is tightly tied to the economic system in which it

is developed, produced, and used, and to the incentives of the actors who may engage with it. The relationship between architecture and the economic system can be understood in a broader analytical framework that economists use to understand the evolution of the economy as a whole. The framework has two components: first, a set of actors and the relationships among them; second, the constraints under which the actors operate.

In the American financial sector, for example, the actors—both individuals and organizations (i.e., consumers, banks, regulatory agencies, firms)—are connected through a network of social, political, and economic relationships. Sometimes these actors interact through markets: consumers take out loans from banks or deposit money in savings accounts; buyers and sellers trade shares and stocks in markets like the stock exchange. Actors may also interact through other governance structures, as when banks participate in associations such as the American Bankers Association to better pursue their interests.

According to economic theory, the structure of the economic system (the actors active in the economy, the relationships among them, and the governance structures through which they interact) and the behavior of the actors in the economic system emerge from the choices individuals make as they pursue their own interests within a set of constraints. The panoply of possible constraints might include laws, social norms, the natural and technical environments in which the actors exist and economic constraints such as prices or available assets.²³ Constraints delimit the options available to individuals and organizations, and influence the costs and benefits associated with these options. In the financial sector, for example, laws define the structure, responsibilities, and power of the regulatory agency. They define which entities can become banks, and what commercial activities they are allowed to engage in. By imposing reporting requirements and capital-structure requirements on banks, financial regulations can make their operations more or less costly.

By enabling different actions or imposing different costs and benefits, constraints shape firms and markets and affect economic activity (for an example, see box 1.2).²⁴

Thus, economists distinguish between the *framework* provided by the constraints and the *system* of actors and relationships that emerges as a consequence of these constraints. The economic system evolves as a result of the dynamic interactions between economic actors and the constraints they face. On one hand, economic structures and behaviors are shaped by

Box 1.2

The Effect of Constraints in the Financial Sector

From the Great Depression until the 1990s, financial regulations in the United States prevented banks from taking ownership interests in firms and did not let firms have interlocking directorates—that is, directors serving on the boards of competing firms.^a German law did not impose similar constraints.^b These differences resulted in very different relationships between banks and firms in the two countries. German firms tended to be funded by banks and elite families, who could influence corporate decision making, while German public financial markets were relatively insignificant.^c In the United States, where there was a clear separation between banks and firms, publicly traded firms were funded by financial markets. Researchers have argued that the German structure encouraged more risk-taking, because the banks providing the financing had better information than the American public about the firm's plans and more ability to influence those plans.^d In addition, the combination of restriction on interlocking directorates and prohibition of mergers meant that American firms tended to diversify so they could handle more aspects of a business. In Germany, where interlocking directorates were allowed and there were fewer restrictions on cartels, firms remained smaller and less integrated.^e

a. The Glass-Steagall Act of 1933 restricted US banks from ownership in non-finance firms. Its core provisions were repealed in 1999 by the Gramm-Leach-Bliley Act (U.S. Senate Committee on Banking 1999). The Clayton Anti-Trust Act of 1914 restricts interlocking directorates in the US, i.e., having a director sit on the board of more than one corporation at a time. This restriction applies only to competing corporations and to firms that have "capital, surplus, and undivided profits" exceeding a jurisdictional "trigger," which was recently raised to about \$25 million (Clayton Antitrust Act, U.S. Code 15 (2000), §19; Federal Trade Commission 2008). The prohibition is still in effect. A recent *USA Today* survey showed that some non-competing firms have highly interlocked directorates (Krantz 2002).

b. Fligstein and Freeland 1995, pp. 33–37.

c. Roe 1994, pp. 169–177; Allen and Gale 2001, pp. 71–76; Calomiris 2000, pp. 212–219.

d. Calomiris 2000, pp. 213–219. This argument is heavily contested. For a flavor of the debate, see Edwards and Fischer 1994; Easterbrook 1997; Franks and Mayer 1997; Carney 1997.

e. Fligstein and Freeland 1995, pp. 33–37.

individuals' and organizations' responses to constraints. On the other hand, if economic actors feel that their interests are better served by a different set of constraints, they may work toward changing them through lobbying, voting, and so on. Naturally, some constraints may be easier to change than others.²⁵

To complicate matters further, constraints do not operate autonomously; rather, one constraint may influence the effectiveness and the evolution of another. Two constraints operating simultaneously may support or contradict each other, ultimately reinforcing or weakening the other constraint's effect on an actor.²⁶ For example, laws aligned with social norms may last longer than laws that oppose them; conversely, laws can strengthen or weaken social norms.²⁷ Constraints imposed by the natural and technical environment may operate analogously: technical systems may effectively displace laws, and changes in technology can undercut a law's effectiveness even if the law's text remains unchanged. Laws can support architectural constraints by criminalizing attempts to circumvent them, or can weaken their effectiveness by prohibiting their use (for some examples, see box 1.3).

Box 1.3

Interactions between Legal and Technical Constraints

Legal and technical constraints may interact in different ways. Technical systems may effectively displace laws. Constraints imposed by a system influence what users can or cannot do with the system. If these constraints disable or discourage behavior that would be necessary in order to exercise certain rights, the system changes the effectiveness of these rights: while using the system, users cannot exercise these rights even if they would be legally entitled to do so. Thus, the system determines which rights are *effectively* available to the users of the system, effectively displacing the law.

For example, "digital rights management systems" control access to and use of copyrighted digital content. A digital rights management system that prohibits *any* copying not only protects the content against illegal copying, it also makes it impossible for a user to make copies that are justified under the fair-use doctrine of copyright law. Thus, although the law stays the same, this system effectively displaces the fair-use doctrine of copyright law. A system with a different set of constraints may disable only illegal copying while enabling copying for fair-use purposes, respecting the fair-use doctrine of copyright law.^a Thus, the constraints established by these systems effectively displace the law to different degrees.^b

Box 1.3

(continued)

Changes in technology can reduce a law's effectiveness even if the law's text is not changed. For example, the Internet has changed the effectiveness of copyright law. Before the Internet, only a few intermediaries had the technological means to copy and distribute copyrighted works on a large scale. As a result, copyright could effectively be enforced by finding and holding liable those intermediaries who engaged in illegal copying. In contrast, the Internet lets many different people copy and distribute digital content at almost no cost and with no deterioration in quality. In addition, peer-to-peer file-sharing networks let users engage in mass distribution of illicit copies. Thus, the Internet and peer-to-peer file-sharing networks greatly increase the number of potential copyright violators by eliminating the need for intermediaries in the copying and distribution of copyrighted works. As a result, those enforcing copyright can no longer focus on a limited number of intermediaries but must direct their efforts toward a huge number of individual violators, which makes copyright enforcement considerably more difficult and costly. Thus, although the copyright laws have not changed, copyright owners today are less effectively protected against infringement than they were before the Internet.^c

Law can support architectural constraints by criminalizing attempts to circumvent them, or can weaken their effectiveness by prohibiting their use. For example, the American Digital Millennium Copyright Act makes it a felony to circumvent copyright-management schemes. By legally protecting copyright-management schemes from circumvention, the law strengthens the constraints imposed by them.^d

- a. On technical solutions for digital rights management systems that may enable copying for fair-use purposes, see Burk and Cohen 2001.
- b. See Lessig 1999b, pp. 521–530.
- c. See Wu 2003b, pp. 131–136.
- d. See Digital Millennium Copyright Act, U.S. Statutes at Large 112 (1998): 2863, §1201(a)(1). For a critical discussion of this provision, see Lessig 1999b, pp. 538–541.

One type of constraint can also be used to change another. In particular, economic actors may try to use legal constraints to change other, non-legal constraints.²⁸ For example, laws can influence prices by imposing taxes or granting subsidies, or can affect the structure of a market by regulating entry into an industry.²⁹ Laws can change norms, too: a law may provide financial incentives to ignore a norm, or may weaken a norm by creating tension between the existing norm and the norm of engaging in law-abiding behavior.³⁰ Laws can affect the technical environment by regulating technical systems, or by encouraging the development of specific technologies by letting public entities participate in standard setting, funding the development of desired technologies, or restricting public procurement to the technologies the state wants to foster.³¹

The relationship between architecture and the economic system can be understood within this analytical framework. Like laws or social norms, architecture shapes human behavior by imposing constraints on those who interact with it. By imposing constraints, the architecture of a complex system affects the economic system for its development, production, and use—that is, the actors who will develop, produce, or use the system, the relationships among them, the governance structures they use to interact with one another, and the behavior of these actors. And by changing existing architectures or creating new ones, economic actors can change the constraints that architecture imposes.

The Relationship between Architecture and Innovation

This book explores the relationship between architecture and the economic system in the context of a specific economic activity: innovation.³² The word ‘innovation’ refers to creating or improving goods, services, or methods of production.³³ The book focuses on innovation that can be influenced by architectural features—developing components for a new system, improving existing components or developing new ones, changing an architecture to enable new functionality, or creating new architectures. These innovations drive a system’s initial development and its continued evolution.

The literature on innovation often uses the term to describe the innovative activities that firms perform to increase their profits.³⁴ To appreciate more fully the relationship between architecture and innovation, this book adopts a broader definition, using the word ‘innovators’ to denote a wide range of actors—including individuals, groups, firms, and other organizations—whose actions and products may be motivated by various economic or non-economic concerns.

The architecture of a system influences economic structures and behaviors regarding the development and evolution of the system, and affects the amount and kind of innovation that might occur. In particular, architectural features influence which actors may develop and change a complex system, the incentives under which they act, and the governance structures through which their activity is organized. Conversely, innovation may change existing architectures or create new ones.

Architecture affects behavior by imposing constraints on the actors who interact with it. Constraints, in turn, influence behavior by defining and limiting available courses of action, and by influencing the costs and benefits associated with specific actions. Actors respond to changes in incentives in observable, predictable ways.³⁵ By creating incentives and disincentives, or by making some actions available and precluding others, particular architectures may lead to specific firm structures or market structures, or may enable different levels of innovative activity.

Imagine that several potential innovators each have an idea for a new system component and have to decide whether, and how, to build it. Suppose that under Architecture A the only actor able to design and build new components is the system’s original architect. As we will see in chapter 5, this may happen if the architecture has been designed using an integrated approach. In this scenario, the potential innovator’s sole option is to sell his idea to the system architect. Suppose that under Architectures B and C developers are able to design and implement the innovation themselves, but that the costs of realizing the innovation are higher for all innovators under Architecture C than under Architectures A and B (table 1.1). Owing to these differences among architectures, the potential innovators may make different choices when deciding whether to realize their innovative

Table 1.1
Effects of different architectures.

Aspects influenced by architectural features	Architecture A	Architecture B	Architecture C
Actors able to realize the innovative idea, i.e., design and build a new component	System architect	System architect or potential innovator	System architect or potential innovator
Costs of realizing the innovative idea, i.e., designing and building a new component	Low	Low	Higher

idea. For some innovations, a potential innovator may expect the benefits to outweigh the costs under Architecture B, but not under Architecture C; these innovations will not be realized under Architecture C. Some innovations that may be realized by an independent innovator under Architecture B may not be realized under Architecture A, perhaps because the independent innovator did not enter into negotiations with the system architect or because the negotiations were not successful; if they are realized under Architecture A, they will be realized by the system architect and not by the person with the original idea. As a result, who realizes innovative ideas (the system architect or the developer with an innovative idea) and how many innovative ideas are realized will differ depending on the architecture.

Understanding that architecture may constrain potential innovators is only the first step toward explaining the relationship between architecture and innovation. An economic actor's reaction to a constraint depends on the characteristics of the economic actor, on the other constraints on the actor, and on the actor's existing relationships and anticipated interactions with other actors.

First, architectural constraints take effect through the reactions of those exposed to them—that is, they induce behavior rather than enforce compliance. The economic actors can be individuals or organizations—“groups of individuals, bound together by a common purpose”³⁶ and founded to achieve certain social, cultural, political, or economic objectives. Certainly organizations consist of individuals, each pursuing his own interest within the framework of constraints imposed on him, but in general this book treats organizations as singular units with resources, cost structures, goals, motivations, and cost-benefit assessment processes that influence how they make decisions. Processes and structures *within* organizations will be discussed only when they are relevant to the effect of architecture on the organization of innovation (chapter 5) and to the reactions of organizations to opportunities for innovation (chapter 8).

Actors may have different resources, cost structures,³⁷ goals, motivations, and cost-benefit assessment practices; consequently, their perception of costs and benefits is idiosyncratic, and they may react differently to the same architectural constraint.³⁸ An actor who hopes to profit from selling her innovation may be disinclined to innovate if she is operating within an architecture that reduces her potential sales revenue. In contrast, an actor who innovates to use the innovation himself may remain unaffected by the same architectural change (so long as the architecture lets him use the innovation). Thus, the effect of architecture depends on the characteristics of the actors exposed to it.

Second, architecture is not the only constraint on potential innovators. Laws, social norms, market conditions, and the natural and technical environments also influence innovators' actions.³⁹ Like architecture, these constraints delimit the available choices and affect the corresponding costs and benefits. Intellectual-property laws, for example, influence the expected benefits of an innovation by conferring a temporary monopoly on the innovator but impose costs on subsequent innovators who want to build on the original innovation. By striking different balances between the interests of the original innovator and those of subsequent innovators, alternative intellectual-property regimes that differ in duration and in scope facilitate different types and amounts of innovative activity.⁴⁰ Antitrust laws shape the range of legal strategies available to an innovator once her innovation reaches the product market, thus affecting the benefits an innovator can expect to gain. Social norms that encourage risk-taking and challenging conventions but also tolerate failure may be especially conducive to innovation. The structures and operations of capital and labor markets (both heavily influenced by laws and norms) may affect the ease with which a potential innovator can acquire the financial and human capital necessary to realize his innovation.

Finally, a potential innovator's response to a constraint may be influenced by his extant relationships and anticipated interactions with other actors. For example, if a market has only a few potential or anticipated innovators, each firm will base its level of investment on the level of investment it expects from the others. A firm that has long collaborated with another firm may be able to work with that firm on innovative projects that require constant interaction, because the trust and coordination mechanisms resulting from the long-term relationship may enable it to overcome the coordination problems that usually prevent firms from coordinating closely interdependent activities across firm boundaries.⁴¹

Thus, the same architecture, operating in environments with different actors and constraints, may enable very different types and levels of innovative activity.⁴² For example, consider an architecture that encourages entrepreneurial innovation but requires more capital than entrepreneurs usually are able to invest individually. In this case, we would expect to see more innovation from entrepreneurs in an environment in which funding from venture capitalists is available than in an environment in which similar financial actors do not exist.⁴³ All this implies that the effect of a specific architecture on innovation cannot be determined without considering the characteristics of the actors exposed to it, the other constraints under which they operate, and the actors' existing or expected relationships.

Just as architectures shape economic behavior, economic behaviors may shape the evolution of architectures.⁴⁴ This is most apparent when an innovation involves changing an architecture or building a new one. There is, however, a more general link between economic actors and architectures: architectures are created by economic actors. Functional requirements rarely determine the architecture of a system fully, since there are usually several alternative architectures that meet the functional requirements. Thus, an architect makes many choices not formally prescribed by the functional requirements. Like other decisions, these choices—and, consequently, the evolution of the architectural design—will be influenced by the architect's individual characteristics, by his relationships within the economic system, and by the constraints he faces (laws, norms, economic constraints, and so on). Different constraints, actors, and relationships may therefore result in different architectures. Furthermore, since economic considerations shape actors' decisions, system architects will tend to favor architectures that support their own economic interests. At least in part, then, architectural designs hinge on the choices of economic actors, all of whom pursue their own interests under their particular constraints. Thus, the evolution of architectures is partly endogenous.

In summary, an analytical framework that explains the relationship between architecture and the economic system includes a set of actors, all subject to constraints imposed by system architectures, existing laws, prevailing social norms, the actors' technological and natural environments, and the well-known constraints described in traditional economic theory, such as prices and the resources available to actors. When operating simultaneously, such constraints interact and influence one another in complex ways: they can influence one another's evolution or removal, and they can emerge from purposeful and idiosyncratic actions of a diverse set of economic actors.

Within the framework outlined above, this book focuses on the effects of one constraint (architecture) on one type of economic behavior (innovation). To this end, the book focuses on a particular architecture (the original architecture of the Internet, which governed the Internet from its inception to the early 1990s⁴⁵), abstracting from it the circumstances and motivations that led to its creation. To highlight the impact of the end-to-end arguments (the design principles on which the original architecture of the Internet was based), the original architecture of the Internet will be compared with hypothetical architectures that deviate from the end-to-end arguments in different ways. To focus on the specific effect of architecture,

the analysis assumes that while the architecture changes, everything else (i.e., the group of actors exposed to the architecture and the other constraints under which they operate) stays the same.⁴⁶

Understanding architecture as a constraint on economic actors makes it possible to use economic theory to predict how economic actors will react to architectural features. The framework developed so far does not depend on specific models of human behavior; instead, it is compatible with several models, including the rational-actor model, models of bounded rationality, and models used in behavioral economics.⁴⁷ Since the appropriateness of a model depends on the situation under analysis, modeling will be discussed more thoroughly in the chapters that follow.

This chapter has outlined the general mechanism through which architecture influences innovative behavior. To predict how a specific architectural structure will affect an innovation, we must understand exactly how architecture constrains economic actors. How do design principles and specific architectural features shape the costs and benefits associated with a certain innovation? How do they affect who can design and build an innovation? Before we can explore these questions, we need to understand the original architecture of the Internet and the design principles that led to it. Those will be the subjects of the next two chapters.

II The End-to-End Arguments and the Original Architecture of the Internet

2 Internet Design Principles

Design principles shape architectures by imposing rules that system architects must follow.^{1,2} They may constrain how a system is decomposed into components, how functionality is distributed among these components, or how components may depend on one another. Since different design principles impose different constraints, the technical characteristics of the resulting architectures may differ depending on the design principles that were used to create them. The technical differences among architectures, in turn, may translate into different economic constraints on those who design, produce, and use the resulting systems. Thus, before we can understand how architectures (and the design principles that shaped them) constrain *economically*, we need to understand how they constrain *technically*.

To this end, chapters 2 and 3 introduce the architecture we will use to study the effect of architecture on innovation—the original architecture of the Internet—and the design principles that were used to create it. Though policy debates often focus on the end-to-end arguments, they are not the only design principles on which the Internet’s original architecture was based. Before the end-to-end arguments can be applied, other, more fundamental design decisions have to be made. The end-to-end arguments guide how functionality is distributed in a multi-layer network; thus, the layering principle has to be applied first. Layering, in turn, is a special case of modularity. Together, these design principles—modularity, layering, and the end-to-end arguments—shaped the Internet’s architecture and its technical and economic characteristics.

This chapter describes the constraints that each of the three design principles imposes on the design of an architecture and highlights the trade-offs that underlie each of these principles. Because of the high level of confusion surrounding the end-to-end arguments, they are analyzed in detail. As will become apparent, some of the confusion can be attributed

to the silent coexistence of two different design principles under the same name: the narrow version and the broad version of the end-to-end arguments. While the narrow version only applies to some functions within a system, the broad version applies to the complete functionality of a system. The two versions constrain designers to different degrees, which makes it important to distinguish between them.

Modularity

Architectures differ fundamentally in the degree to which their components are “loosely” or “tightly” coupled. Coupling is a measure of the degree to which components are interdependent. Modularity is a design principle that intentionally makes components highly independent (“loosely coupled”).³ Components of modular designs are called *modules*. When designing a modular architecture, system architects decompose the system in a way that minimizes dependencies among components. The remaining interdependencies and points of interaction between components are addressed and resolved in the course of the architectural design process; the resulting detailed interface specifications are not allowed to change during detailed design.⁴ There are degrees of modularity because modular architectures may differ in the degree of coupling between components.⁵

The goal of modularity is to create architectures whose components can be designed independently but still work together. (On other types of modularity, see box 2.1.) To make components independent, and to maintain their independence, modularity employs abstraction, information hiding, and a strict separation of concerns.⁶

A modular approach to design distinguishes between two types of information.⁷ Information that is relevant to more than one module is completely specified as part of the architectural design process and is not allowed to change during detailed design. This information is available or “visible” to all designers, and is therefore called the architecture’s *visible information*. In particular, this includes all the information modules need to work together, such as the information necessary to use a module or to otherwise interact with it. For each module, this information is completely defined in detailed interface specifications that describe all interdependencies or points of interaction between that module and other modules and specify how they will be resolved.^{8,9} Thus, a module’s visible information provides an external view of the module that lets other designers treat the module as a black box, letting them abstract from the complexity of the module.

Box 2.1

Modularity in Design, Production, and Use

The modularity described in the text is not the only type of modularity. Modularity as described in the text creates a high degree of independence among components with respect to design. In the literature on physical products, the word ‘modularity’ is sometimes also used to describe approaches that intentionally create a high degree of independence among components with respect to production or use. To distinguish between these different forms of modularity, they are called *modularity in design*, *modularity in production*, and *modularity in use*. If a product is modular in design, its components can be designed independently. If it is modular in production, its components can be produced independently. If it is modular in use, users of the product can replace or “mix and match” components at a later stage. A specific product can but does not have to exhibit all forms of modularity. Lego blocks are modular in design, production, and use. Personal computers and stereo systems often are, too. Automobiles are modular in production, but not modular in design and mostly not modular in use.^a

a. See Baldwin and Clark 2000, p. 78. For an overview of the literature on modularity in design, production, or use in assembled hardware products, see Fixson 2001, pp. 23–27.

In contrast, information that affects only one module—the inside of the black box—is hidden from everyone except its designers; it is called the architecture’s *hidden information*. This information is not specified during the design of the architecture, but is determined during the detailed design phase; it is free to evolve within the framework provided by the architecture’s visible information until the detailed designing ends.

To ensure that a module remains a black box, access to its data and services is restricted to its interface. Designers working on other parts of the system are not allowed to make any assumptions about a module beyond its visible information. These constraints prevent the designers of a module from relying on another module’s hidden information when deciding how to develop their own module’s hidden information; they are allowed to rely only on the architecture’s visible information. Since the design of their own module does not rely on the hidden information of other modules, the designers of a module are not affected by changes in other modules’ hidden information, as long as the other modules’ visible information stays the same. This means that the detailed designing of

different components—i.e., the designing of the components' hidden information—can proceed independently.¹⁰

For example, personal computers are based on a modular architecture. The interfaces between the peripheral devices (e.g., printers and screens) and the rest of the system are completely specified by the industry's interface standards. These standards describe the physical shape of the plug and the format and meaning of data transferred through the plug. As a result, the designers of a peripheral device have to worry only about their own device. They do not need to know how the computer will treat the data received from their device. As long as their design follows the interface specification, they know the computer will be able to treat their data properly. If the designers of the computer and the designers of peripheral devices abide by the interface specifications, the design of the computer and of particular peripheral devices can proceed independently.

Modularity in design reduces complexity by allowing component designers to treat other modules as black boxes. By shielding the rest of the system from internal changes to a module, modularity increases designers' flexibility regarding the detailed design of their own modules and lets the detailed design of different modules proceed independently. A modular approach to design may also improve the correctness of the system (box 2.2). The positive impact of modularity in design on the modifiability of the resulting system is discussed in chapter 4.

Modularity is not without costs. An architecture's visible information is not allowed to change during detailed design and is difficult to change later.¹¹ Thus, modularity makes it more difficult to experiment on the architecture's visible information (such as a module's interfaces) in order to facilitate experimentation on the architecture's hidden information (such as a module's internals). In a way, the inflexibility of the visible information is the price for the flexibility with respect to the architecture's hidden information. To prevent this from becoming problematic, a good modular architecture is designed so that parts that may benefit from improvement or variation are enclosed within a module, so that parts that are likely to change together belong to the same module.¹²

In addition, designers' flexibility with respect to an architecture's hidden information is limited by the constraints imposed by the architecture's visible information.¹³ Although interfaces free designers from having to coordinate their detailed design decisions with designers of other components, they introduce a new constraint on the designers of affected modules: all subsequent design is bound by the interface specifications.¹⁴ The more detailed the interface, the more severe the constraint.

Box 2.2

How Modularity Affects the Correctness of Software Systems

When decomposing a system while designing a modular architecture, system architects seek to minimize interdependencies among components in order to reduce the number of design decisions that must be coordinated across components. In the language of software engineering, they strive for high cohesion and low coupling^a: the internal elements of a module should be strongly interconnected, while the connections to other modules should be weak. Such a design has a positive effect on the correctness of the resulting system: Weakly coupled components are easier to understand, easier to build, and easier to test.^b The strength of coupling between two modules influences the complexity of the corresponding interfaces, because a module's connections to other modules are completely addressed and resolved in that module's interface specifications.^c If components are weakly coupled, the connections to other modules are weak, so little communication or interaction across modules is necessary; this results in simple interfaces. A module with a simple interface can usually be understood and built independent of the context in which it is used. Highly independent modules usually can be tested separately.

In contrast, highly coupled components work together closely and must be understood as a whole. Strong connections among modules require interfaces to be complex and highly specific to their interactions. Since the effect of complex interfaces is often difficult to grasp, errors in subsequent development are more likely to occur. In addition, when components interact in complex ways, errors introduced by one component may cause other components to fail.^d Diagnosing problems may be difficult and time-consuming, since testing components independently is often impossible.

All these factors result in a higher correctness of weakly coupled architectures. Empirical studies show that modular systems with low coupling are less error-prone than architectures whose components are highly coupled.^e

a. Whereas 'cohesion' describes the strength of intra-module connection (i.e., the degree to which a component's sub-elements are related to one another), 'coupling' describes the strength of interconnection between two components. "Highly coupled" components are joined by strong interconnection; "loosely coupled" components are joined by weak interconnections. (See Yourdon and Constantine 1975, pp. 85, 105–108.) Coupling and cohesion were identified as the two most important criteria for judging the quality of modular designs (effective modularity) in "Structured Design," an important 1974 paper by Wayne Stevens, Glenford Myers, and Larry Constantine (reprinted as Stevens, Myers, and Constantine 1999). The underlying theory

Box 2.2

(continued)

was set out in detail in Yourdon and Constantine 1975. Since then, low coupling and high cohesion have become widely accepted as characteristics of good or effective modular design. (See Pressman 1997, pp. 357–361; Sommerville 1996, p. 218; van Vliet 2000, pp. 299–303. For empirical studies, see Troy and Zweben 1981; Selby and Basili 1991.) The usefulness of the concepts is not restricted to particular programming languages such as modular programming languages or to particular design paradigms such as functional design. For example, coupling and cohesion have been shown to be useful indicators of the structural quality of object-oriented code and design as well. See Sommerville 1996; Briand et al. 2000; Briand, Devanbu, and Melo 1997; Briand, Lounis, and Wüst 1999.

b. See Stevens, Myers, and Constantine 1999, pp. 232–237, 254; Pressman 1997, pp. 359–361; van Vliet 2000, pp. 301–303.

c. Of course, an architect has considerable freedom as to the exact design of a particular interface for a given degree of coupling between two modules. Although the architect could choose to make the interface highly specific to the interaction in question or specify a very general interface, the level of complexity forced upon the architect depends on the strength of connections between the two components.

d. Stevens, Myers, and Constantine 1999, p. 233; Pressman 1997, pp. 358–359.

e. The high rate of errors in highly coupled systems has been confirmed empirically for conventional systems as well as for object-oriented ones. For example, a study based on a conventionally designed 148,000-source-line system from a production environment found that the routines with the highest ratios of coupling to strength had 7 times as many errors per 1,000 source statements (excluding comments) as the routines with the lowest ratios of coupling to strength had. The study also found that errors in routines with the highest coupling/strength ratios were 21.7 times as costly to fix. (See Selby and Basili 1991.) For an early empirical study of the quality of structured designs supporting this view, see Troy and Zweben 1981. In recent years, researchers have empirically measured the relationship among coupling, cohesion, and error-proneness in object-oriented systems. For example, Briand, Devanbu, and Melo (1997) found that some coupling measures can predict fault-proneness. In a comprehensive empirical evaluation of all object-oriented design measures found in the literature, Briand et al. (2000) found that many measures of coupling and inheritance are strongly related to the probability of fault detection in a class.

Alternative interface specifications may constrain experimentation to different degrees. For example, until the mid 1990s the standards defining the interfaces between peripherals and the rest of the system were specific to certain types of devices: the printer, the monitor, and so forth. This specificity constrained what kind of innovation could occur with respect to a particular interface. Only a device whose communication with the rest of the system could be designed to fit through the interface could connect through it. New or improved devices had to interact with the rest of the system in ways similar to the way the device for which the standard was originally developed interacted with the rest of the system; otherwise a new type of interface had to be developed first. For example, it was possible to connect ink-jet printers, laser printers, and scanners to the printer interface, but it was not possible to connect a completely new type of device. By contrast, the Universal Serial Bus (USB) standard that lets peripherals connect to the rest of the system was explicitly designed in a more general way to ensure that a wide range of devices can be attached to a personal computer (PC). As a result of this design, cameras, MP3 players, mice, keyboards, joysticks, printers, and portable hard disks all can connect to a PC using the same USB interface. Similarly, architectures whose components are loosely coupled leave more freedom to the designers of individual modules than architectures consisting of tightly coupled modules (box 2.3).¹⁵

A modular architecture may negatively affect “global performance” (i.e., the overall performance of the system).¹⁶ In a modular architecture that fosters flexibility, the parts of the system that are likely to change together are grouped within one module. This makes it more likely that later changes to the system can be implemented by changing the internals of the module without requiring any changes to the rest of the system. Though this decomposition is optimal with respect to the flexibility of the system, it may not be the most efficient with respect to performance. Among other things, the performance of a software-intensive system depends on the amount of communication between modules.¹⁷ A decomposition that isolates the parts of the system that are likely to change together in order to reduce the costs of changing them in the future may separate parts that cooperate to provide a particular functionality, thereby increasing inter-module communication and decreasing performance.¹⁸

A modular approach to design thus assumes that the gains from lower complexity, independent component design, higher modifiability, and higher correctness are more important than the negative effect of the

Box 2.3

Coupling and Flexibility

An architecture whose components are loosely coupled leaves more freedom to the designers of individual modules than an architecture that consists of tightly coupled components. In a strongly coupled architecture, the interfaces needed to enable independent component design will be very complex and may be highly specific to the operation of the affected modules.^a From the point of view of component design, this is not very desirable. The more detailed the interface, the higher the risk that detailed design decisions will be made during architectural design, when a lot of detailed information may not yet be available. In the worst case, the architects may choose highly specific but inefficient (from a performance perspective) interface specifications that may significantly impede component designers' ability to create an efficient component design. In any case, complex interfaces considerably constrain the flexibility of subsequent component design.

In contrast, a system whose components exhibit low coupling and high cohesion strikes a more productive balance between complexity-reducing certainty and creative uncertainty^b: few design choices have to be made in the interfaces to enable independent component design; all other choices will be left to the discretion of component designers, leaving more room for creative solutions.

a. For an example, see Sullivan et al. 2001, section 5.1.

b. For an example, see *ibid.*, section 5.3.

constraints imposed by the system's visible information and the potential negative effect on global performance.

Integrated Design

A modular approach to design can be contrasted with an integrated approach, such as has traditionally been used in the designing of physical products.¹⁹

In an integrated approach to design, all design information is common knowledge for the designers. The system is decomposed into components, but interdependencies among components are not resolved during architectural design. Instead, interface specifications are produced during detailed design and can be changed until the detailed designing ends.

In software systems, interdependencies among components usually arise because different components operate on the same data or because

they must interact in order to provide a particular functionality.²⁰ In physical systems, interdependencies may occur if different components share physical space. If parts of the design are interdependent, the corresponding design decisions must be mutually consistent if the system is to work properly.²¹ Thus, design decisions that affect interdependent parameters must be coordinated. Independent component design is not possible.

In an integrated approach to design, unresolved interdependencies that cross the boundaries of components may create chains of dependencies between different parts of the system such that each design choice has to be consistent with all the other decisions in the chain. If a design parameter is part of such a chain, changing this parameter may require widespread adaptations in the rest of the system. In addition, a solution that is locally optimal with respect to one component may not be perfect with respect to design parameters in other components. Finding a solution may require difficult trade-offs across several components, which may add considerably to the complexity of the design process.²² Since design choices in regard to one component are closely linked to decisions in regard to other components, the resulting components are closely coupled.²³ Architectures with closely coupled components are more error-prone than architectures whose components are loosely coupled (box 2.2).

Although it increases the complexity involved in the design process, resolving interdependencies among components in the detailed design stage may enable designers of different components to optimize their designs with respect to each other. Whereas designers of a modular system do not know how other modules implement their functionalities, designers of integrated systems can fine-tune their choices to the exact circumstances in the other component. In this respect, an integrated approach to design tries to simultaneously optimize *all* of a system's parameters. The resulting design may be more efficient or may exhibit better global performance than a modular system that—in order to reduce complexity and increase flexibility—hides information that may have been useful in optimizing its performance. For example, the Apple iMac is based on an integrated approach to physical design. Since all of its design information is common knowledge, designers can tweak components in order to physically interleave and arrange different components in ways that use minimal space.²⁴ This allows them to combine all the usual components of a personal computer system—central processing unit, monitor, hard drive, screen, camera, speakers, microphone, wireless interfaces—in one sleek case that looks more like a conventional monitor than like an Intel-based personal computer.

In sum, an integrated approach to design trades off potentially higher global performance against increased complexity, the need to coordinate the detailed design of components, and the potentially lower correctness of the system.

Layering

Modularity in design does not impose any restrictions on the interactions between modules; a module can use or be used by any other module (figure 2.1).

Layering is a special form of modularity that—in addition to the usual constraints imposed by a modular approach to design—restricts the allowable interactions between modules.²⁵ In a layered system, modules are organized in layers that constrain dependencies between modules. A module assigned to a particular layer can use any of the other modules in the same layer or in a lower layer; however, it cannot use a module belonging to a higher layer. Layering thus organizes modules into a partially ordered hierarchy.²⁶

Like any modular architecture, a layered architecture completely specifies all the information that components need to interact during the architectural design and makes that information visible to all designers. A

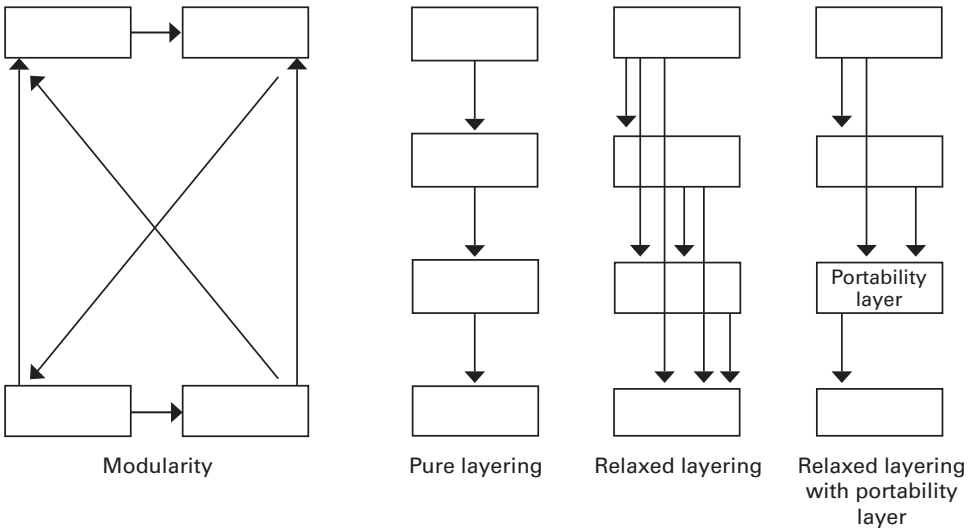


Figure 2.1
Variants of the layering principle.

layer's visible information defines the services it offers to higher layers and the interfaces through which higher layers invoke those services.²⁷ The internal implementation of a layer—the way it realizes the services offered to higher layers—is hidden from everyone except that layer's designers. This information is not specified during architectural design and is free to evolve until detailed designing ends.

Different variants of the layering principle constrain dependencies between layers to different degrees.

In the “pure” version of layering, a layer is allowed to use only the layer immediately below it (figure 2.1).²⁸ As a result, layers interact only with their immediate neighbors: each layer provides services to the layer above and uses the services of the layer below. Since each higher layer adds unique capabilities, a layered system provides increasingly elaborated or specialized services at higher levels. Since each layer is allowed to use the services of the layer immediately below but not those of any other layer, the lower layer hides all the layers below it. Thus, in a system designed according to the pure version, any layer can be seen only by the next higher layer.

In the “relaxed” version of the layering principle, a layer is permitted to use any layer that lies below it (figure 2.1). This is often called *layer bridging*.²⁹

As we will see in chapter 3, the architecture of the Internet is based on a variant of the layering principle called *relaxed layering with a portability layer* (figure 2.1).³⁰ In this version, one of the lower layers is chosen as the portability layer. The layers above the portability layer are allowed to use the services of all layers between them and the portability layer. They are not allowed to use any layers below the portability layer.

For example, the applications, operating system, and hardware of a personal computer are parts of a layered system.³¹ The system's hardware—central processing unit, memory, hard disks, monitors, printers, and so on—all resides at the lowest level. Programming hardware is a complicated task that requires detailed knowledge of the inner workings of a device. Therefore, the next higher layer, the operating system, provides *application programming interfaces* (APIs) to applications at the system's highest layer. APIs let the application developers use the system's hardware without having to worry about how it operates or is controlled. In a sense, they know how the hardware *behaves* but not how it *works*. For example, they let application designers create a file, open or read the file, and save the file somewhere without having to worry about how the file

system is organized and managed or how to write the data onto a disk. In this context, the operating system constitutes a portability layer.³² Because applications are not allowed to invoke the services of the computer's physical hardware directly, they must access them through the application programming interfaces provided by the operating system. The operating system operates as a "floor": layers above it cannot access layers below it.

By basing design on increasing levels of abstraction, layering greatly reduces complexity.³³ Owing to the partitioning of fixed, visible information and malleable, hidden information, layers can be designed independently as long as they conform to the architecture's visible information. Since each layer only uses the services of layers below, a layered system can be implemented and tested incrementally. This facilitates implementation and improves the correctness of the final system.

Layering increases the modifiability of the system by shielding lower layers from changes in higher layers, whether or not the change involves visible information. The hardware of the hard disk is not allowed to rely on the operating system or on the applications and thus is not affected by changes to these layers. In addition, as long as the interface to the lower layer stays the same, higher layers are protected from changes in a lower layer. Application designers are not allowed to access specific hardware directly; they must use the operating system's APIs. As a result, changing how the hard disk and the operating system interact does not affect the applications, as long as the API stays the same. Finally, like any modular architecture, a layered architecture shields the rest of the system from changes to the internals of a layer.

Layering may affect performance negatively. First, only a layer's interface is visible to higher layers, while the layer's implementation is hidden. Thus, the interface provides an abstraction of the layer. While a higher layer can implement its services on the basis of knowledge about the interface, sometimes information about details of the implementation may have enabled a more efficient implementation of the higher layer. As a result, the performance may be worse than in a more closely coupled architecture where more information is available.³⁴ This is a common problem in networking contexts (box 2.4). Second, in a layered architecture, functionalities that operate together may be distributed over different architecture layers. Since communication across components and layers is usually more costly than communication within a component, performance may decrease.³⁵ This may be ameliorated through prudent implementation.³⁶

Box 2.4

How Layering Affects Performance in Communication Networks

By hiding lower-layer information from higher layers, layering may reduce performance. For example, different physical network technologies have different maximum transmission units (MTUs). The maximum transmission unit for a particular physical network technology is the largest size of an Internet Protocol datagram that can be transferred across the network without being broken up into smaller pieces.^a In the architecture of the Internet, the Internet layer is designed to shield the transport layer from such details of the underlying physical network technologies in order to provide an abstract, technology-independent interface to the services provided by the underlying physical networks. As a result, the designers and implementers of transport protocols only have to pass data packets to the Internet Protocol; they do not have to worry about how the Internet Protocol manages the task of sending the data packets across the different physical networks that make up the Internet and delivering them to the transport-layer protocol at the destination host.

This reduction of complexity, though, may harm performance. If the data packets that the transport protocol passes to the Internet Protocol are too large to be transported across the physical networks on the path from sender to receiver without being broken up into smaller pieces, they must be fragmented and reassembled by the Internet Protocol.^b This is a complex and time-consuming process. If the transport-layer protocol knew more about the smallest MTU of all the physical networks that a datagram must traverse in the path from sender to receiver, it could adapt the size of data packets passed to the Internet Protocol accordingly. In this case, the datagrams could travel from sender to receiver without time-consuming fragmentation and reassembly.^c To avoid performance problems of this kind, protocol implementers often relax the strict layering that characterizes the design in the implementation of the protocols and allow information from lower layers such as network MTU to propagate upward.^d

a. See Peterson and Davie 2007, pp. 239–240.

b. Whereas datagrams may be fragmented at the sending end host and at every router along the way, the Internet Protocol reassembles fragments only at the receiving end host. See Peterson and Davie 2003, p. 245.

c. For an overview of fragmentation and reassembly of IP datagrams by the Internet Protocol, see Peterson and Davie 2003, pp. 242–249.

d. See Comer 2000, p. 192. On the use of upcalls in layered systems generally, see D. D. Clark 1985.

Modularity and Layering in Network Architectures

The design principles described so far can be used in any system. To provide the basis for a discussion of the Internet's architecture, this section introduces the basic concepts that are necessary to understand the issues associated with architectural design in the context of communication networks; it also outlines the constraints imposed by the layering principle in the context of network design.³⁷

The goal of a communications network is to make it possible for applications to interact through a network. This is a very complex task. In order to reduce the complexity of the design problem, the functionality required for communicating over a network is commonly decomposed into components and subcomponents by means of modularity and a version of the layering principle that accounts for the distributed nature of communication networks.

At the highest level of a network architecture, designers distinguish two classes of components³⁸: the computers or devices that are "on" or "attached to" a network and those that are "in" a network.

Computers "on" the network support users and run application programs; they use the services of the network to communicate with one another. The home computers many people use to surf the Internet, personal digital assistants that send and receive e-mail using the Internet, the Web servers that carry the content provided by Yahoo or the *New York Times*, and the servers through which users access their Gmail accounts are all examples of computers "on" the network.

Computers "in" the network form or implement the network. They establish connectivity among the computers attached to the network.³⁹ They include the cable modem termination system operated by a cable provider (to which a cable modem is connected to give the user access to the Internet) and the routers that network providers use to forward Internet data from one physical network to another. Viewed from computers in the network, computers on the network are the sources and destinations of data. Data flows travel through computers in the network, but they originate or terminate in computers attached to the network.

Because the communication flows through the network originate or terminate in computers attached to the network, these are also called *end-points*, *end systems*, or *end hosts*. Computers in the network form its *core*.⁴⁰ This terminology denotes a purely functional distinction between users and providers of communication services. It does not imply any topological or administrative relationship.⁴¹ Topologically, an end host may be

co-located with routers belonging to the network’s core. Administratively, who has ownership or control of the computer in question isn’t relevant. A server owned and operated by a network provider is still an end host.

This distinction between devices that use the network and devices that form the network exists in other networks too. End devices such as telephones, fax machines, and modems use the public switched telephone network; the various switches involved in establishing connections from one caller to the other form the core of the network. Electrical devices such as computers, washing machines, and stereo systems use the electricity network; the power transformers used to increase or reduce voltage on the way from the power plant to the customer are devices in the core of the network that work together to supply power to the end devices using the network.

Each computer, whether it belongs to the edge or the core of a computer network, is further subdivided into layers, which are thought of as being arranged in a vertical structure (figure 2.2). Each layer has one or more architectural components called *protocols*. Each protocol provides a well-defined set of services to the layer above, using the services of the layer below.⁴² To implement the services provided by the protocol, different instances of the same protocol located on different computers (“protocol peers”)⁴³ cooperate by exchanging messages with one another. Thus, in the networking context, layering operates both horizontally (through the exchange of messages between protocol peers located on different computers) and vertically (through the use of lower-layer protocols) (figure 2.3).

As in any layered approach to design, a protocol that is assigned to a layer can use any of the other protocols in the same layer or in a lower

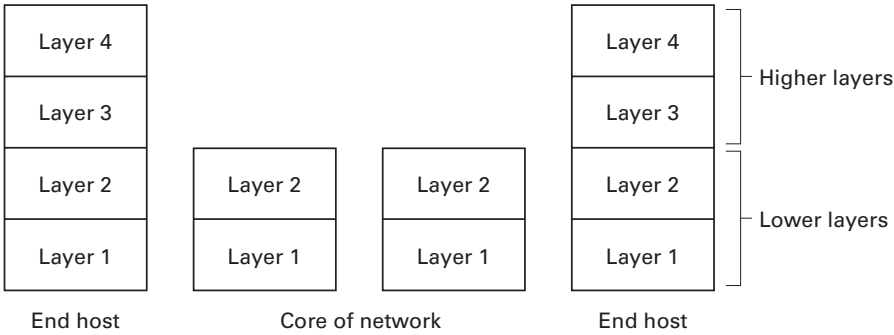


Figure 2.2
Layers, end hosts, and core of the network.

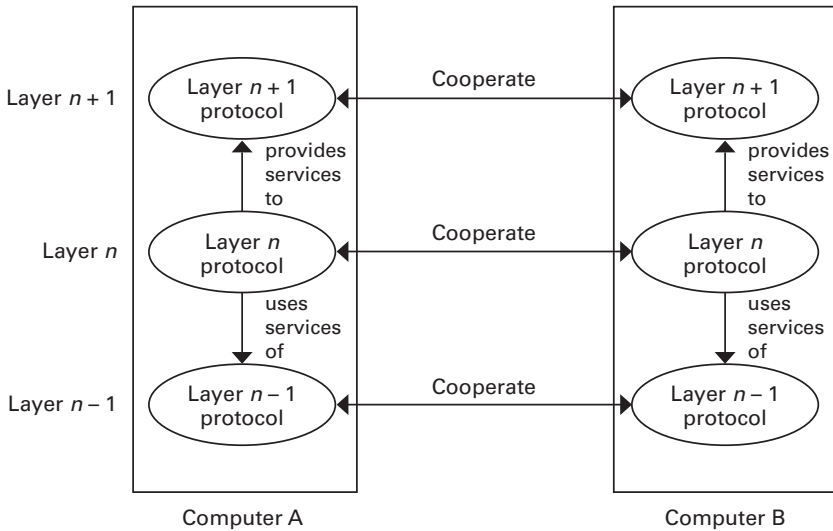
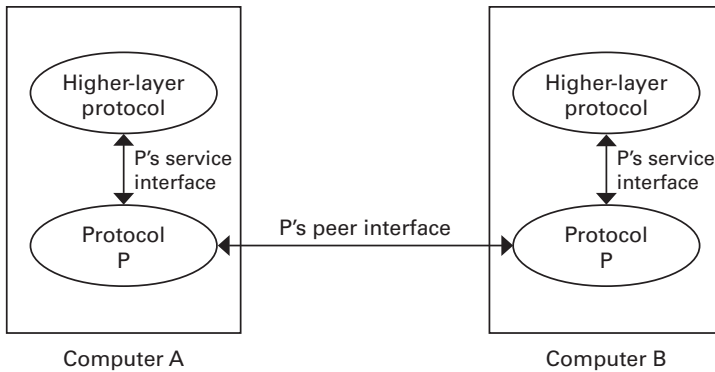


Figure 2.3
Layered protocols.

one, but it cannot use a protocol in a higher layer. In addition, the interaction between protocols is restricted by two additional constraints on the architecture that are specific to layering in the networking context. First, a protocol at a specific layer at a receiver must receive exactly the same object as sent by its protocol peer at the source. Second, a lower-layer protocol is not allowed to make any assumptions about the message passed to it by a higher-layer protocol for delivery to its higher-layer protocol peer. As we will see, these additional constraints maintain the beneficial characteristics of layered architectures in spite of the additional complexity introduced by the need for horizontal communication between protocol peers.

As in any modular or layered system, information necessary to let the components of the system interact is specified during the design of the architecture. Owing to the vertical and horizontal dimensions of layered networking, the visible information of a protocol consists of two different interfaces: a vertical service interface and a horizontal peer interface (figure 2.4). The vertical service interface defines the services that this protocol provides to higher layers. The horizontal peer interface specifies the form and the meaning of messages exchanged between peers and the actions to be taken by the communicating protocol peers upon transmission or receipt of such messages. In other words, the peer interface defines how

**Figure 2.4**

The interfaces of a protocol.

the protocol peers communicate and cooperate to realize the services that the protocol provides to higher layers.^{44,45} In contrast, how a protocol achieves the functionality implied by this visible information is hidden from all except those implementing a protocol.

To see how vertical and horizontal cooperation between protocols work, consider a slightly modified analogy from the real world. Imagine a researcher, working in the research lab of a big company, who wants to submit travel-reimbursement forms to the company's travel-reimbursement department, which is located in the company's headquarters in another city. At the highest layer, the researcher fills in the form for the reimbursement request, attaches the required documents, and hands them over to the mailroom—the next lower layer—with the name of the receiving administrator within the reimbursement department. Someone in the mailroom puts the form and the supporting documents in an interoffice envelope, adds the internal mail code for the administrator, and hands it over to the next lower layer—the postal service—with postage and the postal address of the receiving location. The postal service puts the interoffice envelope in a postal envelope, adds the address, transports the postal envelope to the company's headquarters, removes the postal envelope, and delivers the interoffice envelope to the mailroom. The mailroom uses the code on the interoffice envelope to identify the recipient, removes the interoffice envelope, and delivers the reimbursement-request form and the documents to the administrator in the reimbursement department. The administrator then processes the reimbursement request according to the company's regulations (figure 2.5).

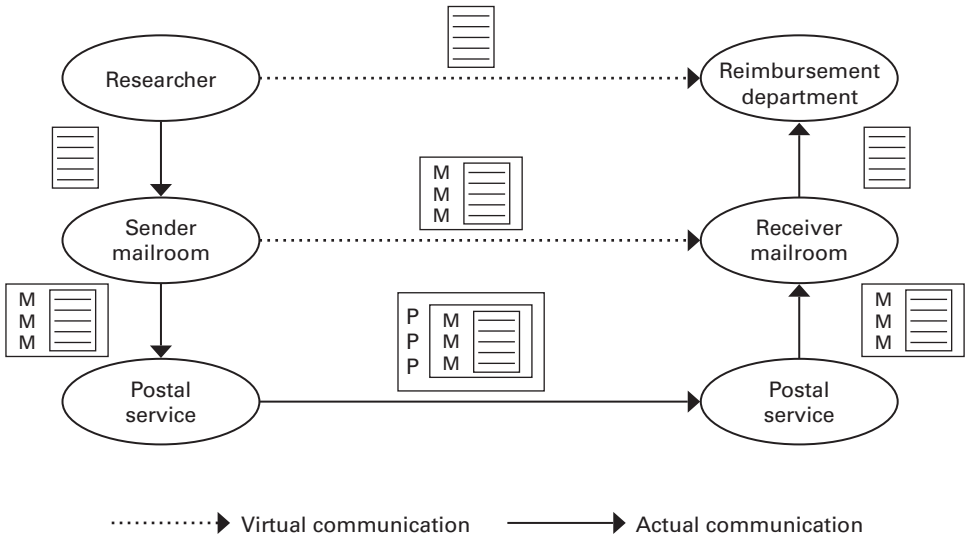


Figure 2.5

Virtual versus actual communication.

In this example, each layer provides a certain service to the next higher layer. The researcher-reimbursement layer provides processing of travel reimbursements. The mailroom layer provides person-to-person delivery within the company. The postal layer provides mailroom-to-mailroom delivery between locations. The way in which a higher layer takes advantage of the services provided by a lower layer and the information that a higher layer must provide to enable the lower layer to fulfill its task are specified by the lower layer's service interface: to deliver a message from one mailroom to the other, for example, the postal service requires the message that the mailroom wants to have delivered to its peer, the appropriate postage, and the address of the receiver.

To realize a layer's service, an entity at that layer cooperates with its peer at the receiving end by exchanging messages.⁴⁶ The format of the messages to be exchanged and the actions to be taken by the receiving peer are specified by the protocol's peer interface. At the highest layer, the message consists of the reimbursement-request form and the supporting documents. The company's rules specifying the format of the form, what documentation is required, and what action the reimbursement department will take upon receipt of the message constitute that protocol's peer interface. At the mailroom layer, the message consists of the interoffice envelope containing the higher layer's message⁴⁷ plus some control infor-

mation⁴⁸ that tells the receiving mailroom what to do with the envelope, such as the internal postal code that identifies the receiver at the higher layer or additional instructions such as a request for rush service. The internal rules that require the use of interoffice envelopes, define the internal mail code, and specify the format and meaning of additional instructions constitute that layer's peer interface. And so on.

You may have noted that each layer receives exactly the same message that was sent to it by its peer (figure 2.5): the reimbursement department receives the reimbursement-request form and the documentation, the mailroom receives the interoffice envelope. If a lower layer adds information (e.g., to communicate with its own peer), that layer's peer at the receiving end removes that information and only delivers the higher layer's message to the higher-layer protocol at the receiver. This is not an accident; it is the result of the first of the two additional constraints imposed by the version of the layering principle used in network design: that layer n at the destination must receive exactly the same object sent by layer n at the source.⁴⁹ This rule requires that a lower-layer protocol may not permanently modify the object passed to it by a higher-layer protocol for delivery to its higher-layer protocol peer.⁵⁰ This constraint lets protocol designers think about the horizontal features of protocol design separately from the vertical aspects: if a protocol at a specific layer can be sure that it will receive exactly the message sent to it by its peer at the sending location, how the message gets from one to the other does not make any difference for the interaction between the protocol peers. Although communication between protocol peers at higher layers happens indirectly⁵¹ through the use of lower layers, the protocol's peer interface can be designed as if the two peers communicated directly.⁵² Put differently, the lower layers are transparent to the horizontal interactions among communicating protocol components. This greatly reduces the complexity involved in protocol design. The designers of a peer interface can ignore the difficulties associated with getting a message from one protocol peer to another and can instead focus on how to structure the exchange of messages between the peers to realize the additional functionality provided by that protocol. In our example, people thinking about what information is needed to process a reimbursement and under which condition the reimbursement request should be granted do not need to think about how to get the form from the employee requesting the reimbursement to the reimbursement department. Given that the lower layer will deliver exactly the same message that the employee sends, it does not make any difference for the peer interface at the reimbursement layer whether the

reimbursement request is delivered in person, by fax, by interoffice mail, or by some other means.

The second constraint specific to layering in the networking context is that a lower-layer protocol cannot make any assumptions about the object (protocol data unit) passed to it by a higher-layer protocol for delivery to its higher-layer protocol peer.⁵³ Thus, the lower-layer protocol may neither access nor act on the information contained in higher-layer protocol data units. This constraint preserves the central feature of layering: the independence of lower layers from higher layers. To access, interpret, or act upon information contained in higher-layer protocol data units, a lower layer must know the format and the meaning of higher-layer messages. This introduces a dependence on the higher-layer protocol. If the higher-layer protocol changes, the lower-layer protocol will not be able to function correctly without being changed. In the example above, the employee gave the mailroom a message to be delivered and the name of the recipient of the message; the mailroom used the name to look up the recipient's internal mail code in the company directory. Imagine instead that the job of the mailroom was to look at the message provided by the higher layer and infer the recipient itself, using the knowledge that travel reimbursements are processed by the reimbursement department. Now consider what would happen if the responsibility for processing reimbursements were to be transferred to another department. In the original example, the operation of the mailroom layer (to look up the internal mail code for the name given to it by the higher layer) would not be affected by the change. In the variation just discussed, the change at the reimbursement-processing layer would require a change at the mailroom layer, as the existing procedure (deliver reimbursements to the reimbursement department) would deliver the reimbursement request to the wrong department.

The description of the travel-reimbursement example glossed over an important difference between how higher layers and lower layers operate. At the reimbursement layer and at the mailroom layer, a protocol entity at the sending location and a protocol entity at the receiving location exchange messages to realize the service provided by their protocol. In contrast, at the postal layer, the postal-service entities in the sending and receiving locations and additional intermediate entities on the path from origin to destination contribute to the realization of the service. In the postal service, letters are not transported from sender to receiver directly, but through a series of intermediate locations, such as the local mail processing and distribution center, hub and spoke processing centers, and the mail processing and distribution center close to the company's head-

quarters. At each postal-service location, an entity belonging to the postal layer checks the address on the envelope to determine the next destination on the path and uses the services of yet another layer (let us call it the *transporting layer*) to deliver the envelope to that destination, where the same process is repeated until the envelope arrives at its final destination.

As we will see below, a similar difference in the operation of different layers can often be found in layered networks. Consider a communications path that consists of a series of nodes. Sometimes a higher layer may operate directly between the first and the last node on the path (i.e., end-to-end between the endpoints of the path), while a lower layer may operate hop-by-hop between adjacent nodes along the path (box 2.5, figure 2.8). As we will see, differences in operation between layers that operate hop-by-hop rather than end-to-end have important implications for the inability of lower layers to correctly implement a certain class of functions. This inability, in turn, is an important rationale behind the narrow version of the end-to-end arguments.

The End-to-End Arguments

The end-to-end arguments guide the placement of functionality in a multi-layer system. Although the layering principle advises that the functionality of a system should be organized in layers, it does not provide any rules for assigning functions to layers. Thus, when designing a layered system, architects still have to decide how to divide its functionality among layers. In particular, when building a general-purpose system that is intended to support a variety of applications, architects often create a layered structure. The applications reside on top of the general-purpose system, which forms the lower levels of the hierarchy. With this general structure, there are many application requirements that could be implemented at a higher or a lower level, or at several layers in the hierarchy. In particular, functionality that is needed by specific applications could be implemented as part of the application (a higher layer) or could be incorporated in the general-purpose system (which forms the lower layers of the system). With respect to this choice, the end-to-end arguments provide a rationale for moving application-specific functionality upward in a layered system.⁵⁴

In the context of communication networks, the end-to-end arguments have vertical and horizontal dimensions. In layered communication networks, the allocation of functionality among layers affects how functionality is distributed among end hosts and computers in the network's core.⁵⁵ Higher layers only have to be implemented on end hosts; they do not have

to be implemented on computers in the core of the network. As a result, the decision to place functionality in a higher layer places this functionality at the end host. While lower layers are implemented on all computers, they are the only layers that have to be implemented on computers in the network's core.⁵⁶ As a result, their design determines the functionality present in the core of the network.

The end-to-end arguments, which had been implicitly used in system design for years, were first identified, named, and described by Jerome Saltzer, David Reed, and David Clark in 1981.⁵⁷ In the networking community, the role of the end-to-end arguments as a fundamental principle governing the design of computer networks is generally acknowledged.⁵⁸ The end-to-end arguments are also believed to be among the few architectural principles underlying the design of the Internet.⁵⁹ However, there is a great deal of uncertainty about how to apply the end-to-end arguments in specific cases. Often, proponents and opponents of a technical solution refer to the end-to-end principle to justify their view. In view of this widespread disagreement, it is necessary to clarify exactly what is meant by "the end-to-end arguments" before we can test whether a specific technical solution complies with this design principle.

It seems natural to look to Saltzer, Reed, and Clark for an authoritative definition of the principle. Since 1981, when their paper was published, they have revisited the end-to-end principle in several papers. In a joint paper, they evaluated active networking in the context of the end-to-end arguments.⁶⁰ In two recent papers, David Clark and Marjory Blumenthal have analyzed the role of the end-to-end arguments in light of the changing requirements of today's Internet.⁶¹ In addition, David Reed and Jerome Saltzer have written short comments referring to the considerations underlying the original paper.⁶²

Studying the papers mentioned in the preceding paragraph yields a surprising result: there are two versions of the end-to-end arguments that represent different rules for architectural design.⁶³ To see this, consider the following two statements of "the end-to-end principle": "A function should only be implemented in a lower layer, if it can be completely and correctly implemented at that layer. Sometimes an incomplete implementation of the function at the lower layer may be useful as a performance enhancement"⁶⁴ (first version) and "A function or service should be carried out within a network layer *only if it is needed by all clients of that layer*, and it can be completely implemented in that layer"⁶⁵ (second version). The first version paraphrases the end-to-end principle as presented in the original paper (the complete original definition is given below). The second version

is directly taken from the paper on active networking and end-to-end arguments. Clearly, the second version establishes much more restrictive requirements for the placement of a function in a lower layer. This insight may be surprising—the authors never explicitly drew attention to the change in definition.⁶⁶ In the literature, most texts that refer to “the end-to-end arguments” simply quote either the first or the second version.⁶⁷ Whereas technical discussions often focus on the first version,⁶⁸ policy texts and descriptions of the Internet’s architecture usually use the second version.⁶⁹ Yet nearly all texts refer to the example of reliable file transfer that was used to explain the reasoning behind the end-to-end arguments in the original paper.⁷⁰

Have the end-to-end arguments simply evolved over time?⁷¹ Should we distinguish between the two versions? As the subsequent analysis will show, it is better to distinguish between the two. Both provide a justification for moving functionality upward in a layered system, but there are real differences between the two versions in scope, content, and validity. For example, while the first version (the “narrow” version) applies only to a certain group of functions within a system, the second version (the “broad version”) applies to the complete functionality of a system. The terminology (“narrow” vs. “broad”) used throughout this book reflects this difference in scope.⁷² As we will see, the rules imposed by the broad version are more restrictive than those imposed by the narrow version. Whereas the narrow version is based on an argument about correctness and claims absolute validity, the broad version is the result of a trade-off based on a specific prioritization of values. Designers must follow the narrow version if they want to design a correct system, but they can choose whether to apply the broad version. They should apply the broad version only if the trade-off among values underlying this design principle reflects the combination and the prioritization of values they want to achieve.⁷³

Since both versions constrain designers to a different degree, their application may result in different architectures. Some architectures may be compliant with one version but not with the other. Failing to distinguish between the two may lead to incorrect systems, may create confusion, and may lead to unnecessary debates about the correct application of the end-to-end arguments, both in policy debates and in the networking community.

The following analysis explores these questions in more detail. Starting with the narrow version that was presented in the original paper by Saltzer, Reed, and Clark, it sets out the two versions of the end-to-end arguments and outlines the reasoning behind them. It then compares the

two versions and highlights the reasons in favor of distinguishing between the two.

The Narrow Version

The Design Principle The narrow version of the end-to-end arguments was first articulated in the original end-to-end paper by Saltzer, Reed, and Clark, where it was applied to the division of functionality between an application and a communication system⁷⁴:

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible [i.e., it is not possible to provide a complete and correct implementation of the function within the communication system only]. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)⁷⁵

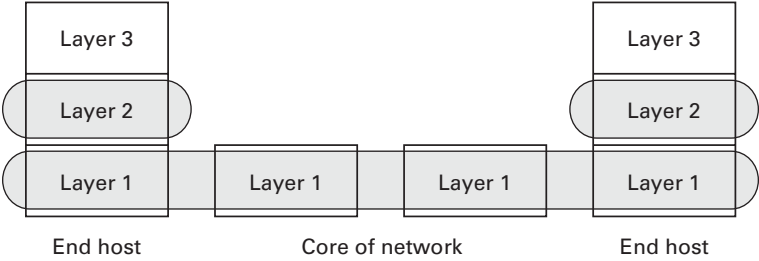
Later the principle was applied to the division of responsibility between end systems and the core of the network and generalized to the placement of functionality in higher or lower layers:

A function should not be implemented in a lower layer, if it cannot be completely and correctly implemented at that layer. Sometimes an incomplete implementation of the function at the lower layer may be useful as a performance enhancement.⁷⁶

In the context of the narrow version, the choice between layers is not always equivalent to the choice between end systems and the core of the network. It is equivalent if the lower layer is implemented in the network and the higher layer operates end-to-end between the end hosts. In this case, placing a function in the higher layer also puts the function at the end host, whereas placing the function in the lower layer puts the function in the network (figure 2.6, case 1). The choice between layers is not equivalent to the choice between end systems and the core of the network if the layers under consideration terminate in the same pair of end systems. In this case, either choice places the function at the end host, but in different layers (figure 2.6, case 2). Thus, the importance of the narrow version goes beyond the division of functionality between end systems and the core of the network.

Analysis The narrow version of the end-to-end arguments highlights the existence of functions that can be completely and correctly implemented only at the endpoints, i.e., end-to-end between the original source and ultimate destination of data; these are the functions to which the argument

Case 1: Should the function be placed in layer 1, or in layer 2?



Case 2: Should the function be placed in layer 2, or in layer 3?

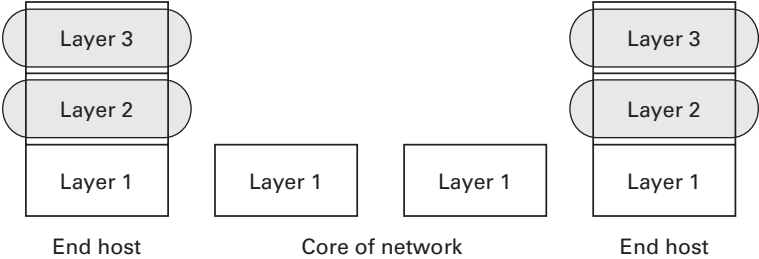


Figure 2.6

The choice between layers is not always equivalent to the choice between end hosts and the core of the network.

applies. Throughout this book, these functions will be called *end-to-end functions*. As we will see, implementing these functions completely and correctly at a lower layer (in the network) is not possible.⁷⁷ If designers want to build a correct system—a system that behaves as it is supposed to—these functions must be implemented at a layer where a complete and correct implementation is possible. In general, this will be a higher layer where protocols operate end-to-end between end systems, i.e., between the original source and the ultimate destination of data (at the end hosts). In some cases, it may even be the highest layer—the application itself.

Also implementing a necessarily incomplete version of these functions at a lower layer (in the network) is redundant and usually constitutes an inefficient use of system resources. In some cases, however, an incomplete implementation at the lower layer may be justified to enhance system performance.

The narrow version of the end-to-end arguments, then, provides two design rules for end-to-end functions: first, end-to-end functions must be

implemented at a layer where they can be completely and correctly implemented. Second, whether the function should also be implemented at a lower layer must be decided case by case.

But what functions are end-to-end functions to which the narrow version applies, and why can they not be completely and correctly implemented at a lower layer within the network?⁷⁸

In general, a function (e.g., error control to ensure data integrity) cannot be completely and correctly implemented at a lower layer if events falling into the function's responsibility (e.g., corruption of data) can occur outside the scope of control of the lower-layer protocol. In this case, the lower-layer protocol cannot cover all events for which the function is responsible; it is not able to perform the function completely and correctly.

This problem may arise in two cases: in the first case, the lower layer operates hop-by-hop, while the higher layer operates end-to-end, and events falling into the responsibility of the function may occur within the nodes along the path (figure 2.7, threat 1).⁷⁹ In this case, the lower layer will not be able to implement the function completely and correctly. In the networking context, the canonical examples of this type of problem are error control to ensure data integrity and encryption to ensure confidentiality.

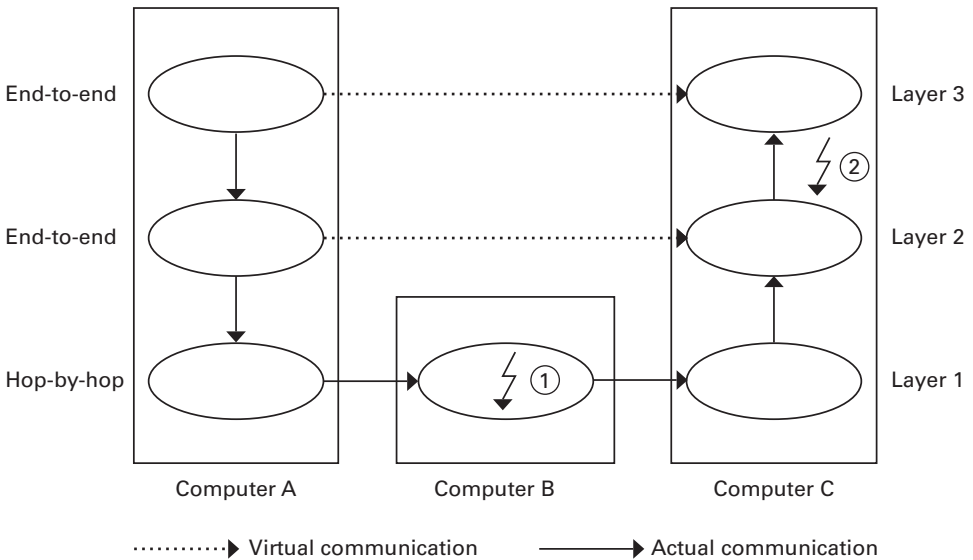


Figure 2.7

Only a higher-layer protocol that operates end-to-end can protect against threat 1. Only a protocol at layer 3 or higher can protect against threat 2.

Consider a communication path that consists of a series of nodes from the origin to the destination. The higher layer operates end-to-end between the origin and the destination, while the lower layer operates hop-by-hop between the nodes along the path (box 2.5).

If error control is performed by a hop-by-hop protocol at each hop along the path, the protocol ensures that the data that leaves the protocol entity

Box 2.5

End-to-End versus Hop-by-Hop Execution of Functionality

Consider a communication path that consists of a series of nodes from the origin to the destination (figure 2.8).^a A protocol that operates end-to-end between the origin and the destination operates “directly” between these two systems, the endpoints of the path. In such a protocol, the protocol peers at the origin (A) and the destination (B) communicate with each other directly to realize the protocol’s functionality.^b Functionality implemented end-to-end is performed once between the endpoints. A protocol that operates hop-by-hop along this path operates between neighboring computers on the path from the origin to the destination. Functionality implemented in this way is performed locally at each step or “hop” along the path without regard to the protocol’s operations at the other hops. At each step, the two protocol peers at neighboring nodes cooperate to execute the function. That is, the protocol entity at the origin (A) does not communicate with its counterpart at the destination to realize the protocol’s functionality, but with its peer at the next computer (B) in the path. In the next step, the protocol entity at B communicates with its peer at the next computer (C) in the path, and so on.

a. Whereas the concepts of “end-to-end” and “hop-by-hop” are often presented in the context of communication between end systems, end-to-end versus hop-by-hop is a relative concept that can apply to other sequences of nodes within a network where a source node communicates with a destination node, from which it is separated through a series of nodes. The choice then is whether to implement a function end-to-end between the source and destination or hop-by-hop between the hops along the path. For example, “with respect to the network nodes, flows between switches or routers can be viewed as end-to-end, while links between link repeaters (or bridges) are viewed as hop-by-hop” (Sterbenz and Touch 2001, p. 41). For a more detailed discussion and additional examples, see *ibid.*, pp. 39–41, 345–346.

b. “Direct communication” is a conceptual abstraction. In reality, protocol peers on different computers do not communicate directly; they use the services of lower layers to send messages to their peers.

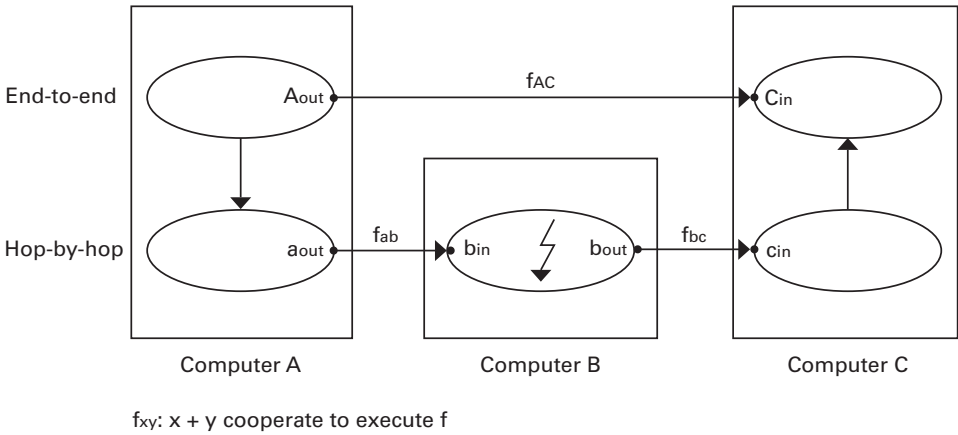


Figure 2.8
End-to-end versus hop-by-hop execution of functionality.

at one node is identical to the data that arrives at this entity's peer at the next node. The protocol does not detect what happens while the data is processed within the node. For example, in figure 2.8, the hop-by-hop protocol would check whether the data that leaves a_{out} is the same as the data that arrives at b_{in} and whether the data that leaves b_{out} is the same as the data that arrives at c_{in} , but it does not test whether the data that arrives at b_{in} is the same as the data that left b_{out} . Thus, if error control is implemented hop-by-hop, data may still get corrupted in the nodes. For example, errors may be introduced within a node while data is transferred from the input to the output buffer, because of errors in the network interface cards or because of memory bit errors.⁸⁰

In contrast, end-to-end error control at the higher layer detects whether the data leaving the higher layer at the first node (at A_{out}) is identical to the data arriving at the higher layer of the last node (at C_{in}). The mechanism identifies all errors that may have been introduced along the path, both between two nodes and within a node. For this reason, error control can be completely and correctly implemented only at a higher layer that operates end-to-end between the data's original source and ultimate destination.⁸¹

Similarly, if data is encrypted by a protocol in hop-by-hop fashion, the protocol encrypts the data when it leaves the protocol entity at the node. Once the data reaches the protocol peer at the next node, it is decrypted so that this layer or higher layers within the node can process it, and it is not encrypted again until it leaves this layer to be sent out. Thus, while

hop-by-hop encryption ensures confidentiality while the data is in transit between two nodes, it does not prevent unauthorized access to data while the data is unencrypted in the nodes along the path. In contrast, if data is encrypted by a higher layer that operates end-to-end between the first and the last node along the path, it is encrypted when it leaves the higher layer at the first node and decrypted when it arrives at the higher layer at the last node. Thus, the data is kept confidential along the entire path.⁸²

There is a second case in which a lower layer cannot correctly and completely implement a function (e.g., error control): the case in which both layers terminate in the same pair of computers, but events that the function should cover (e.g., corruption of data) occur before data enters the lower layer or after data leaves the lower layer (figure 2.7, threat 2).⁸³ In this case, only a function implemented at or above the layer where the event occurs can guarantee the correct execution of the function.

Again, error control and encryption may be afflicted with this problem. Consider two layers that terminate in the same pair of computers. If data may get lost or corrupted after it leaves the lower layer, this layer cannot completely and correctly implement error control. Instead, error control must be performed at or above the layer at which the problem occurs. For example, if the file system introduces errors while storing the data, only an error control performed by the file-transfer application may be able to spot and correct this problem.⁸⁴ With respect to encryption, if the end system itself is not secure, unauthorized access may occur after the data leaves the lower layer. Thus, encryption to ensure confidentiality cannot be completely and correctly implemented at this layer. To ensure confidentiality, the data must be encrypted and decrypted at a higher layer—at the lowest trusted layer within the end system.⁸⁵

Error control and encryption are therefore end-to-end functions, and the narrow version of the end-to-end arguments applies to them. Apart from error control and encryption, the original paper mentions delivery guarantees, suppression of duplicate messages, guaranteeing first-in-first-out message delivery, and transaction management as examples of end-to-end functions.

If designers want to build a correct system (one that behaves as required), these end-to-end functions must be implemented at a higher layer at which they can be completely and correctly implemented. Which layer meets this criterion depends on where events occur that have to be covered by the function.⁸⁶ In general, the appropriate layer will be a higher layer where protocols operate end-to-end between end systems, i.e., between the

original source and ultimate destination of data; in some cases, it may be the highest layer—the application itself.⁸⁷

While an end-to-end function must be implemented at a higher layer to ensure the correct operation of the system, the narrow version of the argument explicitly mentions the possibility of placing an additional implementation of the function at the lower layer.⁸⁸ Because the lower layer cannot implement the function completely and correctly, an instantiation at this layer is necessarily incomplete. Although the lower-layer implementation is redundant, it may improve overall performance.⁸⁹ Thus, the decision to also implement a necessarily incomplete version of the function at the lower layer is the result of a trade-off.⁹⁰ In the words of the original paper, “the amount of effort to put into reliability measures within the data communication system [as an example of a partial low-level implementation] is seen to be an engineering trade-off based on performance, rather than a requirement for correctness.”⁹¹

For example, if some underlying links are very unreliable, performing additional error control hop-by-hop along these links may increase the efficiency with which communication resources are used: as more erroneous data is retransmitted locally over one link instead of end-to-end through the whole network, network load is reduced; so, potentially, is overall delay. However, local error control increases the amount of processing within the network (which may increase delay) and consumes additional bandwidth (because of additional checksums at the lower layer).⁹²

Different technologies may justify a different resolution of this trade-off. Consider a computer that accesses the Internet over a wireless link without error control. Because the only error control operates end-to-end between the sending host and the receiving host, packets will have to be retransmitted across the whole network every time a packet is lost. Since wireless networks are very unreliable, there will be many end-to-end retransmissions. Error control on the wireless link would replace these retransmissions across the whole network with retransmissions across the wireless link. The considerable reduction in network traffic may justify the implementation of error control on this link. In contrast, the technology used by a dial-up modem is much more reliable than wireless links, but has far less bandwidth. Because of the modem’s reliability, local error control would not drastically reduce the number of end-to-end retransmissions resulting from problems on this link. At the same time, the additional bits needed to implement the error control would consume valuable bandwidth on a link with little to spare. Thus, an additional implementation

of error control on the access link may be worth the cost on a wireless access link, but not on the link used by a dial-up modem.⁹³

In addition to these general considerations, the original paper mentions two specific costs of performing an end-to-end function at the lower layer that should be considered in the trade-off. First, executing the function is costly for applications that do not need it.⁹⁴ Second, the application or higher layers may have more information about their needs than lower layers (the network), and may therefore be able to perform the function more efficiently.⁹⁵ These arguments play important parts in the broad version of the argument and will therefore be explained below. They show, however, a bias against partial lower-level implementations of end-to-end functions in the original paper.⁹⁶

The Broad Version

The Design Principle Whereas the narrow version applies only to functions that cannot be completely and correctly implemented at lower layers (in the network), the broad version provides a general rule for dividing functionality among layers in a multi-layer system and for distributing functions between end systems and the core of the network in multi-layer communication networks. The broad version is emphasized in later texts by the authors of the original paper.⁹⁷

According to the broad version, “specific application-level functions usually cannot, and preferably should not, be built into the lower levels of the system—the core of the network.”⁹⁸ Instead, “a function or service should be carried out within a network layer only if it is needed by all clients of that layer, and it can be completely implemented in that layer.”⁹⁹ Lower layers of the system (the network) should provide only general services and functions of broad utility across applications in order to support as many higher-layer applications as possible. Lower-level functions are not optimized to better support specific higher-layer applications. Though this may increase the performance of the particular application, it also constitutes an unnecessary and therefore inefficient feature for applications that do not need this function; it may even rule out the implementation of applications not foreseen at the time of the design.¹⁰⁰

Thus, according to the broad version, lower layers of the system—the network’s core—should provide only general services that can be used by all applications. Application-specific functionality should be concentrated in the higher layers of the system, at the end hosts (figure 2.9). The broad

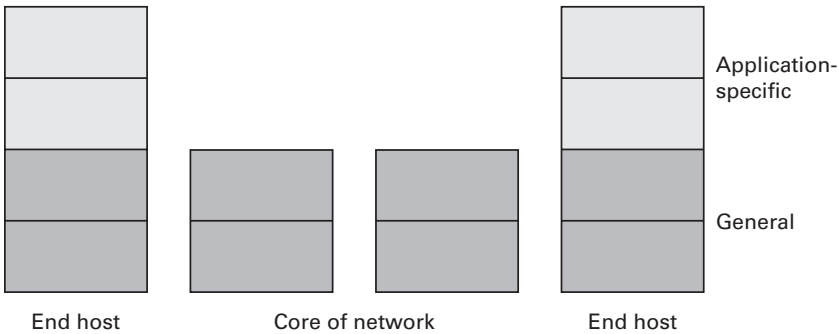


Figure 2.9

Distribution of functionality in a network based on the broad version.

version does not prevent functions from being implemented in the network if they cannot be completely and correctly implemented at the end hosts only.¹⁰¹

Analysis On the one hand, applying the broad version of the end-to-end arguments results in an architecture that can accommodate new applications not foreseen at the time of design. Having the designers of applications (who know the needs of their applications) design application-specific functionality may be more efficient than asking designers of lower layers to anticipate the needs of future applications. Finally, an architecture based on the broad version may be more reliable than an architecture that implements application-specific functionality in the core of the network.

On the other hand, the broad version prevents optimizing the performance of current applications if the optimization involves the implementation of application-specific functionality in the core of the network. This reduced performance is the price for the evolvability of the system. The broad version thus represents a trade-off between long-term evolvability, application autonomy, and reliability, on the one hand, and certain types of performance optimization, on the other. Designers should choose the broad version of the end-to-end arguments if the trade-off among qualities underlying the broad version reflects the relative importance of qualities in the system under consideration. In the words of David Reed: “One of the main conclusions of the end-to-end argument is usually to point out the inflexibility or difficulty of evolution of a design to incorporate requirements not known at the time of system design. Non-end-to-end designs usually fail to meet future needs quickly. That’s the point. No one would make an end-to-end design if they had a fully specified problem and were

searching for the absolute optimum performance where all conditions are known. Yet many of the ‘let’s rewrite end-to-end’ arguments are based on some person’s urgency to deliver the maximum performance possible in the shortest possible time, evolvability-be-damned.”¹⁰²

Evolvability A general-purpose system is designed to support various applications that may have different requirements. Implementing functionality in lower layers of a system that is needed by only some applications, or otherwise optimizing the lower layers for the needs of specific applications, will usually improve the performance of these applications.¹⁰³ Since the lower layers of the system must be used by all applications, applications with diverging needs will have to bear the impact of the optimization too.¹⁰⁴ Functionality that lets some applications work better may hurt other applications whose needs differ.

Problems caused by application-specific functionality in the network are likely to become more severe if the architecture is intended to last a long time. Though it may be possible to perform a constrained optimization that optimizes the network for a known set of uses, the result is not likely to meet the needs of applications that are unknown and unpredictable at that time.¹⁰⁵ In the best case, a new application must pay the costs of a service it does not need. In the worst case, the functionality contained in the lower layers may prevent the new application from being deployed at all.

Consider two examples. The public switched telephone network was originally designed to transmit human speech. Over time, network providers deployed a number of technologies in the network that optimized the network for the needs of telephony. The telephone network transmits sounds within a certain frequency range (300–3,400 Hz).¹⁰⁶ When sound is transmitted over long cables, the strength of the signal declines rapidly. Higher frequencies lose strength faster than lower frequencies, which reduces the quality of transmission. To ameliorate this, network providers deployed devices called *load coils* on cables longer than 18,000 feet. Load coils boost the strength of the higher frequencies to the same level as the lower frequencies. As a side effect, they effectively cut off frequencies above 3.4 kHz. Since telephony does not use the frequencies above 3.4 kHz, network providers did not view this as problematic. Digital Subscriber Line (DSL), the technology used to transmit digital data over the last mile of the public telephone network, uses the frequencies above 25 kHz. Since load coils cut off signals above 3.4 kHz, they have to be removed before DSL can be used over these lines. Thus, load coils optimized the telephone network for the needs of a specific application (voice telephony), but they

are incompatible with another application (high-speed DSL Internet service) that was not foreseen at the time of the optimization.¹⁰⁷

In the 1990s, various aspects of the Internet's architecture were optimized for the needs of a particular group of applications called *client-server applications*, which were the dominant class of applications at the time.¹⁰⁸ These applications included Web surfing, online shopping, and watching video streams. Client-server applications consist of a client program and a server program, both running on end hosts.¹⁰⁹ The client program and the server program interact in a pattern typical of client-server applications: the client requests a service from the server, and the server delivers the service. Whereas the request for service usually does not use a lot of bandwidth, the delivery of the service is often bandwidth-intensive. For example, when you surf the Web, the browser sends a small amount of data to the server to ask for a specific website. The server returns a large amount of data—the webpage—to the browser. Similarly, the client of a streaming video application such as RealPlayer sends a small amount of data asking for a specific video; the server then sends a large amount of data—the streamed video—to the client.

Given that most of the applications that were known at the time followed this pattern, network providers designed the access links (the links connecting the home to the network provider) of their broadband networks to provide asymmetric bandwidth. Technologies such as ADSL (deployed in telephone networks) and HFC (deployed in cable networks) offer significantly more bandwidth from the provider to the home than from the home to the provider.¹¹⁰ This lets providers get more downstream bandwidth out of an access link with a certain capacity than if the upstream and downstream bandwidth are given an equal amount of bandwidth. This solution improves the performance of client-server applications by giving them more bandwidth where they need it and less bandwidth where they do not need it. Today, asymmetric bandwidth creates problems for applications that send and receive an equal amount of data; these applications are called *peer-to-peer applications*. Internet-telephony applications (Vonage, Skype), video conferencing software, and file-sharing applications (Napster, Gnutella, BitTorrent) all have these characteristics.¹¹¹

Whereas functionality implemented in the network usually has to be used by all applications, applications can choose the higher-layer protocols they want to use.¹¹² Thus, following the broad version of the end-to-end arguments and placing application-specific functionality in a higher-layer protocol at the end host restricts the positive as well as the negative effect of application-specific functionality to applications using the protocol.

Since lower layers in a system designed using the broad version provide only very general services and are not optimized for specific higher-level applications, the lower levels can support a wide variety of applications and do not rule out new applications not foreseen at the time of design.

In addition, as we will see in chapter 4, designing a system using the broad version of the end-to-end arguments concentrates functionality that is likely to change in areas of the architecture that are easy to change. Areas of the architecture that are difficult to change contain functionality that is not likely to change over the lifetime of the system.

Application Autonomy The broad version also supports application autonomy. Application autonomy is the idea that an application or higher layers close to it know best what services they need and should therefore be responsible for meeting these needs.¹¹³ This has two aspects. First, it is very difficult, if not impossible, for lower-layer designers to guess in advance what tailored services potential future applications might require. Thus, it is very likely that some higher layers will always end up having to implement application-specific services themselves. It therefore seems more reasonable (and more efficient, as lower-layer designers would put effort into designing functionality that would ultimately have to be designed again at a higher layer anyway) to let higher layers take care of their specific needs instead of shifting this responsibility to lower layers.¹¹⁴ Second, the application or higher layers that are more application-specific than lower layers may have more information about their needs than lower layers, and may therefore be able to perform the function in question more efficiently.¹¹⁵ For example, not all compression algorithms are equally suited to compress certain types of information. Whereas file data such as executable code can only be compressed using lossless compression techniques, the characteristics of video or audio are better suited to the use of lossy compression algorithms.¹¹⁶ The application, or an application-specific higher layer that knows the type of information it wants to transfer, is in a better position to choose an appropriate compression algorithm than a lower layer that does not have this information.

To fulfill their responsibility, higher layers must be able to use the building blocks provided by lower layers according to their needs.¹¹⁷ Application autonomy implies a hierarchical relationship between applications and the network: the applications are in control, and the network has a serving role. Lower layers are responsible for providing very general building blocks, which can then be used by the application designer to realize application-specific needs.

In the original Internet, users control end hosts, while network providers control the core of the network. Under these circumstances, by putting applications on end hosts in control, the principle of application autonomy effectively puts control over the use of the network in the hands of the users.¹¹⁸

Reliability Finally, using the broad version is likely to increase the reliability of applications and the network.¹¹⁹ For applications, implementing application-specific functionality in the network introduces additional points of failure in the network that are not under the control of the designer or the user of the application. In contrast, an application that does not depend on the correct implementation and functioning of application-specific services in the network may be more reliable, since all potential points of failure are under the control of the designer or the user of the application.¹²⁰

For the network, concentrating application-specific functionality and services at the end hosts isolates the network from the consequences of erroneous applications¹²¹: the impact of unexpected malfunctions and failures of application-specific functions is restricted to individual end hosts and data flows and does not affect the network as a whole. In contrast, a failure of application-specific functionality in the network may negatively affect the entire network, not just certain applications. In addition, restricting application-specific functionality to end hosts may reduce the complexity of lower-layer protocols, thereby reducing the complexity of programs that must be implemented on computers in the network's core. Since it is easier to design, implement, and test programs that are less complex, the resulting network is likely to be more reliable.

Lack of Application Awareness and Application Control in the Core of the Network Applying the broad version of the end-to-end arguments results in a network that is not able to distinguish between the different applications running over it, or to control or to positively or negatively affect their execution. This feature is often attributed to the end-to-end arguments directly.¹²² As is apparent in the description of both versions above, however, neither version deals explicitly with this question. Thus, those reading either version of the end-to-end arguments and expecting to find a non-discrimination rule will be rightly disappointed. This does not mean that there is no relationship between the end-to-end arguments and non-discrimination, though. The lack of application awareness and application control in the network's core is a direct consequence of the broad version

of the end-to-end arguments.¹²³ This is because it is not possible to design a network architecture that gives the network the ability to distinguish between the applications running over it or to control their execution without violating the broad version of the end-to-end arguments. In order to identify the different applications using the network and to positively or negatively influence their execution, application-specific filtering and control mechanisms must be implemented in the network's core. The implementation of application-specific functionality in protocols in the core of the network directly violates the broad version of the end-to-end arguments that prescribes that application-specific functionality must be concentrated in and be restricted to higher layers at the end host.¹²⁴ The filtering and control mechanisms in the network's core may also violate the layering principle if the mechanisms operate at the Internet layer or at a lower layer but access or modify the message¹²⁵ (protocol data unit) passed to them by higher-layer protocols.¹²⁶ In that case, the mechanisms violate the second constraint imposed by the layering principle in the context of layered communication networks: a lower-layer protocol may not make any assumptions about the protocol data unit passed to it by a higher-layer protocol for delivery to its higher-layer protocol peer, and consequently may neither access nor act on the information contained in higher-layer protocol data units.¹²⁷

The application-blindness of an end-to-end network prevents the network owner from discriminating against applications running on its network. Thus, the broad version of the end-to-end arguments effectively shields independent application developers from strategic behavior by network owners.¹²⁸ From the perspective of an independent application developer, this is a benefit resulting from the application of the broad version of the end-to-end arguments. From the point of view of a network owner, this feature constitutes a cost of end-to-end design if the network owner has an incentive to discriminate against independent applications. This question is explored more in chapter 6.

Since the application-blindness of an end-to-end network prevents the network owner from discriminating against applications, an end-to-end network is neutral among applications. The "network neutrality" that results from applying the broad version of the end-to-end arguments is at the core of the current policy debates.¹²⁹ Arguing that network neutrality has beneficial effects on innovation, proponents of regulation are asking legislators and regulators worldwide to protect network neutrality by regulating the owners of broadband networks.¹³⁰ Opponents of regulation argue that, because network owners do not have an incentive to

discriminate against independent applications, network neutrality does not have to be enforced through regulation,¹³¹ or, alternatively,¹³² that a network-neutrality regime would reduce network owners' incentives to upgrade their broadband networks in the future.¹³³

Costs of the Broad Version of the End-to-End Arguments The broad version of the end-to-end arguments does not come without costs. First, it may be difficult for inexperienced Internet users to understand and manage complex end hosts. Since all application-specific functionality is concentrated at the end hosts, application software must be installed, configured, upgraded, and maintained by users. For inexperienced users, this may be highly problematic.¹³⁴ The problem can be ameliorated, however, by hiding this complexity from the user—for example, users have the option of buying personal computers with pre-installed, pre-configured software. Similarly, software can be upgraded automatically over the Internet.¹³⁵ Second, for the network owner, an end-to-end design may complicate network management, security, and pricing. The complications arise mainly from the application-blindness of the network. Network management may be easier if you know what applications are running over the network. For example, data on patterns of use may enable network providers to predict or at least observe changes in the behavior of users, and may facilitate capacity planning in a network.¹³⁶ Similarly, a clear overview of the applications using a network at a specific point in time may make it easier to detect security attacks.¹³⁷ (How the broad version of the end-to-end arguments affects security is discussed in chapter 9.) Finally, in an application-blind network, it is not possible to price-discriminate between users on the basis of the applications they use. (How differences in application awareness affect the pricing strategies available to a network operator is explored in chapter 6.)

The analysis in the preceding paragraph refers to a network that realizes the broad version of the end-to-end arguments in its pure form. Depending on a network's architectural design, it may be possible to collect some information about the applications running over the network in a way that, although it violates the layering principle, does not destroy the benefits of applying the broad version of the end-to-end arguments. Thus, depending on the architecture, it may be possible to ameliorate the problems with network management and security mentioned above while affecting architectural integrity only slightly. For example, in the architecture of the Internet, the Internet Protocol (IP) header and the transport protocol header contain information that can often be used to infer what

application is sending the data (e.g., the IP protocol field and the TCP/UDP/SCTP port numbers).¹³⁸ Although these fields were not intended to be interpreted by routers in the network, a packet filter operating at the Internet layer with access to these fields can nevertheless gather this information. Such packet filters are often used by network providers to collect information on user behavior or identify security attacks.¹³⁹ Although this violates the layering principle (the packet filter needs information about the format of transport-layer protocol data units, and accesses transport-layer protocol data units¹⁴⁰), the negative effect of this violation is marginal as long as the packet filter neither modifies nor acts on the data. Since the filter does not modify the higher-layer information or acts upon it, it does not affect the operation of higher-layer protocols. If existing higher-layer protocols are changed, or if new higher-layer protocols are added, the filter may collect useless information, but it does not impede or prevent the operation of these higher-layer protocols. Using such devices constitutes a trade-off between architectural purity on the one hand and network management and security on the other. In contrast, the architectural effect of devices that not only access but also modify or act upon information contained in higher-layer protocol data units is usually quite severe.¹⁴¹

Comparison of the Two Versions

The two versions of the end-to-end arguments differ considerably with respect to scope, content, and validity. In other words, they differ with respect to the range of functions to which they apply, the rules for the placement of these functions, and the justifications for these rules.

Scope During the design of a layered architecture, the complete functionality of the system must be assigned to layers. The narrow version applies only to a sub-group of this functionality: end-to-end functions (functions that cannot be completely and correctly implemented in the lower layers of the system). The narrow version does not offer any guidance on functions that do not have these characteristics. In contrast, the broad version guides the design of the system's complete functionality.

Content The two versions of the end-to-end arguments apply different criteria to determine the proper allocation of functionality. The narrow version distinguishes between "complete" and "partial" implementations of end-to-end functions (functions that can only be completely and correctly implemented in higher layers at the end hosts). The design rules

provided by the narrow version provide guidance not only with respect to the partitioning of functionality between the network and end hosts but also with respect to the allocation of functionality to different layers in the network or on end hosts. In contrast, the broad version distinguishes between “general” and “application-specific” functions and uses this distinction to allocate functions to the network or end hosts; it does not distinguish further between different layers in the network or on end hosts.

Though the two versions agree that functions that cannot be completely and correctly implemented in the network must be implemented on end hosts, they differ with respect to other functionality (table 2.1).

Table 2.1

Differences in design rules between the narrow version and the broad version.

	Narrow version	Broad version
Case 1: The function can be completely and correctly implemented at the higher layer and at the lower layer and is needed by all clients of the lower layer.	The function can be implemented at the higher or the lower layer.	The function can be implemented at the higher or the lower layer.
Case 2: The function can be completely and correctly implemented at the higher layer and at the lower layer and is not needed by all clients of the lower layer.	The function can be implemented at the higher or the lower layer.	The function should be implemented at the higher layer.
Case 3: The function can only be completely and correctly implemented at the higher layer.	The function must be implemented at the higher layer. An additional implementation of the function at the lower layer is possible for performance reasons; the decision about an additional implementation at the lower layer should be made on a case-by-case basis.	The function should be implemented at the higher layer. No additional implementation of the function at the lower layer for performance reasons.
Case 4: The function can only be completely and correctly implemented at the lower layer.	The function should be implemented at the lower layer.	The function should be implemented at the lower layer.

First, the narrow version does not say anything about functions that can be completely and correctly implemented in lower and higher layers of the system. The decision where to implement these functions is left to the system architects. As a result, a system based solely on the narrow version could implement these functions in lower or higher layers of the system.

In contrast, the broad version only allows implementing functions in lower layers of the system, if they are needed by all clients of the layer and can be completely and correctly implemented at that layer, or if they cannot be correctly implemented at the end hosts. Thus, a system that implements a function in a lower layer of the system that can be completely and correctly implemented in a lower and a higher layer, but is not needed by all clients of the lower layer would violate the broad version but not the narrow version.

Second, according to the narrow version, the designers of the system must decide in each case whether they should implement a necessarily incomplete version of end-to-end functions in lower layers of the system to improve the performance of the system. While the original paper shows a bias against such lower-layer implementations, it presents this choice as an open trade-off that system architects must resolve case by case. The performance improvement resulting from such an additional implementation is an important argument to be considered in this trade-off.

In contrast, the broad version has resolved the trade-off between evolvability and performance on a general basis, and it forbids the implementation of functions (including incomplete lower-level implementations of end-to-end functions) in lower layers of the system unless they are needed by all clients of these layers, owing to their negative effect on the evolvability of the system. The fact that such a design reduces performance has already been factored in the decision to use this design principle. If performance had been more important than evolvability, the system architects would not have chosen to use this design principle. Thus, the broad version entails a strong presumption that the performance gains of implementing application-specific functionality in the lower layers of the system do not outweigh the costs for the evolvability of the system. As a result, the performance gains resulting from such an implementation usually cannot be used to justify a deviation from this rule. Reasons that could justify a deviation from this rule may exist, but they must go beyond those that were already considered in the trade-off. In other words, these reasons must be so important or the circumstances must be so different that they justify reversing the trade-off that originally led to this design rule.¹⁴² For

example, as we will see in chapter 9, it may be justifiable to deviate from the broad version to make the Internet more secure.

If designers were allowed to decide whether to deviate from the broad version and implement application-specific functionality in the network case by case, the long-term evolvability of the system would not be adequately protected.¹⁴³ A case-by-case analysis biases system designers in favor of deviating from the broad version and quickly reduces the generality of the system. This is because of the problems associated with resolving the trade-off between evolvability and performance case by case. Whereas the benefits of deviating from the broad version (including the reduction in cost and the improvement in performance of certain applications) are immediately apparent, the costs associated with reducing the system's evolvability are difficult to determine because the applications that may suffer from the deviation are not yet known. This makes it impossible to determine whether and to what extent some of these future applications will be harmed. Since the applications are not yet known, their social value cannot be known either; this makes it impossible to determine what the ultimate costs to society will be. As research in behavioral economics has shown, humans tend to assign disproportionately more weight to present benefits (i.e., the performance optimization or cost saving in the present) than to future benefits (i.e., the ability to use unknown applications in the future), which makes it more likely that designers will deviate from the broad version. That the future benefits are uncertain aggravates this bias. Moreover, designers working on a specific problem may lack the architectural knowledge to adequately assess the architectural effect of their solution. For example, when firewalls and network-address translators (which both deviate from the broad version) were first deployed, their negative consequences for the evolvability of the network were not immediately recognized. It took a while to understand why and how they affect different classes of applications. Though the effect of a specific deviation may seem small, the interactions between several deviations quickly erode the evolvability of the system.¹⁴⁴ For example, network-address translators and firewalls together have made it much more difficult to develop certain kinds of applications, and almost impossible to deploy new transport-layer protocols in the Internet, severely constraining the evolvability of the Internet.¹⁴⁵

Finally, the two versions constrain applications in different ways. As we will see below, the broad version of the end-to-end arguments requires that applications be implemented in higher layers on end hosts. In contrast, the narrow version may require designers to implement certain end-to-end functions in the application itself.

Validity The two versions are based on different justifications. The narrow version is based on an argument about correctness: if designers want to build a correct system (one that behaves as required), they must follow the first design rule embodied in the narrow version and implement each end-to-end function at a higher layer at the end hosts, where it can be completely and correctly implemented. Whether an additional, necessarily incomplete version of an end-to-end function is implemented at a lower layer does not affect the correctness of the system. The second design rule of the narrow version presents this question as an open trade-off that must be resolved by the system architects case by case.

The broad version is based on different considerations. For all functions other than end-to-end functions (which must be implemented end-to-end for reasons of correctness anyway), the broad version's design rules emerge from a trade-off: they reflect the decision to prioritize long-term system evolvability, application autonomy, and reliability over short-term performance optimizations.¹⁴⁶

Owing to these differences, both versions offer different amounts of freedom to system architects. Since designers usually want to build a correct system, system architects and subsequent designers must follow the first rule of the narrow version. In contrast, system architects can choose freely whether or not they want to use the broad version. If they do not agree with the trade-off struck by the broad version, they can choose a different design principle and still build a correct system. Once the system architects have chosen the broad version, however, subsequent designers must comply with the rules established by that principle or else the long-term evolvability of the system will suffer.

Rationale for Separation Even if the authors of the “end-to-end papers” did not intend to propose two different (though related) design principles, it is necessary to distinguish between those two design principles.

Owing to the differences in the content of the rules, both versions constrain the design of an architecture in different ways. Often an architecture is compliant with one version but not with the other. (We will encounter examples in chapter 3.) A clear conceptual separation between the two principles makes it easier for designers to recognize whether none, one, or both of them must or could be applied to solve their particular design problem.

A clear conceptual separation between the two versions would avoid unnecessary arguments about the “correct” application of “the end-to-end argument” and direct attention to the real question: how to resolve the

trade-offs associated with the placement of functionality. Over the years, both versions have been adopted by the scientific community as “the end-to-end argument.” Whereas general books on network architecture often present the narrow version, texts published by the Internet Engineering Task Force or the Internet Architecture Board and papers on Internet policy usually are based on the broad version. Owing to the different rules embodied in the two versions, two designers can apply “the end-to-end argument” to assess a technical solution and arrive at contradicting results. Thus, contradicting views of compliance with the end-to-end arguments often arise because people use different versions of the end-to-end arguments, not because the same principle is incorrectly applied. In this case, the real issues lie elsewhere. The designers disagree about the proper resolution of the trade-off governing the placement of functionality, and they should argue about the prioritization of values, not about the meaning of the end-to-end arguments.

Failing to recognize the narrow version as a distinct design principle may even lead to incorrect systems. Designers can disagree about resolving a trade-off, but they cannot disagree about the requirements for correctness. Even someone who does not think that application autonomy, reliability, and long-term system evolvability are more important than short-term performance optimizations must comply with the narrow version and implement end-to-end functions end-to-end; otherwise the system will not execute correctly. If the two versions are not separated, there is a danger that designers will not recognize that the requirements for correctness established by the narrow version are independent of the broad version, and that, because they disagree with the trade-off that underlies the broad version, they will not comply with the narrow version.

Because recent developments in the architecture of the Internet differ with respect to the challenges they pose to both arguments, the differences between the two versions may become even more relevant in the future. As will be discussed in the concluding chapter, some architectural solutions contradict one version but comply with the other. In this case, it is important to be aware of the differences between the two versions so as to identify the issues posed by the problem under consideration.

A clear distinction would also resolve some common policy disputes. As was noted above, proponents of network neutrality regulation usually point to the beneficial effect of network neutrality on innovation. Network neutrality, however, does not follow from the *narrow* version of the end-to-end arguments—it is a consequence of applying the *broad* version. The argument in favor of regulation is based on the benefits of applying the

broad version of the end-to-end arguments and on the negative consequences of deviating from that version. Opponents of regulation argue that proponents have extended the end-to-end arguments beyond their original meaning, referring to descriptions of the narrow version.¹⁴⁷ Thus, although both sides start from original descriptions of the end-to-end arguments, they arrive at contradicting results. This puzzle might be resolved quickly if the two sides would recognize that they are not using the same argument but rather two different versions of it.

3 The Original Architecture of the Internet

This chapter outlines how the layering principle and the end-to-end arguments shaped the original architecture of the Internet. To situate the discussion, the first section provides an overview of the original architecture of the Internet. (Readers already familiar with the Internet's architecture may skip that section.) The second section discusses the effect of the layering principle and the two versions of the end-to-end arguments on the Internet's original architecture. The final section discusses some common misconceptions about the meaning of the end-to-end arguments and their relationship to the original architecture of the Internet.

Introduction to the Original Architecture of the Internet

The Internet aims to provide universal communication services to applications running on hosts attached to distinct but interconnected networks.¹ A network provides universal communication services if any host can communicate with any other host no matter what network they are attached to.²

Many different network technologies can connect computers. For example, wireless networks in homes or at hotspots usually use the IEEE 802.11 family of standards, better known as Wi-Fi technology, to connect computers wirelessly within a local area.³ Companies or universities often use Ethernet technology to connect the computers that constitute their local network.⁴ A network that uses a specific technology to provide connectivity between the computers attached to it is often called a *physical network*.⁵ Whereas the technologies and protocols used by a physical network let the network deliver data packets to any computer attached to that network, the differences between the technologies used by different physical networks (such as the different addressing schemes used by various physical network technologies) prevent hosts attached to different

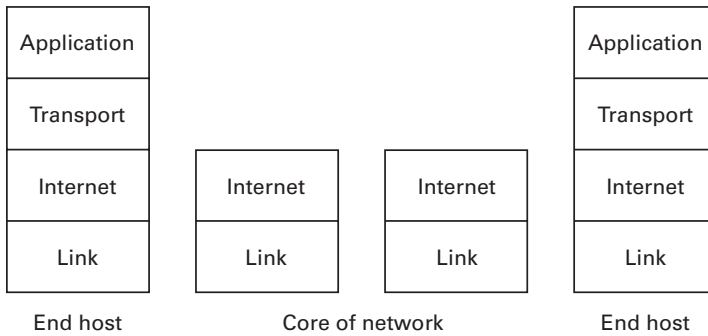


Figure 3.1
Layers in the Internet architecture.

physical networks from communicating with one another. The Internet⁶ is designed to solve this: it connects different physical networks using a set of conventions that let computers attached to these networks communicate.⁷

To provide universal communication services across distinct, interconnected networks, the Internet's architecture partitions network functionality into four layers, with one or more protocols implementing the functionality assigned to each layer (figure 3.1).⁸ Each layer provides communication services to the next higher layer at a decreasing level of abstraction. To implement these services, each layer uses the services provided by the layer below. The lowest layer is called the *link layer*.⁹ Moving upward, it is followed by the *Internet layer*, the *transport layer*, and the *application layer*. The family of protocols used in the global Internet is called the *TCP/IP protocol suite* after its two main protocols, the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

The Link Layer

The link layer contains a wide variety of protocols responsible for transporting data packets to any computer attached to a physical network or for transporting data packets across a point-to-point link.¹⁰ All computers belonging to a particular physical network run the same network protocol. Thus, the collection of protocols at this layer corresponds to the set of physical network or point-to-point technologies that can be attached to the global Internet. The Wi-Fi or Ethernet protocols mentioned above are examples of protocols at this layer.

In the context of the reimbursement-processing example in chapter 2, the link layer consists of the technologies that deliver an envelope from

one mail-processing center to the other—airplanes, trains, trucks, bikes, and so on.

The Internet Layer

The Internet layer consists of a single protocol: the Internet Protocol (IP), which transports data packets from one host to another across any set of interconnected networks. The Internet Protocol enables any pair of hosts to exchange data packets, independent of the particular physical network to which they are attached.

The Internet layer is implemented on end hosts at the edge of the network and on special computers called *routers* in the core of the network; it is the highest layer that must be implemented on computers in the core of the network. Layers above the Internet layer only have to be implemented on end hosts; they do not have to be implemented on computers in the core of the network.¹¹

To higher layers, IP provides connectionless, unreliable, best-effort delivery of datagrams from end host to end host.¹² Datagrams—data packets in a specific format defined by IP—are the basic unit of data transfer across interconnected networks in a TCP/IP Internet. The delivery is connectionless: instead of establishing a virtual connection through the network from source host to destination host over which all datagrams can travel, each datagram is treated independently from all others and travels separately through the Internet. As a result, two successive packets from one host to another may follow entirely different paths. The service is unreliable in that delivery is not guaranteed. If datagrams are lost, damaged, duplicated, delayed, or delivered out of order, IP does nothing.¹³ It merely transmits any datagram arriving at the destination host to the appropriate protocol in the higher layer. Finally, the service model is called *best-effort* because, although IP does its best to deliver datagrams, it does not provide any guarantees regarding delays, bandwidth, or losses.

To implement the aforementioned services, IP specifies an addressing scheme (a way to identify all hosts in the Internet), a method of interconnection between different physical networks, and a set of rules and procedures for moving datagrams from origins to final destinations.¹⁴ IP's addressing scheme can be used to derive unique identifiers—"IP addresses"—for each host and each router in the Internet.¹⁵

As has already been noted, IP interconnects physical networks through routers in the core of the network. A router links two or more physical networks, receiving datagrams from one network and sending them to another. Typically, a sending host sends a datagram to the nearest router.¹⁶

If the destination host is attached to one of the physical networks to which the router connects, the router delivers the datagram to the destination end host directly. If not, the router uses the destination address to compute the next router on the path and forwards the datagram to that router. Thus, the datagram travels through the Internet from router to router until it reaches a router that can deliver it directly. At each step, the IP implementation currently holding the datagram uses the service of a physical network at the link layer to move the datagram across that physical network to the next IP implementation on the path.

The postal layer in the reimbursement example in chapter 2 operates in a similar way. At each postal processing center, someone uses the postal address to determine the next stop on the way from the original sender to the ultimate destination. The postal layer then uses the services of the lower layer (trucks, airplanes) to deliver the postal envelope to the next postal location. This process is repeated until the envelope reaches its ultimate destination. Similar to the IP protocol, the basic service of the postal service does not make any guarantees as to the timing of delivery. Envelopes may get lost or damaged or may arrive in a different order (for example, envelopes sent on the same day may arrive on different days, or an envelope sent later may arrive earlier).

The Transport Layer

The Internet Protocol delivers datagrams from one host to another—it does not distinguish between different applications on the same host. Transport-layer protocols deliver data to and from the applications within an end host. They pick up data from the sending application at the end host at the origin and deliver the data to the receiving application at the destination host. In other words, transport-layer protocols are responsible for differentiating among multiple sources or destinations within one host, letting multiple application-level processes¹⁷ on the same end host use the network simultaneously.¹⁸ Transport-layer protocols therefore ensure that application programs on different end hosts can communicate.¹⁹

Whereas IP is implemented on the routers in the core of the network and on end hosts, transport-layer protocols only have to be implemented on end hosts.²⁰ The transport-layer protocol implementations on the sending and receiving end hosts exchange messages with each other to implement the protocol's services.

Transport-layer protocols build on the services of the Internet Protocol at the Internet layer to provide additional services that may be useful for particular applications. Thus, the transport layer is the lowest layer

implementing application-specific functionality. As we will see below, this division of responsibility follows the broad version of the end-to-end arguments.²¹

The most popular transport-layer protocol, the Transmission Control Protocol (TCP), provides a reliable, connection-oriented byte stream.²² TCP lets two application-layer processes on different end hosts establish a connection, send and receive data over this connection, and close the connection. The connection lets the application-layer processes send and receive a continuous byte stream (i.e., the sender writes bytes into a TCP connection, which the receiver reads out of the TCP connection). TCP does not interject any boundaries between bytes, but treats the sequence of bytes passed to it by the sender as a continuous stream. TCP ensures that data handed to it by the sending higher-layer process arrives at the receiving higher-layer process without errors or duplicates and in the correct order.

In contrast, the User Datagram Protocol (UDP) provides an unreliable datagram service to application-layer processes, where data may arrive at the next higher layer out of order, duplicated, damaged, or not at all. Unlike TCP, UDP accepts and delivers separate messages, not a continuous data stream.²³

Similarly, the postal service in the reimbursement example merely delivers mail from one mailroom to the other. It does not deliver mail to specific people within the building. This is the task of the next higher layer, the mailroom layer, which picks up mail from the sender and hands it over to the postal service at the sending location and which receives mail from the postal service at the receiving location and delivers it to the receiver. Thus, the mailroom layer builds on the mailroom-to-mailroom service of the postal layer to provide person-to-person communication services to the highest layer. Like transport-layer protocols, members of the mailroom staff work in the office building at the origin and at the destination, but not in the postal processing centers in between. In our example, there is only one mailroom-layer protocol that provides a service similar to UDP: it accepts and delivers separate pieces of mail. The members of the mailroom staff do not implement additional procedures to ensure that the mail really does arrive at the receiver, or that separate pieces of mail sent between the same two people arrive in the order in which they were sent.

The Application Layer

The application layer contains a range of protocols that let applications communicate with one another. E-mail, the World Wide Web, Internet-telephony applications (e.g., Vonage and Skype), file-sharing applications

(e.g., Napster, Kazaa, BitTorrent), and streaming video applications (e.g., Real Player, QuickTime, Windows Media Player) are all examples of applications operating at this layer. An application protocol defines the rules and conventions that govern the interactions among different parts of a distributed application.²⁴ To communicate with an application on a different end host, both application programs exchange data according to the format and conventions specified in the application protocol.²⁵ For example, the Web browsers Mozilla Firefox and Internet Explorer are application programs, running on an end host, that communicate with Web servers—application programs running on a different end host—to show users webpages. The browser contacts the server and asks for the desired page; the server then sends a copy of that page back to the browser. Web browsers and Web servers can cooperate in this way because they communicate in accordance with the Hypertext Transfer Protocol (HTTP), which defines the format and the procedure for Web browser-server interactions.

Any distributed application has to define rules governing the interactions among its parts; otherwise, the distributed parts will not be able to communicate or cooperate to accomplish a task. As a result, any designer who builds a distributed application based on the TCP/IP protocol suite specifies an application-layer protocol. For example, the rules that govern the communication among Skype programs operating on different machines form an application-layer protocol. Skype keeps this protocol proprietary, so only Skype implementations produced by Skype can communicate with each other. In contrast, a number of application-level protocols for such popular applications as e-mail and file transfer have been standardized by the Internet Engineering Task Force (IETF), by the World Wide Web Consortium, or by some other standards body to let programs communicate with each other using standardized protocols, no matter who designed them. As a result, an e-mail program produced by one vendor can send e-mail to an e-mail program produced by another vendor if both programs conform to the same application-layer protocol (usually the Simple Mail Transfer Protocol).²⁶

The Internet and the Layering Principle

Apart from specifying the division of functionality between layers, the Internet's architecture imposes constraints on the interactions between protocols belonging to different layers and on the number of protocols that may populate a layer.

Whereas lower layers are not allowed to use the services of higher layers, higher layers are allowed to invoke the services of protocols in the same layer or any layer below them, down to the Internet layer. In other words, layers above the Internet layer are not allowed to make direct use of the services of a layer below the Internet layer.²⁷ Thus, the Internet’s architecture realizes relaxed layering with a portability layer at the second-lowest layer, the Internet layer (figure 3.2).²⁸

Although the Internet layer is restricted to a single protocol—the Internet Protocol—other layers may host a variety of protocols.²⁹ As described above, the link layer contains a wide variety of protocols that correspond to the different types of physical networks over which the Internet Protocol can run. Above the Internet layer, each layer is allowed to contain various protocols, which in turn may support multiple protocols at the next higher layer. The resulting structure of protocol dependencies is often likened to an hourglass, with the Internet Protocol as the hourglass’s waist (figure 3.2).³⁰

This design isolates the transport layer and the application layer from changes below the Internet layer. Owing to the use of the Internet layer as a portability layer, the visibility of the link layer is restricted to the Internet layer.³¹ Thus, as long as the service interface provided by the Internet Protocol does not change, the Internet Protocol can take

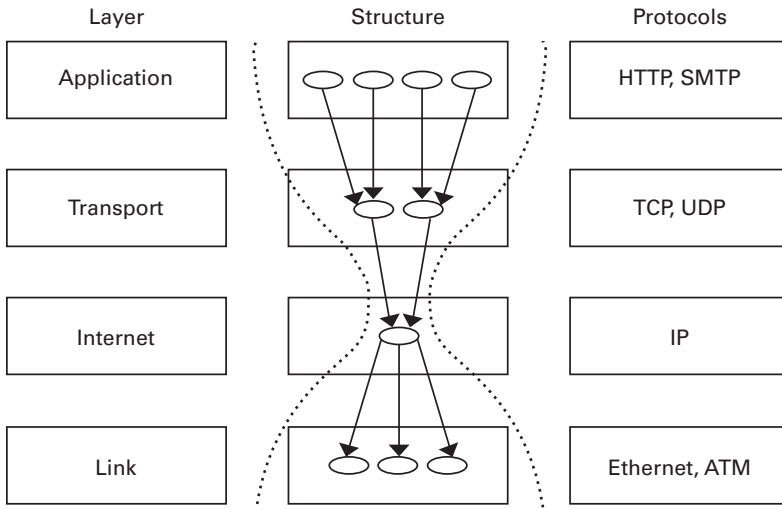


Figure 3.2
Layering in the Internet architecture.

advantage of new physical network infrastructures and new transmission and link technologies without needing changes in the higher layers.

At the same time, since lower layers are not allowed to use the services of higher layers, the Internet layer and the link layer remain unaffected by innovations in the transport and application layers. As a result, the Internet layer divides the protocol stack into two quasi-independent subsystems that can evolve separately.

The Internet and the End-to-End Arguments

The End-to-End Arguments and the Division of Functionality between the Internet Layer and the Transport Layer

During the initial design of the Internet's architecture, end-to-end arguments shaped the division of functionality between the Internet layer and the transport layer. Whereas the narrow version required that functions such as reliable data transfer, duplicate suppression, and reordering be implemented end-to-end between the original source and ultimate destination of data (i.e., end-to-end between end hosts), the broad version influenced the general division of functionality between the two layers. Although the two design principles had not been formally identified, named, and described as design principles when the Internet's original architecture was developed (the relevant papers were published in 1981 and 1998³²), the Internet's architects implicitly applied and followed the rules that later came to be known as (the two versions of) "the end-to-end arguments."³³

The Narrow Version The narrow version of the end-to-end arguments postulates that, since certain functions cannot be completely and correctly implemented in lower layers of the network, they must be implemented end-to-end between the end hosts that are the original source and the ultimate destination of data. As described above, error control, duplicate suppression, and reordering are functions that exhibit this characteristic. Performing these functions hop-by-hop in the Internet layer would not guarantee their correct execution. Consider reliable data transfer. If the Internet Protocol had implemented hop-by-hop error control between hosts and routers and between routers, datagrams would have been transported correctly from host to router, from one router to the next, and from router to host. Datagrams could still be corrupted at the routers themselves (e.g., as a result of memory bit errors in the routers). Only an end-to-end error control covers the entire communication path from source host to destina-

Box 3.1

The Decision to Implement Error Control End-to-End

The decision by the Internet's architects to implement error control end-to-end between the two end hosts involved in the connection constituted a deliberate departure from other network architectures. For example, the Arpanet Host-to-Host Protocol (AHHP) did not use end-to-end acknowledgment and retransmission, but relied on the lower-layer subnet protocols and the host-to-subnet protocols for reliable delivery. As some of the researchers involved in the design of the Internet noted, this design prevented the AHHP from fully guaranteeing reliable delivery (framed in the terminology of the narrow version, they argued that this design does not constitute a complete and correct implementation)—a deficiency the Internet Transmission Control Protocol was designed to correct by performing error control and retransmission end-to-end between the end hosts.^a Similarly, the X.25 architecture, an alternative network architecture favored by traditional telecommunication carriers, originally did not offer end-to-end error control and retransmission between the computers attached to the network, but instead relied on the communication subnet to deliver data reliably. Researchers involved in the design of the Internet criticized this choice, arguing that the lack of end-to-end integrity checks made it impossible for X.25 networks to guarantee end-to-end reliability (framed in the terminology of the narrow version, they argued that only an end-to-end implementation of the function could completely and correctly implement it).^b Although the decision by the Internet's architects to implement error control end-to-end seems obvious today, it was controversial at the time.^c

a. See Garlick, Rom, and Postel 1977, pp. 1–2; Abbate 1999, pp. 127–128.

b. See Bochmann and Goyer 1977, pp. 15–16, 26–29; Cerf and Kirstein 1978, pp. 1397, 1402–1403; Pouzin and Zimmerman 1978, p. 1367; Abbate 1999, p. 158.

c. See Bolt, Beranek and Newman Inc. 1974 (defending Arpanet) or Blackshaw and Cunningham 1980 (defending X.25).

tion host. On the basis of this insight, the Internet's architects decided to implement error control end-to-end between the end hosts (box 3.1).

Functions such as error control and duplicate suppression were originally implemented end-to-end at the Internet layer as part of the Internet network Transmission Control Protocol, but were later split from the Internet layer and moved to a separate transport-layer protocol called the *Transmission Control Protocol* (TCP).³⁴ If one assumes that transport-layer protocols

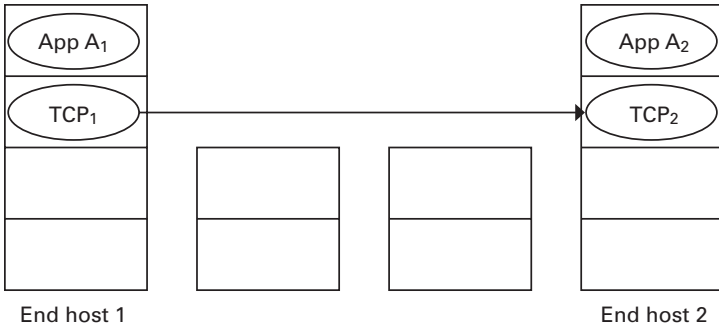


Figure 3.3
TCP operates end-to-end between end host 1 and end host 2.

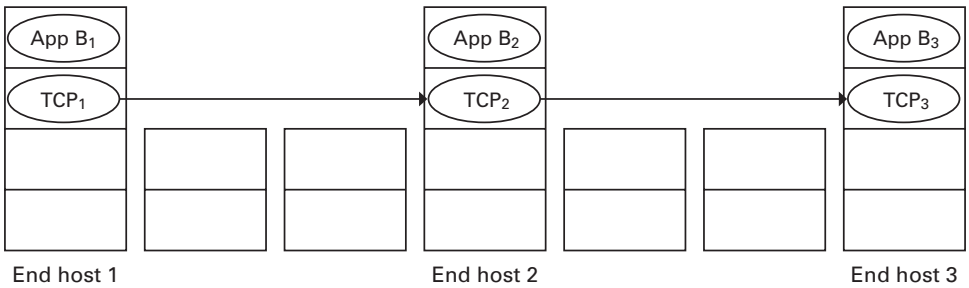


Figure 3.4
TCP operates hop-by-hop between end host 1 and end host 3.

operate end-to-end between the original source and the ultimate destination of data, placing end-to-end functions in a transport-layer protocol such as TCP is compliant with the narrow version (figure 3.3). If this assumption does not hold, end-to-end functions must be implemented at a higher layer that truly operates end-to-end. Ultimately, the appropriate place may be the application itself. For example, if the original source and the ultimate destination of data are connected by a concatenated sequence of transport-layer connections, the transport layer operates hop-by-hop between the original source and the ultimate destination of data (figure 3.4). For example, e-mail or the World Wide Web may exhibit this structure. (For a description of the structure of e-mail, see box 3.6.) As was explained above, however, for end-to-end functions such as reliable data transfer, a concatenation of hop-by-hop functions at the lower layer is not equivalent to a single end-to-end execution of the function at the higher layer.³⁵

Box 3.2

Error Control above the Transport Layer in the Current Internet

If data may get corrupted after leaving the transport layer, the narrow version requires that error control be performed at a layer above the transport layer, at or above the layer at which data may get corrupted. For example, in their 1984 paper describing the narrow version, Saltzer, Reed, and Clark argue that a file-transfer application must implement end-to-end error control itself in order to eliminate all threats to data integrity that may occur after the data leaves the communication system—for example, when the data is stored on the disk.^a In spite of this insight, most present-day applications do not implement such functionality themselves but rather use the reliable data-transfer functionality provided by transport-layer protocols such as TCP. On the one hand, implementing integrity checks is difficult and costly. On the other hand, the remaining threats to data integrity in the end system are negligible in regard to the needs of most applications. Thus, for most applications the costs of providing error control in the application outweigh the benefits of doing so.^b In accordance with the narrow version of the end-to-end arguments, however, it is usually strongly recommended that mission-critical applications that depend on data integrity implement their own integrity checks.^c

a. See Saltzer, Reed, and Clark 1984, p. 280; Border et al. 2001, section 4.1.3.

b. Moors 2002, pp. 1215–1216. The decision where to implement error control is also related to trust (Clark and Blumenthal 2007).

c. See Stone and Partridge 2000, sections 5.4 and 6; Moors 2002, pp. 1215–1216; Border et al. 2001, section 4.1.3.

Placing an end-to-end function in a transport-layer protocol also fails to constitute a complete implementation of the function if events falling within the function's domain occur after the data leaves the transport layer. How likely this is to happen must be judged for each application (box 3.2). If, for a specific application and a specific function, events falling within that function's domain occur after the data leaves the transport layer, the narrow version requires that the functionality be implemented at or above the layer where the event occurs, potentially in the application itself.³⁶ Although the implementation of the function provided by TCP constitutes only a partial implementation for that application, the application can use TCP's services to guarantee the function's execution across the network and up to the transport layer. Providing reliable data transfer or other end-to-end functions across a communication network is a very

complex task that requires sophisticated mechanisms and functionality. Using TCP may thus greatly simplify the end-to-end implementation of the function at the higher layer.³⁷ Although TCP may not be able to provide a complete implementation of end-to-end functions for all applications, implementing this functionality in a transport-layer protocol lets different applications share this functionality without negatively affecting applications that do not need it. Applications that do not need the functionality and do not want to pay its costs can simply use another transport-layer protocol—one that is more appropriate to their needs. Thus, for applications that must implement an end-to-end function in the application itself to guarantee a function's correct execution, the function's implementation in TCP is a partial implementation that, according to the narrow version, can be justified as a performance enhancement.³⁸

Thus, the narrow version shaped the Internet's original architecture by requiring that end-to-end functions such as error control, duplicate suppression, and reordering be implemented end-to-end between the original source and the ultimate destination of data. However, the narrow version does not explain two other, related choices made during the design of the original architecture: the decision to split the original Internetwork Transmission Control Protocol at the Internet layer into two protocols (the Internet Protocol at the Internet layer and the Transmission Control Protocol at the transport layer) and the decision not to implement hop-by-hop error control in the Internet layer.

The researchers working on the Internet's architecture initially proposed an *Internetwork Transmission Control Protocol* for the Internet layer (figure 3.5, left). This protocol combined the hop-by-hop aspects of datagram transfer through the Internet and end-to-end implementation of reliable data transfer, duplicate suppression, and other end-to-end functions. It transferred datagrams hop-by-hop, but it performed error control, duplicate suppression, or packet reordering end-to-end, combining the functionality of today's Internet Protocol and Transmission Control Protocol in a single protocol.³⁹ Under this proposal, all applications that wanted to use the Internet had to use the services of the Internetwork Transmission Control Protocol, including the parts of the protocol that provided connection-oriented reliable data transfer.⁴⁰ This solution complied with the narrow version. As indicated above, the narrow version allows the implementation of functions in a lower layer in the network if the function can be completely and correctly implemented at that layer. Reliable data transfer and duplicate suppression are end-to-end functions that can be completely and correctly implemented only end-to-end between the original source and

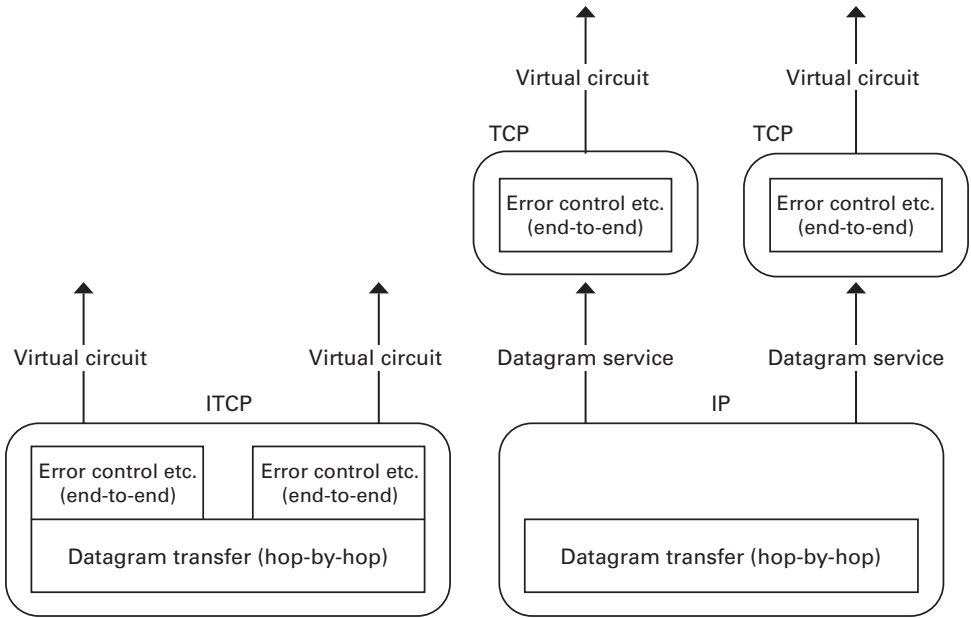


Figure 3.5
Splitting ITCP.

ultimate destination of data. Since the proposed monolithic protocol implemented these functions end-to-end between the sending and receiving end hosts, it implemented them completely and correctly.⁴¹

Although the Internetwork Transmission Control Protocol complied with the narrow version, the Internet’s architects decided to split the protocol into two separate protocols: the Internet Protocol and the Transmission Control Protocol that we know today (figure 3.5, right).⁴² In this architecture, end-to-end functions such as reliable data transfer or duplicate transmission are implemented as part of the Transmission Control Protocol at the transport layer. As we saw earlier in this section, this solution also complies with the narrow version.

In addition, the Internet’s architects had to decide whether to implement hop-by-hop error control at the Internet layer. Since error control can be completely and correctly implemented only end-to-end, hop-by-hop error control at the lower layer provides only an incomplete implementation of this function. The narrow version explicitly allows (but does not require) such an additional implementation to optimize performance, but leaves this decision for the system architects to make in each individual

Table 3.1

Internet design decisions that only the broad version can explain.

Decision	Does solution comply with narrow version?	Does solution comply with broad version?
One monolithic Internet-layer protocol, combining functionalities of TCP and IP	Yes	No
Split TCP from IP	Yes	Yes
Decision	Does solution comply with narrow version?	Does solution comply with broad version?
Hop-by-hop error control in Internet layer	Yes	No
No hop-by-hop error control in Internet layer	Yes	Yes

case. Although the designers of the Internet's architecture chose not to implement such functionality at the Internet layer, a decision to implement hop-by-hop error control at this layer would have complied with the narrow version.⁴³

In neither of the cases discussed above can the narrow version explain why the Internet's architects chose one solution over the other, since all the options under consideration complied with the narrow version. As will be shown in the next subsection, these choices were driven by the broad version of the end-to-end arguments (table 3.1).

The Broad Version The broad version of the end-to-end arguments argues that application-specific functionality usually cannot—and preferably should not—be implemented in the lower layers of the network, the network's core. Instead, a function should be implemented in a network layer only if it can be completely and correctly implemented at that layer and is used by all clients of that layer. Thus, lower layers, or the core of the network, should provide only general services of broad utility across applications, whereas application-specific functionality should be implemented in higher layers at the end hosts.

Separation of the Internet Layer and the Transport Layer The initial proposals for an Internetwork Transmission Control Protocol combined the hop-by-hop aspects of datagram transfer through the Internet with the end-to-end implementation of functions such as reliable data transfer and

duplicate suppression in a single protocol.⁴⁴ In this design, all applications that wanted to use the Internet had to use the Internetwork Transmission Control Protocol, which provided a single service: connection-oriented reliable data transfer (or a “virtual circuit”).⁴⁵

Some of the researchers opposed this design. They argued that reliable data transfer and connection orientation are application-specific functions that are needed by some but not all applications, and that implementing these functions in the Internet layer may be inefficient for, and may even hurt, applications that do not need them.⁴⁶ The 1998 paper describing the broad version later presented this insight—i.e., that performing application-specific functions in the network may harm or even rule out applications that do not need them—as one important argument for keeping application-specific functions out of the network.⁴⁷ Applications have very different requirements regarding reliability, throughput, and delay.⁴⁸ For example, e-mail and file transfer require reliability but are less sensitive to delay than some other applications. Real-time transmission of uncompressed voice or video can tolerate a certain amount of data loss, but suffers from increased delay (or variability in delay).⁴⁹ Ensuring reliable data transfer by performing end-to-end error control and retransmission usually increases delay (and variability in delay).⁵⁰ Thus, researchers working on real-time delivery of voice argued, forcing all applications to use reliable data transfer reduces the performance of applications such as real-time delivery of voice that do not need perfect reliability but suffer from increased delay.⁵¹ Similarly, to provide connection-oriented data transfer, a bi-directional virtual path through the network is established before the actual data transmission starts. Establishing the connection takes time and requires the exchange of several control messages between the sender and the receiver before the actual data transmission can start.⁵² Again, the latency of connection set-up and the corresponding signaling overhead may hurt some applications that do not need connections. For example, as researchers working on local-area networks pointed out, when computers communicate with one another they often exchange single packets (not streams of packets) in patterns that differ from the bi-directional point-to-point connections needed by such traditional applications as remote log-in and file sharing.⁵³ For these types of computer-to-computer communications, setting up a connection to send a single packet creates substantial overhead for a service that the application does not need.

According to the critics, removing all application-specific functions from the Internet layer by splitting the Internetwork Transmission Control Protocol into two protocols and providing only unreliable datagram service in

the Internet layer would enable the network to support a broader range of applications. In comparison with virtual circuits, they argued, unreliable datagram service is a more elemental building block out of which a broader range of services can be built.⁵⁴ A network offering unreliable datagram service at the Internet layer can support applications that require connection orientation and reliability as well as applications that do not need these functions. For applications that require connection orientation and reliability, transport-layer protocols at the end points can construct a virtual circuit based on an the unreliable datagram service at the Internet layer.⁵⁵ Applications that do not require these functions can avoid the associated delay and overhead by choosing a different transport-layer protocol that better meets their needs—for example, one that lets them send individual datagrams without guaranteeing their delivery. In contrast, a network offering a virtual circuit service at the Internet layer cannot efficiently support applications that do not require connection orientation or reliability (i.e., the non-connection-oriented computer-to-computer communications mentioned above) and may even rule out applications (i.e., real-time delivery of voice) that suffer from increased delay. Higher-layer protocols at the end hosts cannot construct an efficient datagram service based on a virtual circuit at the Internet layer, since the delay and the overhead introduced by the Internet layer cannot be removed at a higher layer.

On the basis of these considerations, the Internet's architects decided to split the Internetwork Transmission Control Protocol into two protocols that were allocated to different layers: the Internet Protocol, which offered unreliable datagram service, and the Transmission Control Protocol, which provided connection-oriented reliable data transfer (box 3.3, figure 3.5).⁵⁶ Under the new design, which became the Internet's original architecture, the Internet Protocol, the only protocol at the Internet layer, is designed to provide a basic building block of broad utility across applications.⁵⁷ Transport-layer protocols can use this building block to construct more elaborated services better suited to specific classes of applications.⁵⁸ Thus, the transport layer is the lowest layer that implements application-specific functionality; it is also the lowest layer that only has to be implemented on end hosts (figure 3.6). Although the Internet's architects specified only two transport-layer protocols in the original architecture (the Transmission Control Protocol and the User Datagram Protocol, the latter of which offers unreliable datagram service), they envisaged that more transport-layer protocols would be added over time.

In sum, although the broad version of the end-to-end arguments had not been formally identified and described as a general design principle

Box 3.3

The Decision to Split TCP and IP

The decision to split the initial monolithic protocol into two separate protocols was made at a meeting of the researchers involved in the design of the Internet Transmission Control Protocol on January 30–31, 1978, at the Information Sciences Institute in Marina Del Rey.^a In an e-mail exchange with Bob Frankston and Vinton Cerf, David Reed (one of the authors of the papers describing the end-to-end arguments) later described the events that led to this decision^b:

There were a small number of proponents for specifying the Internet protocols as based on datagrams, with the idea of reliable in-order delivery being an “application layer” mechanism. This all came together at the meeting I attended in Marina del Rey. John Schoch, Danny Cohen and I each presented arguments to that effect, from different points of view. John argued based on the PUP architecture, which was an architecture based on datagrams, where streams were one option among many.^c

Danny argued for the idea that packet speech did not want retransmission, but instead just a sequence-numbered stream of packets, where non-delivery was an option because latency was the key variable and the application could fill gaps. I argued that non-connection based computer-computer protocols, such as those we were developing for interconnected LANs, could be more general, and that end-to-end reliability was ultimately up to the endpoints to assure—for example, there were useful protocols that involved a packet from A to B, handing off of the request from B to C, and a response back from C to A were quite useful, and would be end-to-end reliable at the application level, while gaining little or no benefit from low level “reliability” guarantees. Other protocols, such as multicast, etc. were essentially datagram-oriented. I remember arguing quite strongly that you could support streams on top of datagrams, but by requiring a streams, you’d never get effective or efficient datagram services. Danny equally argued that reliable streams would create latency variability (jitter) where none was necessary. John Schoch argued that PUP was datagram-based, with streams built on top, and that architecture was quite effective. . . .

As I recall, we 3 people, plus Steve Crocker, conspired to argue that we needed a datagram service so we could continue to do research on more general protocols, and in the heat of the argument proposed why not split out the addressing layer from the stream layer, just as we had split the TCP from the functions of the Telnet layer. . . . In the heat of the Marina del Rey meeting, we 3 (sitting next to each other) agreed to push for splitting the TCP packet into two layers, the routing header and the end-to-end stream payload. This resulted in a sidebar meeting in the hall during the next break, where I remember it was Jon [Postel], you [Vinton Cerf], Danny [Cohen], me, Steve [Crocker], and John Schoch, and you [Vinton Cerf] agreed that we should try defining how we’d split the layers, and see if the overhead would be significant. This resulted in 3 protocols, IP, TCP, and UDP (for us datagram nuts). Danny went off being happy that he could define a packet speech protocol layered on UDP, I went off happy that I didn’t need to pursue DSP^d anymore, but could focus on how to use UDP for protocols like TFTP, which we built at MIT shortly thereafter.”

Box 3.3

(continued)

a. See Postel 1978b, p. 1 (describing the results of the TCP meeting) and Postel 1977b, p. 8 (noting the location of the next meeting). See also Abbate 1999, pp. 129–130.

b. Peter, no date.

c. For a description of the Pup architecture, see Boggs et al. 1980.

d. DSP (which stands for “Data Stream Protocol”) was a LAN-centric Internetworking protocol that David Reed had developed at MIT to allow for both datagram-like and stream-like behavior to coexist (Peter, no date). According to Reed (2009a), “DSP was developed as part of our research work at MIT on protocols for high speed networks to support distributed computing applications. Bob Kahn had insisted that we unify our research with the ongoing TCP project as a condition of funding—because the goal of TCP was to interconnect every kind of transport and every kind of application, he wanted to understand how to make TCP a universal interoperability solution.”

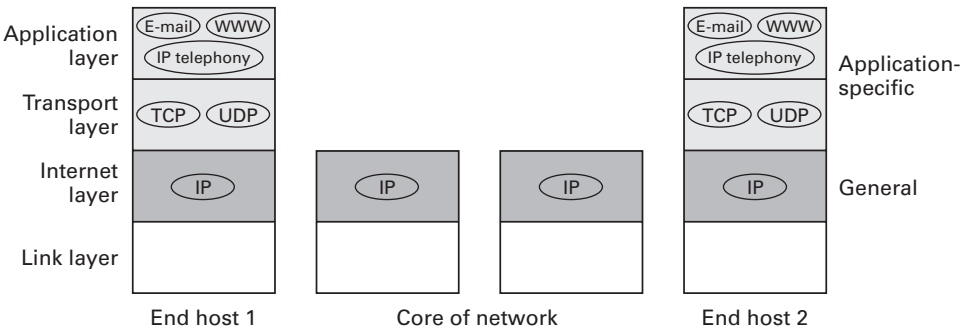


Figure 3.6 Distribution of functionality in the Internet’s original architecture.

when the Internet’s architects decided to split TCP and IP, they implicitly applied the rules established by the broad version. First, the Internetwork Transmission Control Protocol implemented application-specific functionality in the network, which violates the broad version. The architects rejected this solution. Second, their decision to offer unreliable datagram service at the Internet layer instead of the virtual circuit service provided by the Internetwork Transmission Control Protocol was based on the desire to have a more general service that did not contain application-specific functionality and that, as a result, could support a broader range of appli-

cations. Finally, their decision to split TCP and IP placed all application-specific functionality in higher layers at end hosts.

In addition, the choice to offer unreliable datagram service at the Internet layer was motivated by two other design considerations: the designation of the Internet layer as the portability layer and the desire to enable applications to survive partial network failures.

Designating the Internet layer as the portability layer makes it possible to incorporate a wide variety of physical networks into the Internet. As a portability layer, the Internet layer provides an abstract interface independent of the specific characteristics of underlying physical network technologies. To let the Internet Protocol run over a particular physical network, the network only has to be able to transport a packet of reasonable size to any computer attached to it.⁵⁹ Though there should be some level of reliability, perfect reliability is not required. Had the service interface provided by the Internet Protocol made stronger assumptions about the services available from underlying networks, some physical network technologies would not have supported all services. To let the Internet Protocol run over such networks, the missing services would have to be implemented in software to simulate their existence for the Internet layer. This would have complicated the incorporation of such networks. Keeping the Internet Protocol's service interface simple made it possible for a wide variety of physical networks to be incorporated in the Internet without requiring much programming in the Internet layer.⁶⁰

Providing a connectionless, unreliable datagram service at the Internet layer, while placing connection-oriented functionality in transport-layer protocols operating end-to-end between end hosts, makes the Internet more robust.⁶¹ Owing to this division of functionality, information about the state of the connection between the two end hosts (such as the number of packets transmitted or the number of packets acknowledged) is concentrated at the end hosts, while computers in the core of the network do not maintain information about the state of ongoing connections. As a result, even if network components or sub-networks fail, no information essential for the continuation of this connection is lost—the transport-layer-protocol entities can continue to use the connection as long as an appropriate communication path through the network remains. Since all the state information regarding the ongoing connection between two end hosts resides at these two hosts, this state is lost only if one or both of the end hosts involved in the connection are destroyed. Since state and communicating entities live and die together, this feature has been called *fate-sharing*.⁶² In contrast, a connection-oriented packet-switched network

maintains information about specific ongoing connections at packet switches within the network; if one of these packet switches is destroyed, the information about the state of the connections going through this switch is destroyed too, and all connections currently going through this switch fail.

No Hop-by-Hop Error Control in the Internet Layer The Internet's architects also had to decide whether to implement hop-by-hop error control between hosts and routers, and between the routers along the path.⁶³ Although performing hop-by-hop error control in the Internet layer increases the performance of applications that need reliability, it also increases delay (box 3.4). Thus, implementing this function in the Internet layer optimizes the network for the needs of some applications (those that require reliability), but makes it more difficult or even impossible to support applications that do not need reliability but are sensitive to delay (e.g., applications that transmit uncompressed speech or other media streams in real time). Thus, implementing hop-by-hop error control in the Internet layer reduces the range of applications that the network can support. On the basis of this argument—one of the main arguments underlying the broad version—

Box 3.4

The Effect of Hop-by-Hop Error Control

Compared to an architecture that only implements error control and retransmission end-to-end, additional hop-by-hop error control and retransmission at a lower layer may reduce the frequency of end-to-end retransmissions initiated by the end-to-end implementation of the same functionality at a higher layer. Since retransmissions from source to destination across the whole network take longer than retransmissions across a single hop, an additional hop-by-hop implementation may reduce overall delay and improve the performance of applications that need reliability.^a

Hop-by-hop error control, however, increases delay due to the time needed to check integrity and potentially retransmit data at each “hop” along the path. This creates a problem for applications for whom fast transmission is more important than data reliability (e.g., applications that transmit uncompressed speech or other media streams in real time).^b

a. See Saltzer, Reed, and Clark 1984, p. 281; Moors 2002, p. 1216.

b. Saltzer, Reed, and Clark 1984, p. 282, 284–285; Sterbenz and Touch 2001, p. 373.

the Internet's architects decided not to implement hop-by-hop error control at the Internet layer.⁶⁴ Thus, on the basis of the reasoning underlying the broad version, the Internet's architects (implicitly applying the design rule that later was called *the broad version*) decided not to implement an application-specific function in the network.

The End-to-End Arguments and Application Design

Apart from guiding the division of functionality between the Internet and the transport layer, both versions of the end-to-end arguments influence the design of individual applications.

The Narrow Version As we saw above,⁶⁵ the narrow version may require that end-to-end functions be implemented in layers above the transport layer, and ultimately in the application. This is the case if the transport-layer protocols do not operate end-to-end between the original source and the ultimate destination of data, or if events falling into the responsibility of the end-to-end function occur after data leaves the transport layer. In the first case, the function must be implemented at a layer that operates end-to-end between the original source and the ultimate destination of the data. In the second case, it must be implemented at or above the layer at which the event occurs.

The Broad Version The broad version constrains the design of individual applications by requiring application-specific functionality to be implemented in higher layers at the end hosts, not in the core of the network. Within these boundaries, designers are free to choose any application design they like.

Some Misconceptions about the End-to-End Arguments and Their Relationship to the Architecture of the Internet

This section explores some propositions regarding the meaning of the end-to-end arguments and their relationship to the architecture of the Internet which have been advanced in the literature. These propositions have been used to support arguments against using the end-to-end arguments as Internet design principles or policy guidelines.

The propositions in the first group question the role of the end-to-end arguments in the design of the Internet's original architecture. They are usually used in debates about the relevance of the end-to-end arguments to Internet policy. Scholars argue that—because the end-to-end arguments

are no different from the layering principle, or because the original paper describing the end-to-end arguments was published after the original architecture of the Internet was developed, or because e-mail had a certain management structure—the Internet’s original architecture was not based on the end-to-end arguments. If the Internet’s original architecture was not based on the end-to-end arguments, this line of reasoning continues, the end-to-end arguments cannot and should not have any relevance to Internet policy.

The propositions in the second group are often used to argue against the continuing use of the end-to-end arguments as a principle of design or of policy. According to this stance, the end-to-end arguments constrain the development of the Internet’s architecture too much and prevent the network’s core from evolving as it should. For example, researchers advancing this argument assert that the end-to-end arguments rule out architectures that store state in the network, prohibit the provision of Quality of Service in the network, require the network to be simple, or constrain the design of applications by preferring certain application-management structures over others. They then go on to argue that, because the end-to-end arguments interfere with the Internet’s need for state in the network (or Quality of Service, or a complex network), those arguments no longer should guide the Internet’s design.

As we will see, these propositions are based on misconceptions. Though this does not imply that the end-to-end arguments are still the appropriate design principles for the Internet, or that they should guide Internet policy, it does imply that these propositions cannot be used to support the arguments against their use outlined above.

End-to-End Arguments and Layering

Some argue that the broad version of the end-to-end arguments directly flows from the layering principle.⁶⁶ This perception is not correct.⁶⁷

The layering principle usually results in a system in which lower layers provide more abstract services than higher layers.⁶⁸ Thus, both design principles tend to create systems in which lower layers are more abstract or general than higher layers. The broad version of the end-to-end arguments, however, establishes much more restrictive rules for placing functions in lower layers in the network than the layering principle. A system based on the layering principle must comply with the constraints imposed by that principle—i.e., lower-layer protocols are not allowed to use higher layers, and so on. Beyond these constraints, system architects are free to place functions as they see fit.⁶⁹ In contrast, in a system following the broad

version, “a function or service should be carried out within a network layer only if it is needed by all clients of that layer, and it can be completely implemented in that layer.”⁷⁰

Thus, a layered network architecture that complies with all the constraints imposed by the layering principle, but implements application-specific functions in the network, complies with the layering principle, but violates the broad version of the end-to-end arguments. The Open Systems Interconnection architecture’s X.25 protocol suite is an example of such an architecture.⁷¹ It implements error control and error handling at all lower layers up to the transport layer. As we saw above, performing error control in the Internet layer violates the broad version. Thus, this architecture respects the layering principle, but contradicts the broad version of the end-to-end arguments. Similarly, the initial proposals for an Inter-network Transmission Control Protocol that provided reliable connection-oriented data transfer at the Internet layer complied with the layering principle, but violated the broad version.⁷²

If applying the layering principle can result in networks that violate the broad version, the broad version cannot flow from, and cannot be an automatic consequence of, the layering principle.

End-to-End Arguments and Fate Sharing

Some researchers assume that the end-to-end arguments require the network architecture to comply with the principle of “fate sharing” (explained above).⁷³ Thus, they assume that the end-to-end arguments constrain the amount of state in the network. This view can be traced back to “Request for Comments 1958,” a document published by the Internet Engineering Task Force that described the architectural principles of the Internet (Carpenter 1996). After quoting the narrow version of the end-to-end arguments from the original paper, the text continues: “. . . this principle has important consequences if we require applications to survive partial network failures. An end-to-end protocol design should not rely on the maintenance of state (i.e. information about the state of the end-to-end communication) inside the network. Such state should be maintained only in the endpoints, in such a way that the state can only be destroyed when the endpoint itself breaks (known as fate-sharing).”⁷⁴ Neither version of the end-to-end arguments directly establishes rules for state in the network. Whereas the broad version emphasizes evolvability, the narrow version focuses on correctness. In contrast, fate sharing aims at survivability in case of partial network failures. As the original architecture of the Internet shows, there are situations in which all three principles are usefully applied.

This need not always be the case. As a result, it seems more useful to distinguish clearly between the different rules instead of conflating them in a single design principle.⁷⁵

End-to-End Arguments and Quality of Service

Some researchers assume that the broad version of the end-to-end arguments rules out the provision of Quality of Service in the network.⁷⁶ This interpretation of the end-to-end arguments is too strong.⁷⁷

The current version of the Internet Protocol provides the same service to all data packets. The Internet Protocol does its best to deliver a packet to its destination, but does not guarantee a particular bandwidth or a particular delay. If a network provides “Quality of Service,” the network offers different types of service. For example, it may guarantee a minimum bandwidth or a maximum delay, or it may give some traffic priority over other traffic without giving absolute guarantees.⁷⁸ Providing Quality of Service consists of two tasks: specifying which packets get which level of service quality, and providing packets with the service quality specified. The broad version of the end-to-end arguments constrains the two tasks to different degrees.⁷⁹

Although it is possible to build a service that offers reliable data transfer at a higher layer and operates only on end hosts on the basis of an unreliable datagram service at the Internet layer, it is not possible to build a service that offers guaranteed bandwidth or delay at a higher layer if the Internet layer does not guarantee bandwidth or delay.⁸⁰ Thus, providing guaranteed bandwidth or delay is a function that cannot be implemented at the end hosts only. Since the broad version does not prevent functions from being implemented in the network if they cannot be completely and correctly implemented at the end hosts only,⁸¹ it does not prohibit the implementation of functions in the network that are necessary to guarantee bandwidth and delay.

The broad version may constrain which entity determines which packet gets which service quality. The principle of application autonomy described above suggests that applications (or ultimately the users) should determine which type of service they need; this does not imply that they should get this service for free. Imagine that the Internet layer offered different services with different bandwidth and delay characteristics to higher layers at the end hosts. The higher layers would choose the type of service they desired and would communicate that choice to the Internet layer through the Internet layer’s service interface (e.g., by setting a type-of-service field). Such an architecture would not require application-specific

functionality to be implemented in the network's core beyond what is necessary to implement functions that cannot be implemented by the end hosts alone. Thus, it would not violate the broad version of the end-to-end arguments.

In contrast, a Quality of Service architecture in which the network decides who gets what service quality runs counter to the principle of application autonomy that underlies the broad version. In particular, a Quality of Service architecture in which entities operating at the Internet layer or below make this decision on the basis of information obtained by accessing or modifying messages passed to them by higher-layer protocols, or by mechanisms that implement application-specific functionality in the network's core, would directly violate the layering principle and/or the broad version of the end-to-end arguments. For example, using deep packet inspection to "look into" a data packet and determine to which application it belongs and then slowing down packets that belong to specific applications or classes of applications (e.g., slowing down packets belonging to BitTorrent in particular or to peer-to-peer file-sharing applications in general)⁸² would violate the layering principle and the broad version of the end-to-end arguments.⁸³

Thus, the broad version of the end-to-end arguments constrains, but does not rule out, the provision of Quality of Service in the network.

End-to-End Arguments and the Stupid Network

Some scholars assume that the end-to-end arguments require the network's core to be "stupid"—that is, to contain only simple functionality.⁸⁴ The term "stupid network" was coined by David Isenberg to describe the original Internet (box 3.5). Though Isenberg's arguments in favor of stupid networks are similar to those underlying the broad version of the end-to-end arguments, not all networks based on the broad version are "stupid." The broad version of the end-to-end arguments does not require simplicity; it requires only that network functions be general. The broad version of the end-to-end arguments advises that the network should not contain application-specific functionality and should provide only general services useful for a large variety of applications. The argument does not restrict how the network realizes these services.⁸⁵

End-to-End Arguments and Management Structure of Applications

Some researchers assume that the end-to-end arguments prescribe a specific structure for applications. For example, some researchers assume that distributed applications with application-level intermediaries that are part of

Box 3.5

Stupid Networks

The term “stupid network” was coined by David Isenberg in an article in which he compared the “stupid network” design that characterizes the Internet to the “intelligent network” model, on which conventional telecommunications networks are based.^a Isenberg’s arguments in favor of stupid networks are very similar to those underlying the broad version of the end-to-end arguments: contrary to the telephone network, which is optimized for a single application (telephony), a stupid network is “underspecified” (that is, not optimized in favor of specific applications). This makes it very flexible with respect to new applications.^b A stupid network only provides “dumb transport,” while intelligence is concentrated at the end hosts, under the control of the users. This design shifts control over applications and application-level innovation to users.^c According to Isenberg, a stupid network is characterized by three features: it has “nothing but dumb transport in the middle, and intelligent user-controlled endpoints,”^d “the center of the network is based on plentiful infrastructure—cheap bandwidth and switching”—not on scarcity,^e and “transport is guided by the needs of the data, not the design assumptions of the network.”^f

- a. Isenberg 1997.
- b. Isenberg 1998, p. 28.
- c. *Ibid.*, pp. 28–29.
- d. Isenberg 1997, p. 16.
- e. Isenberg 1998, p. 26.
- f. Isenberg 1997, pp. 16, 18.

the application but that are owned and operated by a network or application service provider and that topologically are in the network somehow contradict the end-to-end arguments (figure 3.7). For example, it has been argued that, because e-mail has such a structure (box 3.6), the Internet never was wholly end-to-end.⁸⁶ Based on a similar reasoning, some may think that Web-based applications or cloud computing violate the end-to-end arguments.

This view arises from an interpretation of the terms “end hosts” and “in the network” that differs from the interpretation that underlies the end-to-end arguments. Instead of understanding the two categories as indicating a computer’s functional relationship to the network as a user or provider of communication services, these researchers interpret them in the context of network topology or in the context of administrative ownership and

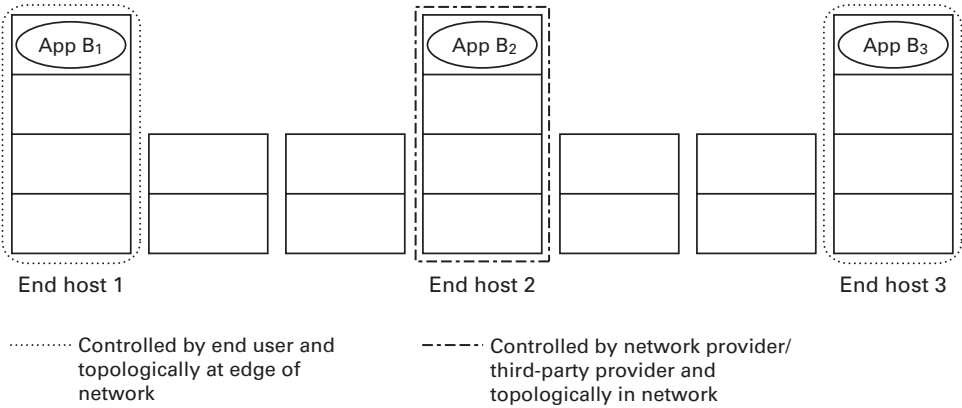


Figure 3.7
Management structure of applications.

Box 3.6
The Structure of E-Mail

In general, the e-mail program running on the sending user’s computer transfers the e-mail to an intermediary mail server (i.e., an application program which is typically running on a computer that is topologically in the network and operated by the user’s Internet service provider). The intermediary mail server then transfers the e-mail to the receiving user’s mail server, typically also running on a computer that is topologically in the network and is operated by the receiver’s Internet service provider. Finally, the e-mail program running on the receiving user’s computer is used to get the e-mail from the receiver’s mail server. Thus, the structure of the overall application consists of one part running on a computer that is topologically at the edge of the network and operated by an end user, two parts running on computers that are topologically in the network and operated by a third party, and one part running on a computer that is topologically at the edge of the network and operated by an end user.^a Some argue that this structure violates the broad version of the end-to-end arguments, because the two intermediary parts of the application are running on computers topologically in the network and operated by third parties.

a. For a description of the protocols commonly used to send and receive e-mail in the Internet, see Kurose and Ross 2008, pp. 116–130. The topological and management structure for e-mail described above is not the only possible structure. For an overview of possible management structures, see Gaynor 2003, pp. 54–60.

control. Thus, they assume that an “end host” is an end system that is topologically at the edge of the network (i.e., co-located with subscriber end systems) and owned and controlled by an end user; a computer “in the network” is topologically in the network (i.e., co-located with network nodes) and owned and operated by a network provider or by a third-party service provider.⁸⁷

Though differences in location, in ownership, or in control may be relevant from a policy perspective and may also have implications for application innovation,⁸⁸ they do not matter for the broad version of the end-to-end arguments. As explained above, the broad version describes how functionality is distributed among layers in a layered network architecture and does not cover topological or administrative relationships.⁸⁹ In the context of the broad version, the distinction between end hosts and computers in the core of the network is a purely functional distinction between users and providers of network communication services. End hosts are computers that run applications that use the network; computers in the core of the network provide services to the computers using the network. In the original architecture of the Internet, this distinction maps to the layers in the protocol stack: layers up to the Internet layer are considered to be in the network, layers above the Internet layer are not. Thus, any application implemented in layers above the Internet layer at end hosts is compliant with the broad version of the end-to-end arguments; who controls the different parts of the application and where they are located is not relevant to this design principle.

Similarly, some assume that the broad version of the end-to-end arguments favors peer-to-peer applications over client-server applications. Peer-to-peer applications are often run by “normal” end users on their end systems, whereas the server component of a client-server application is often owned and operated by a network provider or by a third-party service provider and co-located with computers belonging to the network’s core.

Again, these differences are not relevant to the broad version. As long as distributed parts of a peer-to-peer application and the client and server components of a client-server application are implemented in layers above the Internet layer at end hosts, the two application architectures are equally compliant with the broad version.

End-to-End Arguments and the History of the Internet

Some claim that the end-to-end arguments cannot have influenced the design of the Internet’s original architecture, insofar as the papers that named, identified, and described them as design principles were published

in 1981 (narrow version) and 1998 (broad version),⁹⁰ after the Internet's original architecture had been created.⁹¹ This view neglects the fact that the design rules and accompanying rationales that later came to be known as the "end-to-end arguments" had been used for many years before the papers formally describing them were published. The 1984 paper (the revised journal version of the 1981 paper that is usually cited when scholars refer to the end-to-end arguments) does not claim to invent a new design principle from scratch; rather, it aims to formally articulate and analyze a design principle that had already been used in a variety of areas.⁹² The 1998 paper aims to evaluate whether a new approach to networking ("active networking") complies with the end-to-end arguments (an existing design principle). To do so, it describes the (broad version of the) end-to-end arguments, discussing the trade-offs and the rationale associated with this design principle and its history. Both papers explicitly acknowledge earlier work—by the authors or other researchers—that made an end-to-end argument of the type described in the paper in a specific context and thereby provided the basis for the paper's more general articulation of this design principle.⁹³ For example, the paper describing the narrow version points to earlier papers making an end-to-end argument in the context of encryption, and to work making an end-to-end argument in the context of reliable file backup. The paper describing the broad version points to the authors' own work on local-area networks and secure operating system kernels, and to research regarding the reduced-instruction-set computer (RISC) architecture, as examples from which the more general principle emerged. Both papers—and other papers by Saltzer, Reed, and Clark—explicitly mention the design of communication networks and the design of the Internet as areas in which end-to-end arguments had been applied.⁹⁴ For example, the 1984 paper discusses the decisions on where to place functions such as reliable data transfer, duplicate suppression, or reordering as typical examples of decisions in which end-to-end arguments are applied, and frames the debate over "datagrams, virtual circuits, and connectionless protocols" as a debate about end-to-end arguments.⁹⁵ The 1998 paper points to the authors' work on "end-to-end transport protocols in LANs and the Internet Experiment" as work from which the more general design principle emerged, and discusses the decision not to optimize the Internet for "telephony-style virtual circuits" as an example of the application of the (broad version of the) end-to-end arguments that has been validated by history.⁹⁶

In sum, the papers describing the two versions articulate design rules that had been used in various contexts before they were formally

recognized as a design principle. The design decisions made during the development of the Internet's original architecture were important (but not the only) examples of earlier applications of the principles from which the general description of the principles emerged. Thus, although the papers formally describing the end-to-end arguments as design principles were published after the Internet-related design decisions ascribed to them were made, the underlying design rules shaped the Internet's original architecture.

As was explained in chapters 2 and 3, the original architecture of the Internet was based on a variant of the layering principle called *relaxed layering with a portability layer* and on the end-to-end arguments. Layering, in turn, is a special form of modularity. There are two versions of the end-to-end arguments, the narrow version and the broad version. Although both versions influenced the Internet's original architecture, only the broad version affects the economic environment for innovation. Therefore, parts III and IV of this book will focus on modularity, relaxed layering with a portability layer, and the broad version.

III Architectural Constraints on Innovation

4 Architecture and the Cost of Innovation

To evaluate how the Internet's architecture affects innovation, we must understand exactly how architecture constrains economic actors. How do design principles and the architectures they shape influence the costs and benefits associated with a given innovation? How do they affect who can design and build an innovation? Answering these questions is the goal of part III. Chapters 4–6 each focus on a particular aspect of the relationship between architecture and innovation and explore it in detail. The resulting insights are then applied to assess how the Internet's original architecture influences innovation.

This chapter examines how architecture affects the costs of realizing innovations (that is, the costs of developing and producing the first complete version of the system that contains the new or improved good or service). This book focuses on innovation that can be influenced by architectural features—developing components for a new system, improving existing components, developing new components for an existing system, changing an architecture to enable new functionality, or creating new architectures. Thus, in the context of this book, innovating involves changing components of a system or its underlying architecture.

Architectures differ in the kinds of changes that are needed to realize a particular innovation: The architecture of a system describes how the system is decomposed into components and how these components cooperate to provide the overall functionality of the system. The distribution of functionality among components affects which components need to be altered to realize the innovation. The nature and strength of dependencies among components determines if these changes require subsequent changes to other parts of the system that depend on the changed functionality. Thus, the architecture of a system affects how many and what kinds of components must be altered to realize a specific innovation, which in turn affects the costs of making these changes (figure 4.1). Changing

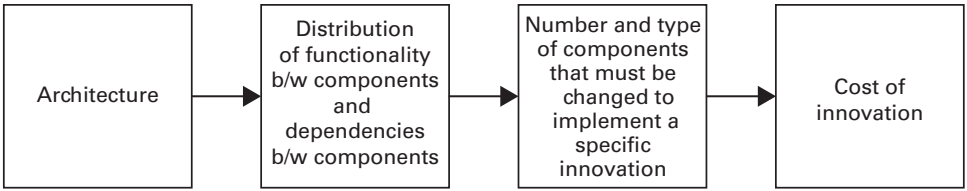


Figure 4.1
How architecture affects the costs of innovation.

many components of an architecture is more difficult and costly than changing only one, even if the total amount of change¹ is the same (box 4.1). And although such costs are decreasing continually, changing a system’s hardware components is usually more costly than changing its software components, because of the higher costs associated with prototyping, testing, and manufacturing hardware components.²

While an architecture implies what kinds of changes are necessary to realize a particular innovation, the actual costs to an innovator of making these changes may differ among innovators. For example, the costs of having a programmer revise the code for a program may differ depending on whether the revisions are being done by an established firm, by a start-up company, or by a freelance programmer. Each innovator will incur costs—e.g., an established firm and a start-up company may have to hire additional programmers, and a freelance programmer may have to decline another project—but the actual costs may differ because of different salary structures or different amounts of overhead among firms, and their effects may be assessed and felt differently. Similarly, an innovator who owns all the components involved in a change will be able to implement the change at lower costs than an innovator who owns only one of these components (box 4.1). Thus, although the activities causing the costs of change depend on the architecture, the actual costs are relative to the actors and the context in which they are operating.

Because of architectural differences, the number and the kinds of components that need to be changed to realize a particular innovation may differ among architectures. As a result, realizing that innovation may be more costly in one architecture than in another. These differences in cost affect the economic environment for innovation in several ways:

First, the costs of change determine the amount and the kind of investment needed to realize a certain innovation, effectively selecting which economic actor is able to realize the innovation. Innovation that involves prototyping, testing, and ultimately producing physical products requires

Box 4.1

Number of Affected Components and Costs of Change

In general, the costs of changing an architecture rise with the number of architectural components involved in the change.^a The effort involved in understanding affected components and then designing, implementing, and testing changes rises with the number of components that must be changed; e.g., making five small changes to one component is usually less costly than making one simple alteration in five different components.^b This is because realizing a change entails more than simply changing code. For example, in large software systems, alterations involving several components must be approved by change control boards, the documentation of affected components must be changed, and the testing of new components and their introduction into the system must be a coordinated across development teams. The higher the number of affected components, the more these development-process costs dominate the economics of a proposed change.

The increase in costs that results from an increase in the number of affected components is particularly large if the various components are owned and maintained by different firms. As we will see in chapter 5, coordinating changes across firm boundaries is considerably more difficult and costly than coordinating changes within a single firm. In addition, one (or more) of the firms involved may use the opportunity to behave strategically in order to improve its (or their) competitive position at the expense of other organizations. For example, a firm may try to “hold up” another (a tactic that will be described in chapter 5).^c

a. In addition, the costs of change generally rise with the size and complexity of code that must be modified or produced to implement the change. Thus, predicting the real costs of implementing a change requires an estimate of the number of lines of code that are affected. These estimates can then be multiplied by productivity factors derived from empirical studies to yield an estimate of the time required to change the code. These factors vary depending on the kind of programming involved; for example, studies have shown that the productivity of writing new code is higher than that of changing existing code. (See Bosch and Bengtsson 2001, p. 171.) In addition, changing code you have written yourself is easier than changing someone else’s code.

b. This is a statement of a general relationship that may not hold in some cases. For example, changing one very complex component may be more difficult than changing several simple ones. Implementing a very simple change across a number of components may be easier than realizing a complex local change. (See Bass, Clements, and Kazman 1998, p. 82.)

c. For an analysis of these and additional factors that may increase the costs of innovation in such a situation, see the discussion in chapter 5 below.

a different type of investment than software innovation. For example, a lone inventor working in a garage may be able to prototype a new piece of hardware but probably is not able to complete the subsequent evaluation, manufacturing, and distribution necessary to commercialize the innovation in the same way a single software programmer may be able to realize and distribute a new piece of code. The effect of the costs of change on the type of economic actors who can realize innovations will be discussed more in the next chapter.

Second, the costs of realizing an innovation establish the minimum benefit that the innovation must provide in order to justify innovating. Different innovators may do their work for different reasons and may evaluate opportunities for innovation in different ways³—expressed more formally, they may have different preferences, and act under different decision rules—but we can assume that they will not innovate unless the expected benefits at least equal the expected costs.⁴ If the costs of realizing a particular innovation differ between two architectures, the difference between expected benefits and expected costs may lead a particular innovator to innovate in one architecture but not in the other.

Finally, the costs of change implied by an architecture determine whether isolated changes to system components can be made at low costs, allowing innovators to improve pieces of a system without having to change the whole system. As we will see, the ability to experiment at the component level—trying out new functions, novel ways of implementing functions, and new relationships among components—may result in a system that is of higher quality than a system that does not enable this type of experimentation. A system that allows experimentation at the component level will evolve more rapidly than a system that can only be replaced as a whole. The value of being able to experiment at the component level increases with the uncertainty about the best way to innovate, since innovators can take risks on components without compromising the quality of the whole system.

Thus, differences in the costs of innovation among architectures may result in a different *number* of innovations being realized and in different *types of innovators* realizing them.

Costs of Change in Modular and Integrated Architectures

The overall costs of innovation consist of three parts: the costs of realizing an innovation, the costs of production and distribution, and the costs of deployment. The costs of realizing an innovation are the costs of

developing and producing the first complete version of the system that contains the new or improved good or service. They are incurred by the innovator and, if the rest of the system has to be changed in order to integrate the new component, by the providers of complementary components.⁵ The costs of deployment are the costs of making changes to the operational system that are necessary before the innovation can be used in that system for the first time. This section will focus on how modular and integrated architectures affect the costs of realizing an innovation.

The costs of realizing an innovation can be separated into two categories. First, there are the costs associated with developing and testing the innovation itself; these are the costs of designing and developing the necessary changes to the components directly affected by the innovation. I will call these the *core costs of innovation*. Second, there are the costs associated with developing and testing the changes necessary in the rest of the system in order to enable the new or improved components to function properly. I will call these the *costs of system adaptation*. For example, if Microsoft wants to change an existing application programming interface to its Windows operating system, the core costs of this innovation are the investments it must make in designing and implementing the new version of the application programming interface and the code within the operating system that implements the functionality provided by the application programming interface. If the new version completely replaces the old version of the interface, applications that relied on the old version will not be able to function properly. The costs incurred in changing the applications so that they work with the new version of the interface are the costs of system adaptation.

Architectural Differences and Costs of Innovation

The architectural differences between integrated and modular architectures result in different costs of innovation. In an integrated architecture, it usually isn't possible to make changes to a component that do not trigger subsequent changes in the rest of the system. Owing to the highly interdependent nature of the design, it is very likely that changes in an integrated architecture will affect at least one design parameter that is linked to design choices in other components. Changing such a parameter may start a chain of changes in other components. The cascade of changes resulting from a single change to an interdependent design parameter is often called the *ripple effect*. Because the complexity of the design makes the implications of changes difficult to understand or predict, it is not easy to identify which parts of the system will be affected and exactly how they

will be affected. Often various parts of a system have to be redesigned, implemented, and tested; in the worst case, all of a system's components have to be changed.⁶

Suppose the designers of the iMac are thinking of introducing a new screen technology into the next version of the computer. The new screen technology may require more space than its predecessor, and it may produce more heat. Owing to the change in the dimensions of the screen, components that were located adjacent to the screen may no longer fit. Other components may be negatively affected by the heat, and trying to move them elsewhere may require adjustments in the rest of the system.

Thus, even small changes to an integrated architecture usually require changing larger parts of the system. Implementing these changes is difficult, time-consuming, and costly; thus, the costs of system adaptation are high.

Compared to this integrated approach to design, modularity reduces the number of cross-component interdependencies and the level of coupling between components. As a result, a large number of changes can be implemented through autonomous changes—that is, changes that affect only one component and do not require altering the rest of the system.

Owing to the strict partitioning between visible and hidden information, modules can depend only on other modules' visible information. As a result, all changes that do not affect the architecture's existing visible information are autonomous changes that do not affect the rest of the system, resulting in zero costs of system adaptation. Changes in this category include replacing a component with a better implementation without changing its interfaces, adding a new interface to an existing component, and adding a new component.⁷ For example, in the modular architecture of the PC, introducing monitors with a new screen technology does not require any changes in the rest of the system as long as the interface between the monitor and the rest of the system stays the same. Similarly, adding a new application programming interface to the Windows operating system without altering the existing application programming interfaces does not require any changes to existing applications, since they can continue to use the existing application programming interfaces.

As we will see later in this chapter, innovations that can be realized through autonomous changes are especially attractive targets for innovative efforts. Since autonomous changes do not incur any costs of system adaptation, the threshold for innovation is considerably lower than for systemic changes, making it more likely that the innovation will be real-

ized. As the next chapter will show, autonomous changes can often be realized independently by actors other than the system architect. They do not require cooperation from the system architect or from other component makers; thus, they free the innovator from the need to convince others of the usefulness of his innovation.⁸

As soon as a change involves altering an architecture's existing visible information, adaptation costs are incurred. These costs depend on the number of system components that directly or indirectly depend on the changed component. This number, in turn, depends on the type and density of interactions among components—i.e., the dependencies and strength of coupling among them—which subsequently depend on the distribution of functionality among components determined by the architecture. For example, the costs of system adaptation associated with changing an existing Windows application programming interface depend on the number of applications that directly or indirectly rely on this interface.

Owing to the costs of system adaptation, innovations that require changes to an architecture's existing visible information are considerably more costly to realize than innovations that can be realized through autonomous changes.⁹ Because the costs of system adaptation quickly subsume the benefits of innovation, such innovations must provide considerably higher benefits if they are to be realized. Thus, large costs of system adaptation can quickly neutralize potential gains from innovation, raising the threshold for innovation.

Owing to the different levels of coupling between components, the costs of realizing a specific innovation probably will be greater in an integrated architecture than in a modular architecture. In an integrated architecture, the innovation will almost always require changing large parts of the system. In contrast, in a modular architecture many innovations may be realized through autonomous changes at much lower costs. Owing to these cost differences, it is likely that more innovations will be justified under a modular architecture. For example, the benefits associated with the new screen technology mentioned earlier may be larger than the costs of designing, building, and testing a monitor with the new technology in the modular architecture of the PC, but may not be sufficient to justify a complete redesign of the integrated architecture of the iMac.¹⁰ The difference in costs may also have consequences for the types of innovators that are able to innovate within a certain architecture: smaller firms may be able to muster the resources necessary to improve one component but may not have the resources necessary to develop a complete system.

Options Theory and the Value of Flexibility under Uncertainty

Architectures affect innovation costs by influencing how functionality is distributed among components and how components depend on one another. If innovators can change parts of the architecture at low cost, they are better and more easily able to react to future developments. This flexibility is particularly valuable if there is uncertainty about the best technical approaches to innovation, or about the features that consumers may value. The theory of real options provides a framework for reasoning about the relationship between the costs of innovation implied by an architecture and the value of experimentation under uncertainty. In particular, it lets us understand how architectures can leverage the risks surrounding innovation and shape the potential rewards from innovating.

Innovating on an architecture or its components can be understood as acquiring a real option. Since modular and integrated architectures differ in how much it costs to change them, they also differ in the types of options they provide. Similarly, alternative modular architectures may differ in how much it costs to change a particular module, which may affect the option value of experimenting on that module. These differences in option value, in turn, indicate how much effort innovators will invest in improving that module under the different modular architectures.

Options theory provides a method of analysis that recognizes and quantifies the value of being able to flexibly choose alternative courses of action in the future under conditions of uncertainty. The framework originated in finance, where it is used to value financial options—for example, options on stocks or bonds.¹¹ Real-options theory extends the framework to options on non-financial (real) assets.¹²

An option is a right, but not an obligation, to take action in the future. For example, a financial option may provide the right to buy a specific stock at a specific price (called the *exercise price*) at some specified time in the future (the *expiration date*). (This is an example of a “European call option.”) If the value of the stock is higher on the expiration date than the exercise price, the owner can exercise the option and buy the stock for the exercise price. His payoff will be the difference between the value of the stock at the expiration date and the exercise price. If the value of the stock is lower than the exercise price, the owner does nothing, since he had a right but not an obligation to buy the stock. In this case, his payoff is zero. Thus, the owner of the option can limit his loss in the case of an unfavorable outcome, but can profit from any positive developments.¹³

Thus, an option is a way to deal with uncertainty. It provides an opportunity to delay deciding about an investment whose future value is not

known until the uncertainty is resolved. If the future value is negative, the owner of the option does nothing; in this case, his payoff is zero. If the future value is positive, the owner exercises the option. His payoff is the positive future value minus the exercise price. Thus, an option gives its owner the possibility to capture potential future gains while avoiding potential future losses.¹⁴

The value of an option increases with uncertainty. If uncertainty is high, then very high and very low future values are more likely. Direct investment in the underlying asset would be very risky: there is a chance of high gains, but there also is a chance of high losses. For the owner of an option, the increase in uncertainty has a one-sided effect: there is an increase in the upside potential (the chance of a very high payoff), but there is no increase in potential losses.¹⁵

Since options can limit risks, it is not surprising that they are valuable. Acquiring an option requires paying a price. The difference between the present value of the option (the equivalent of the future value of the option in terms of wealth today) and its price denotes the net present value, or net option value, of an option. If the net option value is negative, the benefits associated with the ability to limit risks are smaller than the costs of acquiring the option. If the net option value is positive, the benefits are larger than the costs.

Since it is not clear how an option's value will evolve, we need a valuation methodology to determine the present value of the option—a way to translate a range of potential outcomes in the future into a single value in the present. Over the past decades, scholars of finance (including Fischer Black, Myron Scholes, and Robert C. Merton) have developed ways to solve this problem.¹⁶ These methodologies are used to determine the value of options in the financial markets.

Real Options in Modular and Integrated Architectures

The theory of real options extends the theory of financial options to options on real assets.¹⁷ In contrast with financial options, real options are not bought; they arise from investment decisions afflicted with uncertainty that open the opportunity to choose alternative courses of action in the future when the uncertainty is resolved. If the outcome is bad, the firm chooses an action that limits its losses. If the outcome is good, the firm can leverage the situation.

An investment decision can be understood in the context of real-options theory if (1) the decision depends on an uncertain outcome, (2) the decision can be made in a way that opens the opportunity to choose

alternative courses of action in the future when the uncertainty is resolved, and (3) these alternative courses of action let the firm either truncate its losses in case of an unfavorable outcome or profit from favorable developments.¹⁸ If these conditions hold, the real option is the opportunity to make the decision in the future, and the firm acquires this option by making the investment decision now. Thus, the “price” of the real option is the cost associated with the current investment. In the real-options framework, the decision about the current investment becomes a decision to acquire an option: although the outcome of the decision depends on characteristics of the investor (for example, a value-maximizing investor who can take on as many projects as he likes would acquire the option by making the investment decision, if the present value of the option exceeds its price, or, in other words, if the net option value is positive),¹⁹ no investor would buy an option if its net option value were negative. Over the past years, a number of methods have been developed that can be used to calculate the value of a real option.²⁰

Real options are much harder to detect than financial options. Because real options are hidden in the investments of a firm, the first step of a real-options analysis is to identify and define the real options associated with investment decisions.²¹

A firm’s choice to innovate on components of an architecture is such an investment decision. In a sense, when we understand particular types of architectures—e.g., integrated versus modular—in relation to a firm’s real options, we may better understand how decisions to innovate in uncertain environments depend on the architecture of the systems being innovated on and the costs associated with doing so. Innovation on existing integrated and modular architectures can thus be understood from a real-options perspective.²²

First, the decision to innovate is afflicted with uncertainty. It depends on the expected costs and benefits of the innovation. Owing to the inherent complexity of the designs of large systems, it is often impossible to predict how valuable changes to an architecture will be.²³ It may be difficult, for example, to predict which technical approach will provide the best solution to a specific problem (technical uncertainty) and whether and how consumers will value the resulting product (business uncertainty). As a result, the expected value of the innovation is uncertain at the time of the decision. It may be worth more or less than the existing design. Often, the value of the innovation can be determined after it is complete. In particular, this applies to technical uncertainty, insofar as the technical quality of an innovation can be determined once it has been realized.

Although the use of focus groups or customer prototype testing may reduce business uncertainty once an innovation is realized,²⁴ often only the market can resolve business uncertainty.

Second, innovating on an existing architecture provides the right, but not the obligation, to use or sell the result of the innovative process instead of the existing product. Thus, innovating opens up alternative courses of actions when the uncertainty is resolved.

Third, these alternative courses of action let the firm either truncate its losses in case of an unfavorable outcome or profit from favorable developments. If the result is better than the existing system, the innovator can use the new system and reap the rewards of her innovation. If the result is worse than the existing system, she can continue using or selling the existing product without having to suffer profit losses or produce an inferior product. The innovator can minimize her losses by maintaining the status quo.²⁵

Thus, the decision to innovate opens the real option to decide whether or not to use or sell an innovation in the future when its value is known. The cost of acquiring this option is the cost of developing the innovation. An innovator will not innovate if the net option value is negative—that is, if the present value of the innovation is smaller than the cost of acquiring the option.²⁶ To determine the present value of the innovation, one needs a methodology that translates the uncertain future value of an innovation into a known value today.²⁷

Though innovators who innovate on modular or integrated architectures all acquire a real option to use or sell the innovation in the future, they acquire different types of options depending on the architecture. Innovating on an integrated architecture provides an option on a portfolio, whereas innovating on a modular architecture provides a portfolio of options.²⁸

In an integrated architecture, it is usually not possible to make changes to a component that do not trigger changes in the rest of the system. As a result, isolated innovation at the level of individual components is rarely feasible. Improving one component requires that large parts of the system be redesigned and tested. Often it is more efficient to redesign an integrated system from scratch than to try to improve the existing system. Thus, innovation on an integrated architecture usually takes place at the system level, providing the option to replace one system with another. Because the new system consists of a fixed set of components, the innovator acquires an option on a portfolio of components.²⁹

In contrast, a modular architecture enables innovation at the module level.³⁰ These changes do not affect the rest of the system as long as the

architecture's existing visible information stays the same. As a result, it is possible to improve or change individual components while retaining the rest of the system. For example, it is possible to replace one module with a better version, to add components that offer new functionality using the services of existing modules, or to remove modules that are not used by other modules—all without changing the architecture's existing visible information. Existing modules can be split or merged without any changes to the architecture, or with only minor changes, depending on the system.³¹

Innovating on a specific module provides the option of replacing the existing version of the module with the result of the innovation.³² If all modules are changed, the innovator acquires corresponding options for all new modules. Instead of getting one option on a complete system (as with an integrated architecture), she acquires as many options as there are modules in the system. As a result, she can choose with respect to each module whether the new version is better than the old one, and can act accordingly. Thus, a modular architecture provides a portfolio of options.³³

Under very general assumptions (the result has been proved for general probability distributions for any distribution of underlying value, assuming conservation of aggregate value), it can be shown that a "portfolio of options" is more valuable than an "option on a portfolio."³⁴ This is so because a portfolio of options exploits the lack of correlation in the development of the value of the underlying assets. If the value of the underlying assets is positively correlated, the value of the assets increases and decreases together. If the value of one asset increases, the value of the other assets increases too. Often the value of the underlying assets is not positively correlated. While the value of some assets may rise, the value of others may fall. If one owns an option on a portfolio, one can get the portfolio only as a whole. Even if the option has a positive payoff, increases in the value of some assets may have been neutralized by decreases in the value of other assets. In contrast, the owner of a portfolio of options has separate options on any of the underlying assets. For each of these options, he can exercise the option if the payoff is positive, and can do nothing otherwise. Because his gains will not be diluted by the losses of other assets in the portfolio, he is better off than the owner of the option on a portfolio.

Therefore, from an options perspective, a modular architecture is more valuable than an integrated architecture if the underlying distribution of aggregate value is the same.³⁵ In general, the results of experiments

on different modules will not be correlated.³⁶ While the value of some components rises, the value of others may fall. In a modular architecture, innovators can use any module innovation that is better than the existing module, and can reject the others. In an integrated architecture, an innovator has to accept or reject a complete system, improvements in some parts of which may be offset by deteriorations in others. As a result, the expected value of the resulting system is higher in the modular architecture.

The difference in option value between modular and integrated architectures also affects how innovators react to the risk of failure associated with uncertainty.³⁷ In a modular architecture, innovators can experiment on a module without affecting the quality of the whole system. If the innovation fails, they can limit their losses by continuing to use the existing module. Because a modular architecture enables innovators to capture the potentially large benefits associated with risky projects and avoid the downsides, risky projects become very attractive. In an integrated architecture, the failure of a risky approach to one component may compromise the quality of the whole system, owing to the interconnected nature of the design. As a result, innovators in integrated architectures will try to avoid risky approaches so as not to endanger the success of the next generation of the integrated design.

The option to replace an existing module with a better version is not the only option provided by modular architectures. A modular architecture contains a much more complex set of options.³⁸ In addition to options at the module level, a modular architecture provides options at the system level. For example, with a modular architecture it is possible to first design and produce a minimal system at a lower cost than would be incurred designing and producing a complete system. Instead of requiring innovators to bear all development costs initially, this approach gives them the chance to save a part of the development costs if, for example, demand for the system is too low or the technical challenges become too great. If the system sells well, or if the technical hurdles are overcome, new modules can be added later. If the system does not sell well enough, or if the technical barriers are too great, no more modules are developed. Thus, by developing a minimal system, designers gain the option to proceed with further system design in the future.³⁹ In the terminology of options theory, a minimal system provides an "option on a portfolio of options."⁴⁰ Though it is important to recognize the complex structure of options embedded in modular architectures, it is almost impossible to calculate the value of these more complex options.⁴¹

The decision to innovate with respect to a specific module, however, depends on the option value of that module, not on the option value of the architecture.

Real Options in Different Modular Architectures

In the preceding subsection we focused on the differences in option value between modular and integrated architectures. In this subsection we will examine more closely the differences among modular architectures and the option values they provide. For a specific module, alternative modular architectures may imply different costs of innovation. Consider the designers of an operating system. Should the functionality needed to communicate with the printer be implemented in hardware, or in software? Because hardware is more difficult and costly to change than software, the core costs of innovation will be much lower if the functionality is implemented in software. Should applications be allowed to interact with the printer directly, or should they only be allowed to interact with the printer through the application programming interfaces provided by the operating system? Relative to the first option, allowing applications to access the printer functionality only through application programming interfaces constrains the dependencies within the system, resulting in lower costs of system adaptation if the printer functionality should change. These architectural decisions affect the option value associated with innovating on a component of a particular architecture by affecting the costs of innovating on a given module. How do these differences affect the amount of effort that innovators will invest in improving the module and the rate of change of the module?

Until now, we have focused on the option to be acquired by developing one innovation on a specific module. Instead of developing one innovation, the innovator could experiment and start several independent, parallel innovative efforts with respect to the same module. As a result, he would acquire the option to replace the existing module with the best of the new versions, if it is better than the existing one. The value of this option depends on the number of approaches pursued by the innovator.⁴² If there is uncertainty, increasing the number of experiments will increase the expected quality of the best result. At the same time, innovating on a module is not costless. After a certain number of experiments, the additional benefits may not be worth the costs.⁴³ Thus, the costs of innovation put an upper limit on the number of approaches that may be pursued to improve a module in a specific architecture.⁴⁴ Owing to the differences in the costs of innovation among architectures, different architectures may

justify a different number of experiments on a module. By comparing the number of experiments on a module that may be pursued under alternative architectures, one can get an indication of how much innovation will occur on this module in the different architectures.⁴⁵

The following model, which is based on work by Baldwin and Clark (2000), will help us to think more formally about the relationship between the benefits of additional experimentation and the costs of innovation on a module.

Innovating on one module provides the option to replace the existing module if the new module proves to be better. However, by starting *several* innovative efforts on a module, the innovator can replace the existing module with the best of the new ones if it is better than the existing one.⁴⁶ To understand the benefits that might accrue from such parallel innovation, we need to know the net option value of being able to select the best module among the module versions created by the different innovative efforts.

The model is based on the following scenario: an innovator starts k innovative efforts on the same module. After the innovations, she chooses the best of the resulting k module versions and compares it with the existing module.⁴⁷ If it is better than the existing module, she exercises the option by replacing the existing module and making the necessary adjustments in the rest of the system. If it is worse, she continues to use the existing module.

Thus, the model assumes that the selection of the best of the k alternative module versions can take place at the *module* level once the innovation is realized.⁴⁸ It is therefore not necessary to implement any changes in the rest of the system before the best innovation has been chosen and the decision to replace the existing module has been made. As a result, the innovator acquires the option by investing in k independent approaches to innovation. She exercises the option by making the adjustments in the rest of the system.

To simplify this idea, we can assume that there is only one innovator who owns the whole system and, for now, ignore the difficulties associated with coordinating changes across the components of a system. These challenges will be discussed in the next chapter.

The *net option value* of the option consists of the value of the option minus the cost of acquiring and exercising the option. To determine the present value of an option, we must translate uncertain outcomes in the future into a single number in the present. Options theory solves this problem by viewing the time between now and the future as a random

experiment leading to a specific outcome. The random experiment begins with acquiring the option and results in a specific payoff. In the terminology of probability theory, the value of the payoff function is a random variable that can be described by a probability distribution.⁴⁹ The *present value* of the option equals the *expected value* of this distribution, adjusted for time and risk.⁵⁰

The value of the option to select the best of k versions of a module equals the expected value of the best of k experiment results, as long as the best of the k experiment results is better than the existing module.⁵¹ Following Baldwin and Clark,⁵² we assume that a single approach to innovation on one module (module i) can be modeled by a random variable X_i that denotes the outcome of one innovative effort in terms of the increase in consumers' willingness to pay for the improvement in functionality or performance caused by the innovation; in other words, X_i denotes consumers' willingness to pay for the incremental value added by the improved functionality or performance of the module. We assume that X_i can be approximated by a normal distribution with mean μ_i and variance σ_i^2 . In a normal distribution, values around the mean are more likely than values far from the mean. Thus, this modeling assumes that, for innovative efforts, values close to the performance of the existing module are more common, and very good or very bad outcomes are rare.⁵³ The mean, μ_i , represents the existing performance of the module, normalized to zero. The variance, σ_i^2 , is a measure of the uncertainty underlying the innovative process with respect to this module.⁵⁴ This modeling embodies the assumption that very good or very bad outcomes are more likely if uncertainty is high (figure 4.2).

Given these assumptions and assuming the k approaches are independent,⁵⁵ the option value of replacing the existing module i with the best of k designs, $Q_i(k)$, can be described by the equation

$$Q_i(k) = \sigma_i \cdot Q(k),$$

with $Q(k)$ denoting the expected value of the best of k designs, as long as the best of the k designs is better than zero, of a standard normal distribution with mean 0 and variance 1.⁵⁶ (See box 4.2.)

The value of $Q_i(k)$ rises with k (that is with the number of experiments), but at a decreasing rate.⁵⁷ The value of $Q_i(k)$ also rises with uncertainty.

The cost of acquiring the option is the cost of making k experiments on the module. In the terminology of this book, the costs of making one experiment on the module are the "core costs of innovation" with respect to that module. The costs of acquiring the option increase with the

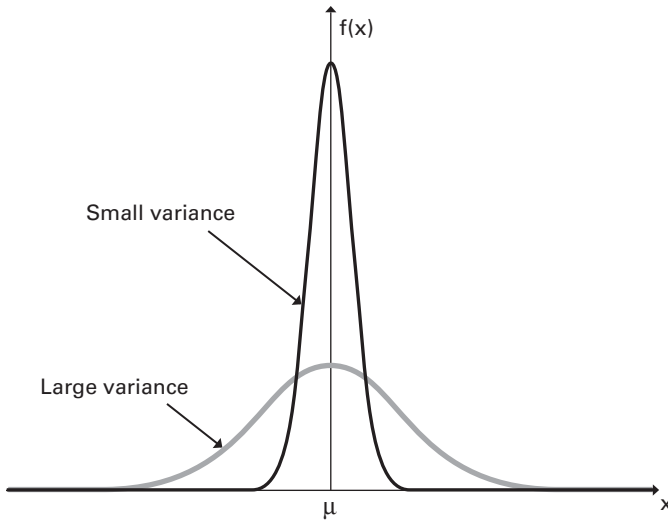


Figure 4.2

Probability density functions $f(x)$ of normal distributions with the same μ but different variance σ^2 .

number of experiments and with the core costs of innovation on that module.

The costs of exercising the option are the costs of adjusting the rest of the system to the changes between the existing module and the new module. In the terminology of this book, these are the “costs of system adaptation”; they must be paid if the innovator chooses the best of the k alternative innovations over the existing module. Changes in one module trigger corresponding changes in the rest of the system only if the change affects the visible information of the changed module *and* other modules directly or indirectly depend on this information. The number of modules that have to be changed is the number of modules that directly or indirectly depend on the module’s changed visible information. Furthermore, since existing modules will depend on the new module in different ways, the costs of adjusting a particular module may differ among affected modules. Thus, the cost of exercising the option increases with the number of modules that directly or indirectly depend on the module that is the target of the innovation, and increases with the costs of changing the affected modules.

Formally, the net option value of the option to select the best of k designs for module i is the value of the option minus the costs of acquiring

Box 4.2

Calculating the Option Value of Being Able to Select the Best of k Experiments

To calculate the option value, one must first construct a random variable that describes the value of the payoff function of the underlying experiment. In our case, the experiment consists of the following steps: First, an innovator takes k independent approaches to develop an innovation on module i . Each of these approaches can be interpreted as an independent experiment that is modeled by the random variable X_i that describes the result of one innovation on module i . The results are k specific values of X_i that describe the value of the resulting module relative to the value of the existing module. Next, the maximum of these values is selected. This experiment—"choose the best of k independent experiments described by X_i "—can be directly described by another random variable that yields the value of the best experiment. In statistics, this random variable is called *the maximum order statistic of a sample of size k* .^a The probability distribution of "the maximum order statistic of a sample of size k " describes all potential values of the best of k designs and their associated probabilities. It can be derived from the probability distribution of the random variable X_i using statistical theory.

The value of the option to select the best of k alternative designs of module i , $Q_i(k)$, can be determined by calculating the expected value of the positive value portion of the distribution of "the maximum order statistic of a sample of size k ."^b On the assumption that the k approaches are independent, calculating the value of the option yields

$$Q_i(k) = \sigma_i \cdot Q(k),$$

where $Q(k)$ denotes the expected value of the best of k designs (as long as the value of the best of the k designs is greater than zero) of a standard normal distribution with a mean of 0 and a variance of 1:

$$Q(k) = k \int_0^{\infty} z [N(z)]^{k-1} n(z) dz$$

where $N(z)$ and $n(z)$ are, respectively, the standard normal distribution and density functions.^c

a. Lindgren 1962, pp. 268–271.

b. The probability distribution of the maximum order statistic of a sample of size k describes all potential outcomes of the best of the k designs, both positive and negative. Because the innovator will not use the best innovation if it is inferior to the existing module, outcomes lying on the negative-value portion of the distribution (i.e., outcomes that are worse than the existing module, whose performance was normalized to 0) do not influence the option value of the innovation; for these cases, the value of the innovator's payoff function is 0. See Sanchez 1991, p. 30; Baldwin and Clark 2000, pp. 255–256, note 7.

c. See Baldwin and Clark 2000, p. 264, note 17.

and exercising the option:

$$\begin{aligned} \text{NOV}_i(k) &= Q_i(k) - k \cdot C_{\text{Innovation}}(\text{Module}_i) - \sum_j \text{"sees"} i (C_{\text{Adaptation}}(\text{Module}_j)) \\ &= \sigma_i \cdot Q(k) - k \cdot C_{\text{Innovation}}(\text{Module}_i) - \sum_j \text{"sees"} i (C_{\text{Adaptation}}(\text{Module}_j)), \end{aligned}$$

where module j “sees” module i if it directly or indirectly depends on module i 's changed visible information. The net option value varies with k , the number of approaches. The number of approaches k_{\max} that maximizes the net option value depends on the specific characteristics of the module. In particular, it depends on the degree of uncertainty about how best to improve the module as well as on the module's core costs of innovation and costs of system adaptation. The costs of system adaptation, in turn, depend on the visibility of a module, i.e., on the number of modules in the rest of the system that directly or indirectly depend on the module. If no other modules depend on a particular module, it is called a *hidden module*.⁵⁸ Innovating on such a module will not incur any costs of system adaptation, so that the costs of exercising the option will be zero.

The net option value formula can be used to predict how differences among modules (e.g., varied costs of innovation among architectures) affect how many innovative approaches innovators will pursue with respect to a specific module. The number of parallel efforts indicates how much innovation will occur on a module.

Exactly how many approaches innovators will pursue depends on the market structures of the design and product markets and on potential innovators' predictions and expectations of the nature of the competition in these markets. It therefore cannot be derived from architecture alone.⁵⁹ For example, if the design and product markets for a particular module are controlled by a single actor, he will choose the number of approaches that maximizes the net option value associated with that module. The monopolist will add an approach only if its additional costs—the core costs of innovating on this particular module—are less than the associated increase in expected value of the best approach. More formally, the monopolist will add an additional approach only if the cost of the $(k + 1)$ th approach is less than $|Q_i(k + 1) - Q_i(k)|$. In this case, the number of approaches is calculated by determining k_{\max} , the number of approaches that maximizes the net option value of the module. In contrast, if there is free entry into the design market and the product market is characterized by winner-take-all competition, firms will start additional approaches as long as the net option value of the total number of approaches is positive.⁶⁰ In this case, the number of approaches is calculated by determining the largest k for which the net option value is positive.

Table 4.1

Architectural features and likelihood of innovation.

	Parameters that change	Parameters held constant	Effect of change on likelihood of innovation
Observation 1	Core costs of innovation	Visibility, uncertainty, market structure	As the core costs of innovation on a module increase, fewer approaches will be pursued with respect to that module.
Observation 2	Uncertainty	Core costs of innovation, visibility, market structure	As the uncertainty surrounding a module increases, additional approaches on the module become more valuable and more parallel approaches are justified.
Observation 3	Visibility	Core costs of innovation, uncertainty, market structure	As a module's visibility increases, the number of experiments that affect its existing visible information decreases rapidly.
Observation 4	Core costs of innovation, visibility, uncertainty	Market structure	Relative to other modules with the same market structure, hidden modules with high uncertainty and low or medium core costs of innovation justify a particularly large number of parallel approaches.

Even without more information about the specific market structure, we can make the following four observations about how differences among modules may affect the relative likelihood of innovation on these modules (for a summary, see table 4.1).⁶¹ Different modules may incur different core costs of innovation, be more or less visible,⁶² be subject to varying degrees of uncertainty, and be developed and sold in markets with different market structures. For the first three observations, assume that out of the four parameters (core costs of innovation, visibility, uncertainty, and market structure) only one parameter changes, and the others stay the same.

First, other things being equal, the higher the core costs of innovation on a module, the fewer the approaches that will be pursued with respect to this module. The higher the number of independent approaches, the higher the option value of the best approach; that is, $Q_i(k) < Q_i(k + 1)$.

Thus, an additional approach always increases the expected value of the best approach.⁶³ The incremental value of an additional approach is decreasing in k ; that is, the incremental value of the $(k + 1)$ th approach is smaller than the incremental value of the k th approach; that is,

$$|Q_i(k + 1) - Q_i(k)| < |Q_i(k) - Q_i(k - 1)|.$$

At some stage, the incremental value of an additional approach is lower than the costs of the additional approach.⁶⁴ Thus, the number of approaches is limited by the costs of additional experiments, or by the core costs of innovation on that module. Remember the example given at the beginning of this subsection. Implementing a computer's functions for communicating with a printer in hardware rather than in software probably would result in fewer approaches for improving this functionality, owing to the higher core costs of innovation if the functions are implemented in hardware.

Second, other things being equal, the higher the uncertainty surrounding a module, the more valuable it is to add approaches, and the more parallel approaches will be justified. This is because the value of an option increases with uncertainty. If uncertainty is high, very good or very bad outcomes are more likely. Because of the loss-limiting characteristic of options, an innovator profits from the increase in upside potential without being affected by the increase in potential losses.⁶⁵ Therefore, the option value of experimenting on a module is highest in modules with a great deal of uncertainty.

Third, other things being equal, as the visibility of a module increases (i.e., as the number of modules that depend on this module increases), the number of experiments that affect its existing visible information will decrease rapidly. For example, an architecture that reduces the visibility of the printer to the rest of the system (e.g., by restricting applications' access to the functionality of the printer to the application programming interfaces of the operating system) will attract a higher number of innovations aimed at improving the printer than an architecture that lets applications access the printer's functionality directly. The system-adaptation costs associated with innovation on a module's visible information increase with the number of modules that depend on this information. Although they are incurred only once, high system-adaptation costs associated with ensuring that changed component behaves correctly with the rest of the system quickly neutralize potential gains from experimentation. As a result, it may not be economically justified to innovate on the existing visible information of highly visible modules.⁶⁶ This does not mean that there will be *no* innovation on visible modules. Any module can be a subject of

autonomous changes that do not require changes to the rest of the system and therefore do not suffer from the negative effect of high system-adaptation costs. Thus, some experimentation may go into improving the implementation of visible modules without changing their visible information. In addition, adding new interfaces to an existing module without changing its existing interfaces does not require any adaptations in the rest of the system. Existing modules can continue to use the old interfaces, and new modules can take advantage of the added functionality by using the new interfaces. The improved module is thus “backward compatible.”⁶⁷ In both cases, however, the constraints imposed by the need to adhere to the old visible information may considerably decrease the potential gains from such experiments. As a result, the net option value associated with such changes—and, consequently, the number of approaches that will be justified in view of this net option value—will often be lower than the net option value of (and the number of approaches to) innovation in hidden modules, where innovators can innovate freely without incurring any costs of system adaptation. Thus, although some innovation will take place no matter how visible a module is, visible modules will change much more slowly than hidden modules. In addition, owing to the need for backward compatibility, successive versions of visible modules will exhibit more continuity than successive versions of hidden modules. In essence, the *degree* of innovation will be less among versions; there will be fewer *meaningful* differences among iterations. Thus, the visibility of a module constrains its evolution. Put differently: By influencing the dependencies among modules, an architectural design influences the future evolution of the architecture: by placing certain functionality in hidden modules, architects enable that functionality to evolve more rapidly; by placing certain functionality in modules that are highly visible, architects accept that this functionality will evolve more slowly.

Fourth, relative to other modules developed and sold under the same market structure, hidden modules with high uncertainty and with low or medium core costs of innovation will justify a particularly large number of parallel approaches. The option value of experimenting on a module is highest for modules with a great deal of associated uncertainty. Since no other modules depend on them, changes in hidden modules never require changes in the rest of the system, so the option value is not reduced by system-adaptation costs. Finally, in view of the high value of additional approaches (due to the high uncertainty and the low to medium core costs of innovation), it takes a large number of approaches until the incremental benefit of an additional approach becomes lower than the costs of an

additional approach (the decision rule under monopoly in the design and product markets for the module) or until the net option value becomes negative (the decision rule under free entry into the design market and winner-take-all competition in the product market).

Costs of Change in the Original Architecture of the Internet

The general insights about the effects of architecture on the costs of innovation and on the likelihood of innovation in modular and integrated architectures gained above can help us understand how specific design choices in the Internet's original architecture affect these factors. The costs of innovation in the original architecture of the Internet were influenced by two design decisions:⁶⁸ using relaxed layering with the Internet layer as a technology-independent portability layer populated by a single protocol, and applying the broad version of the end-to-end arguments. Together, these led to the Internet Protocol's service interface that provides to higher-layer protocols a technology-independent and application-independent interface to the network infrastructure of the Internet. This design isolates applications and transport-layer protocols from the effect of innovation in the network infrastructure, and vice versa.

The following two subsections explain how these design decisions affect the costs of innovation at the application layer and at the link layer; and how the differences in innovation costs between architectures that are or are not based on these design principles affect the amount of innovation happening at these layers. They also highlight the effect of these architectural choices on the costs of production, distribution, and deployment.

By affording or constraining the implementation of functionality in either hardware or software, the architecture of a network influences the costs of producing and distributing an innovation; these costs affect the resources required to commercially exploit the innovation. Since they are borne by the economic actor seeking to produce and distribute the innovation (e.g., the innovator herself, or someone who licenses the idea from her), these costs influence what type of economic actor can commercially exploit an innovation.

Deployment costs consist of the one-time outlays that operators or users of a system must make before they can use an innovation for the first time.⁶⁹ These costs are especially noticeable in communication networks. Whereas in stand-alone systems changes can be installed one system at a time, communication networks are distributed systems that

consist of computers running protocols that must interoperate to create a functioning network. Before an innovation can be used in the operational network, the new software or hardware components that are needed to let it run must be installed on a potentially large number of computers in the network. A network architecture defines the “what” and the “how” associated with the deployment of an innovation—that is, which components in the operational network will have to be updated or replaced, and whether this can happen incrementally, through individual action, or whether it must be done in one large, coordinated effort that involves a potentially large number of actors. In essence, the scope of changes required for deployment and the associated costs define how difficult it is to adopt the innovation.

In addition, the deployment requirements implied by a network architecture interact with the economic system in which the network is used to constrain the evolution of the architecture.⁷⁰ As we will see in the final subsection, the deployment context of the commercial Internet⁷¹ has made it more difficult to change certain parts of the Internet’s architecture than a purely technological analysis of the structural dependencies within the architecture would suggest.

The Broad Version of the End-to-End Arguments and Innovation at the Application Layer

Networks that have been designed according to the broad version of the end-to-end arguments differ from other networks in how they distribute application-specific functionality among their layers. In networks that follow the broad version of the end-to-end arguments, applications and all application-specific functionality are concentrated in higher layers at the end hosts. Networks that deviate from these arguments place varying amounts of application-specific functionality in the core of the network. When designers create new applications, the differences in the distribution of functionality among the architectures result in different costs of innovation and deployment.

Adding an application in a non-end-to-end network will often require changing the network’s core.⁷² Existing application-specific functionality in the network’s core may prevent a new application from operating. In this case, the core of the network will have to be changed to enable the new application to function. Usually, this will require changes to the service and peer interfaces of lower-layer protocols. Changing a lower-layer protocol’s peer interface means coordinating changes to *all* computers running this protocol, both at end hosts and in the network. And since

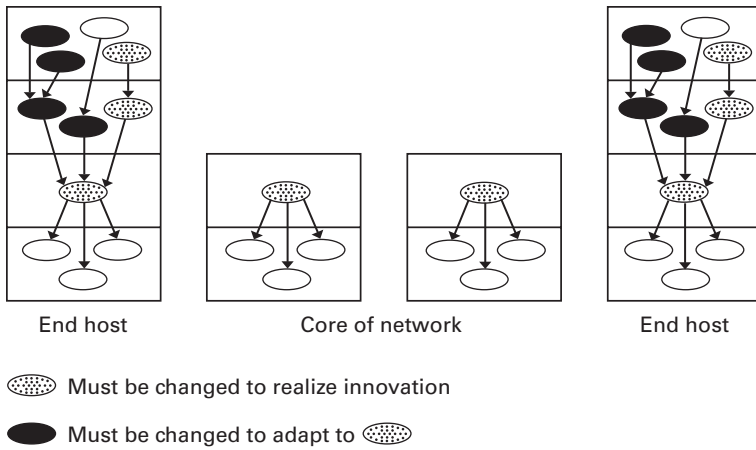


Figure 4.3
Application innovation in a network deviating from the broad version.

existing higher-layer protocols and applications may have come to depend on features in the core of the network, altering the network may require changing them, too. Changes to the core of the network may also be required if the new application has to implement supporting functionality in the core of the network—again requiring changes to higher-layer protocols, if they directly or indirectly depended on the changed network functionality (figure 4.3).

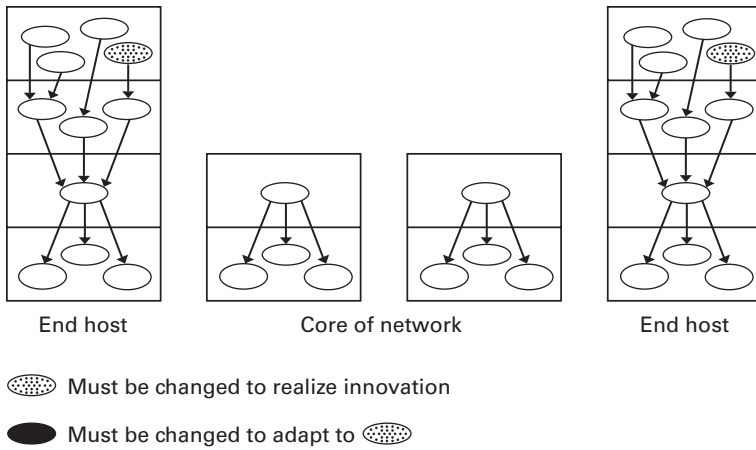
Consider an innovator who wants to develop a new Internet-telephony application in a network that provides reliable data transfer at the Internet layer, both hop-by-hop between routers and end-to-end between sending and receiving end hosts. Such an architecture violates the broad version of the end-to-end arguments.⁷³ Ensuring reliable data transfer in the Internet layer creates delays, because lost packets are retransmitted until they are transmitted successfully. Internet-telephony applications do not need this functionality, since they do not necessarily require that every one of their packets be successfully sent and received; their principal uses—people talking and listening to each other—can tolerate a significant amount of packet loss without affecting the overall experience. Telephony applications are, however, especially sensitive to delays. If the delay introduced by the reliability control is too large, the application will not be able to operate over the network.⁷⁴ This functionality may have to be removed to let the new application work properly. As a result of the changes to the core of the network, every application that depended on the reliable data

transfer provided by the network will have to be updated, which will incur large costs of system adaptation.

Thus, the core costs of innovation in a network that deviates from the broad version of the end-to-end arguments consist of the costs of designing, implementing, and testing the changes to higher layers at the end hosts and to the core of the network. Whereas higher-layer protocols are usually implemented in software, changes to the core of the network may require experimentation with hardware and software, which increases the costs of innovation. Since the application cannot be tested on the operational network, the innovator usually must have access to a test network.⁷⁵ Owing to the need to change not only the end hosts, but also the core of the network, the core costs of innovation associated with new applications are considerably higher in such a network than in a network that is based on the broad version. If the changes to the core of the network require new hardware, it must be manufactured and physically distributed. Thus, the costs of production and distribution associated with the innovation are potentially higher too. In addition, the changes to the core of the network may trigger huge costs of system adaptation.

A new application designed for such a network cannot run unless the core of the network has been changed, but changes to the core of the network usually cannot be made incrementally. They require a coordinated deployment of new hardware or software throughout the network. Since lower-layer protocols usually operate hop-by-hop across a network, adding a new lower-layer protocol requires changing all computers running the protocol. Therefore, to enable the application to run in a particular network, that network must be changed—and to ensure that the application can run across *all* networks attached to the global Internet, *all* networks must be changed.⁷⁶ These costs of deployment quickly can become exorbitant.

In an end-to-end network (such as the original Internet) in which network operators and application designers follow the broad version of the end-to-end arguments, designing, implementing, testing, and deploying a new application do not require changing the network's core (figure 4.4). First, since the core does not contain application-specific functionality, and since it provides only general services, it can support a wide variety of applications. As a result, it is unlikely that the core of the network will block a new application or that it will have to be changed to allow an application to be deployed.⁷⁷ Second, since the end-to-end arguments prohibit applications from incorporating application-specific functionality in the network, a new application will deliberately concentrate its functionality at higher layers at the end hosts. Third, since lower layers are

**Figure 4.4**

Application innovation in a network based on the broad version.

independent of higher layers, an isolated change at a higher layer will never require changes at a lower layer.

Thus, in an end-to-end network, developing a new application consists of writing a program that will run in a higher layer at an end host (figure 4.4). Higher-layer protocols operate end-to-end between end hosts: two specific protocol implementations—one at the source host and one at the destination host—cooperate to implement the protocol's services.⁷⁸ As a result, a new application can be tested using the operational network by installing it on two end hosts.

For example, an innovator who wants to develop a new Internet-telephony application in a network based on the original architecture of the Internet writes a software program that can be tested by installing it on two end hosts attached to the Internet. There is no need to change the core of the network: the Internet Protocol was designed to be as general as possible, containing no application-specific functionality (such as reliability controls to which telephony applications are especially sensitive) that might hurt an application's performance or degrade a user's experience (box 4.3).

All this results in very low core costs of innovation and zero costs of system adaptation. Because the innovator can build on the Internet's existing lower-layer infrastructure, she only needs to know how to design and write a distributed application (which requires access to and knowledge of a programming language), to have access to the application programming

Box 4.3

The Generality of the Internet Protocol

Some have argued that the service provided by the Internet layer is not general enough, insofar as the Internet Protocol does not offer Quality of Service.^a In particular, its best-effort service does not guarantee specific bandwidth or delay. Since it is not possible to construct a transport-layer service that guarantees delay, if the Internet layer does not guarantee delay, the best-effort service may preclude the use of applications that strictly require guaranteed delay. The same argument applies to applications that strictly require guaranteed bandwidth.^b Viewed from this perspective, the Internet Protocol's best-effort service may not be general enough, insofar as it effectively precludes development of applications in a certain class (those that cannot function properly without guaranteed bandwidth or delay). Though this argument is valid, Skype and other Internet-telephony applications have become quite successful in practice, even though they are forced to use the Internet's best-effort service. In particular, network engineers and application designers have developed techniques that help real-time applications cope with the problems resulting from best-effort service.^c Thus, so far, the fact that the Internet Protocol does not guarantee bandwidth or delay has not prevented the deployment of real-time applications (such as Internet telephony or video telephony) that may benefit from guaranteed delay. It is possible, though, that this may change if network congestion increases. As we saw in chapter 3, the broad version of the end-to-end arguments does not prevent the introduction of Quality of Service, although it constrains how Quality of Service can be realized.

a. Wu 2003a, pp. 147–149. Technologies that would enable the Internet to offer Quality of Service have been standardized by the Internet Engineering Task Force. As we will see later in this chapter, these technologies have been implemented, but they have not been widely deployed in the operational Internet.

b. Kurose and Ross 2008, p. 199.

c. Kurose and Ross 2008, pp. 608–618.

interfaces to the Internet's lower-layer protocols and knowledge of how to use them, and to have access to an end host connected to the Internet.⁷⁹ Insofar as software can be copied and distributed over the Internet at essentially no cost (the exact costs of distribution depend on the costs of operating a server or renting server space from which the software can be downloaded), the innovator has to incur only minor costs to produce and distribute the application.⁸⁰ In essence, this type of innovation requires almost no physical capital.⁸¹

The overall costs of developing an application and making it available to others on an ongoing basis (whether for a fee or not) differ depending on the type of application. For applications that run entirely on users' machines, not on computers provided by the application developers, the costs of renting or operating a server for distributing the software are the only necessary ongoing costs associated with the application, and even these costs can be reduced if the innovator uses an alternative peer-to-peer distribution mechanism such as BitTorrent.⁸² Peer-to-peer applications fall in this category: they usually consist of software that users run on their own computers and are largely independent of central servers. When it first came on the market, Skype was entirely based on a peer-to-peer structure. Apart from the servers used to download the software and to authenticate users, the application ran entirely on users' computers. As a result of this structure, adding a new user cost Skype one-tenth of a cent, as *Fortune* reported in 2004.⁸³ Client-server applications that run entirely on users' computers, not on computers provided by innovators, are another example of applications with this cost structure. For example, the Web server software developed by the Apache Foundation or Microsoft is downloaded by users, who then use the software to run their own Web servers.

In contrast, applications that need servers run by the innovator herself require a higher level of investment. Not only does the innovator incur the one-time costs of acquiring the hardware and software necessary to operate a server; these applications are also more expensive to operate on an ongoing basis, because of the costs of operating the server infrastructure. While an application does not have many users, innovators without additional funding may be able to run a server themselves at relatively low costs. But as the application gets more successful, the costs of operating the server infrastructure (e.g., renting physical space for the servers and paying for electricity, for air conditioning, and for the employees who manage the servers) and connecting it to the Internet become significant. At the same time, operating a server infrastructure is subject to considerable economies of scale. Firms (such as eBay, Amazon, and Google) that operate large data centers that host their servers are able to buy bandwidth, equipment, and software at much lower costs than smaller firms, putting smaller companies at a disadvantage.

Recent developments have drastically reduced the minimum level of investment needed to develop and operate server-based applications.⁸⁴ First, in the past, application developers interested in developing server-based applications had to buy the software necessary to operate a server

and buy programming tools to program the application-specific part of the server. Today's application developers can rely on a complete stack of open-source software for running servers, consisting of a Linux operating system, Apache Web server software, an open-source database such as MySQL or PostgreSQL, and an open-source application programming and development language such as PHP, Perl, Python, or Ruby. All these components are available for free. For the remaining programming tasks, Web-based matching platforms enable innovators to find freelancers from all over the world, which increases access to talented programmers and which potentially lowers costs. Second, following Moore's law, the cost of computer hardware has come down. Third, new hosting services such as those offered by Amazon enable small innovators to benefit (at least some) from the economies of scale realized by the hosting company. Pay-as-you-go offerings that let innovators increase the amount of server space as they need it, while paying only for used resources, reduce the need for up-front investments in server infrastructure when the ultimate demand for the application is not yet known.⁸⁵ As we will see in chapter 5, these developments have greatly increased the number and type of innovators who can develop these types of applications.

Finally, since a new application only has to be installed on two end hosts to be used between these hosts, it can be deployed incrementally.⁸⁶ The costs of deployment are the costs of installing the application on an end host, and are borne by the user of the application.

Since a non-end-to-end network has application-specific functionality in the network's core, innovators developing a new application that requires changes to the network's core will incur high core costs of innovation and high costs of system adaptation. Furthermore, because the application requires changes in the network, it cannot be run unless the network itself is altered; thus, deployment of the application requires a significant investment by network providers before user demand for the new application is known (table 4.2). In contrast, new applications developed for end-to-end networks have very low core costs of innovation and zero costs of system adaptation, and can be deployed incrementally (table 4.2).⁸⁷ Thus, deviating from the broad version of the end-to-end arguments considerably raises the threshold for application-level innovation and the requirements that potential innovators must meet. Owing to the significant differences in innovation costs, a network that uses the broad version of the end-to-end arguments will yield far more new applications, since such programs will more easily meet the threshold required to invest in an innovation. Architectures that deviate from the broad version will produce

Table 4.2

Costs of application-layer innovation.

	Network based on broad version of end-to-end arguments	Network deviating from broad version of end-to-end arguments
Core costs of innovation	Low (developing a software program)	High (developing a software program and developing changes to the network)
Costs of system adaptation	Zero	Potentially high
Deployment	Can be deployed incrementally by users	Changes to the network must be deployed first

far fewer new applications, because, in many cases, the associated costs of innovating will likely not exceed the benefits associated with the new application. In particular, applications targeting niche markets or low-value markets may be profitable under the architecture that is based on the broad version of the end-to-end arguments, but not under the architecture that deviates from that design principle.

The costs of innovating on an existing application depend on the circumstances. If the innovation requires changing the network's core, innovating on an existing application in non-end-to-end networks will cost the same as creating a new application. In this case, the real-options framework developed above would predict that, even if there is uncertainty, no or only few parallel approaches aimed at improving the innovation will be justified (owing to the high core costs of innovation and the high system-adaptation costs), which results in little or no improvement in existing applications (table 4.1, observations 1 and 3).⁸⁸

In contrast, an application in a network based on the broad version, on which no other application depends, is the prototype of a "hidden module" (that is, a module on which no other module depends).⁸⁹ Under these circumstances, the architecture lets innovators innovate on existing applications at low costs, without requiring adjustments in the rest of the system. According to real-options analysis, hidden modules with low core costs of innovation may justify a large number of parallel approaches aimed at improving the module (table 4.1, observation 4). The larger the uncertainty, the larger the number of parallel approaches that will be justified. At the same time, being able to experiment at low costs becomes more valuable to innovators as the uncertainty surrounding their applications increases (table 4.1, observation 2). Thus, the real-options framework

developed above predicts that, relative to their counterparts in an architecture deviating from the broad version of the end-to-end arguments, applications in end-to-end architectures will evolve much more rapidly, and that the rate of change will increase with uncertainty (box 4.4).

Box 4.4

Factors Influencing an Application's Rate of Change

A specific application's rate of change depends on (among other things) the architecture of the application, the ownership of the architecture's visible information and of the component implementations, the deployment context, the technical or business uncertainty surrounding the application, the structure of the application market, and the demand by consumers for the application. For example, an application-layer protocol (such as HTTP, the protocol governing the interaction between Web browsers and Web servers) that is maintained by a standard-setting organization will change more slowly than a proprietary application protocol such as Skype, since negotiating and coordinating changes among the members of the Internet Engineering Task Force, which maintains the HTTP standard,^a is more difficult and costly than having Skype (the company) decide on the proper evolution of the Skype protocols internally. The problems associated with distributed ownership of architecture are discussed in chapter 5.

As we will see in the discussion of innovation at the Internet and transport layer later in this chapter, changes that must be deployed by a large number of actors will happen more slowly and will be more incremental than changes that must be deployed by one actor. For example, if the client of a popular client-server application changes, the changes must be deployed by a large number of users before they can be used by all users; at the same time, the application must continue to function even if not all users have deployed the change yet. These requirements constrain the ways in which traditional client-server applications can evolve. For this reason, new versions of AOL's instant messenger are usually backward compatible; only a few new versions were not. Similarly, new versions of the Skype software are usually backward compatible. In contrast, Web-based applications are impeded neither by distributed ownership nor by deployment requirements, and thus they are able to evolve particularly rapidly. Web-based applications reside on a server run by the application provider; they are accessed by users through their browser. As a result, changes to the application do not require users to make any changes; this eliminates the need to deploy the application to a large number of users. In addition, the application code resides on the server and is under the sole control of the application provider; this eliminates the need to coordinate changes with other parties. As a result, Web-based applications can

Box 4.4

(continued)

evolve particularly rapidly. The ability to make, deploy, and reverse changes at low incremental costs makes these applications perfectly suited to a real-options approach to the resolution of technical or business uncertainty. For example, during the early days of Flickr, the Flickr team deployed more than ten new features a day. If the users liked them, the changes stayed; otherwise, the Flickr team restored the original feature (the “fallback option,” in real-options terminology).^b Similarly, Amazon and Google test the effect of changes to their website or search engine in real time on some of their users. If the changes are successful, they are extended to the whole user population; if not, the company continues to use the existing version of the application.^c

a. Within the Internet Engineering Task Force, the HTTP protocol is maintained by the Hypertext Transfer Protocol Bis (httpbis) Working Group (IETF 2008).

b. Garrett 2005.

c. Helft 2008; Varian 2007; Hansell 2007.

In addition, the two architectures support risk taking with respect to market uncertainty to different degrees. Whereas technological uncertainty is usually resolved during application development, market uncertainty often can be resolved only by the market. Since the incremental costs of making an application available to users in an end-to-end network are low, application developers can test their innovations in the market. Thus, the broad version of the end-to-end arguments enables them to resolve not only technical uncertainty but also market uncertainty at low cost.⁹⁰ If the application is successful, the innovator reaps the benefits. If users do not like the application, the application developer can stop offering it, or can change it (again, at low cost) in response to consumer feedback. As we will see in chapter 8, the online payment service PayPal and the photo-sharing platform Flickr emerged in this way. The relatively low incremental cost of failure due to market uncertainty (as opposed to technical uncertainty) will make it easier for application developers to pursue ideas for which market uncertainty is high.

In contrast, in a network deviating from the broad version, applications that require changes to the network’s core are very costly to deploy, which reduces network provider’s willingness to take risk with respect to market uncertainty. If the application is not successful, the application provider

can stop offering it, but the changes to the network will have been made. Thus, the incremental cost of failure due to market uncertainty (as opposed to technical uncertainty) are high. As a result, application developers and network providers will be less willing and able to take on projects for which market uncertainty is high.

Relaxed Layering with a Portability Layer and Innovation at the Link Layer

The Internet's architecture uses relaxed layering with a portability layer at the Internet layer. Compared with an architecture based on relaxed layering without a portability layer, a portability layer constrains the dependencies within the system. These architectural differences result in different costs of innovation at the link layer, and, consequently, different rates of change at this layer under the two architectures.

In both architectures, changing the link layer involves creating new or improved network architectures and protocols for the physical networks underlying the Internet. Innovating at this layer usually includes experiments with hardware and software. Even if the innovation is restricted to software, the innovator must have access to a physical network to experiment with and test the innovation.⁹¹ If the innovation involves new or improved hardware and is to be anything more than an experiment, the corresponding products have to be manufactured and physically distributed. As a result, this type of innovation has higher core costs of innovation than developing a new application in a network based on the broad version of the end-to-end arguments and requires a different type of investment: whereas developing a new application requires programming skills, access to application programming interfaces, and access to an end host, innovation at the link layer requires physical capital for developing the innovation; if the product includes hardware, it also requires that the innovation be produced and distributed.

In the architecture without a portability layer, all higher-layer protocols can potentially use the link-layer protocol. Changing the service interface of protocols at the link layer may require changing all higher-layer protocols that directly or indirectly depend on the changed interface, which will incur huge system-adaptation costs. For example, introducing a new physical network technology such as Wi-Fi may have required changes to transport-layer protocols or applications such as the browser, streaming video, or online gaming (figure 4.5).

In contrast, designating the Internet as a portability layer limits the dependencies within the system: higher-layer protocols can use all protocols down to the Internet layer but are not allowed to use protocols at

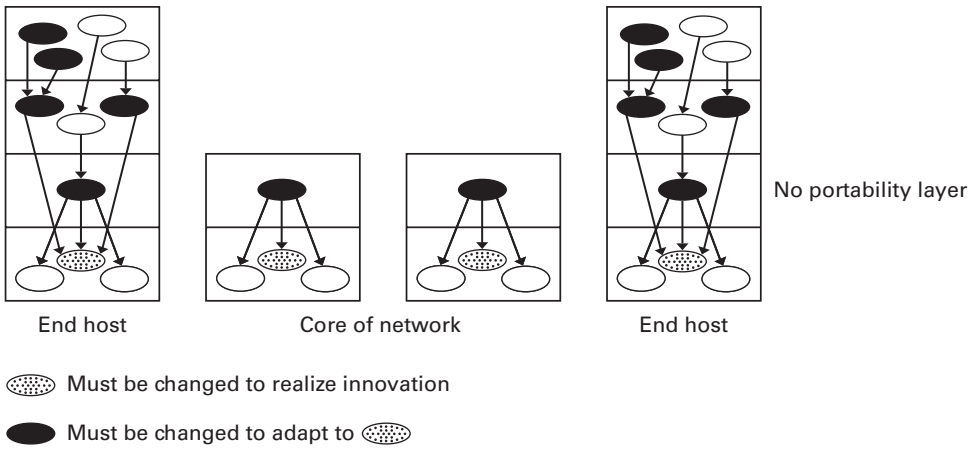


Figure 4.5

Link-layer innovation in a network without a portability layer.

layers below the Internet layer (e.g., the link layer). The service interface of the Internet Protocol has been carefully designed to be an abstract interface to the Internet’s network infrastructure and not to depend on particular features of any physical network technology. As a result, IP’s service interface does not have to change if the underlying network technology changes, as long as the changes to the underlying network technology conform to the IP abstraction.⁹² Since higher-layer protocols are only allowed to use the Internet layer, they are not affected by changes to the link layer, as long as IP’s service interface stays the same (figure 4.6).⁹³ Compared with an architecture without a portability layer, this design eliminates the system-adaptation costs associated with changing the link layer. Because this design was used for the architecture of the Internet, introducing Wi-Fi did not require changes to particular applications.

Thus, both architectures have the same core costs of innovation but very different costs of system adaptation (table 4.3). For new physical network technologies, the potentially high costs of changing an architecture without a portability layer considerably increase the threshold for innovation. Instead of simply being larger than the costs of developing the new technology, the expected benefits must also cover the costs of changing a large number of higher-layer protocols. Though the benefits associated with some technologies (e.g., Wi-Fi) may be large enough to justify these changes, the benefits of other technologies may justify their development costs but not the additional investments in adjusting higher-layer protocols.

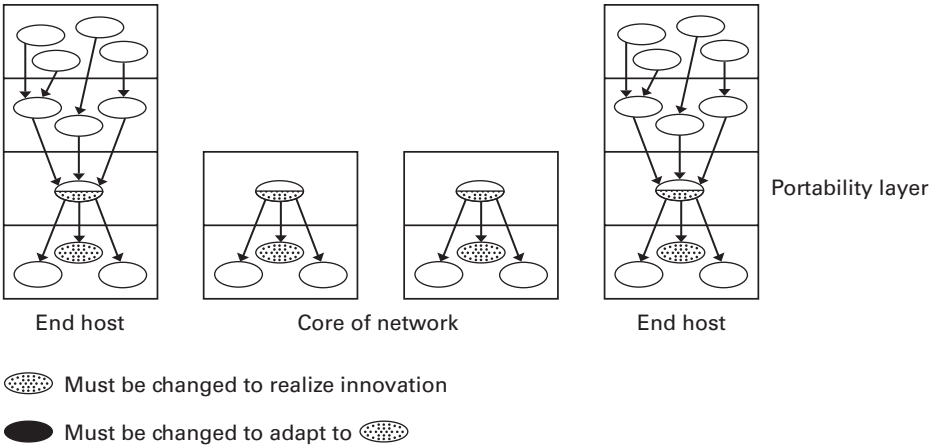


Figure 4.6
Link-layer innovation in a network with a portability layer.

Table 4.3
Costs of network-layer innovation.

	Network based on relaxed layering with portability layer at Internet layer	Network based on relaxed layering without portability layer
Core costs of innovation	Medium to high (may involve hardware and software and require a testbed)	Medium to high (may involve hardware and software and require a testbed)
Costs of system adaptation	Zero	Potentially high
Deployment	Can be deployed one subnet at a time	Depends on network architecture

According to the real-options framework developed above, components with no system-adaptation costs are particularly attractive targets for innovation and change rapidly. Components with the same uncertainty and the same core costs of innovation but higher system-adaptation costs will evolve much more slowly. To avoid these higher costs, subsequent versions of the service interface probably will be backward compatible. (See table 4.1, observations 3 and 4.) Thus, the introduction of a portability layer at the Internet layer drastically increases the likelihood and the rate of innovation at the link layer.

The evolution of the Internet is in line with these predictions. The Internet has been able to incorporate a large number of new physical network technologies (e.g., Wi-Fi, WiMax, new Ethernet technologies, ATM, frame relay, optical network technologies) without changes to the Internet layer.⁹⁴

In the Internet's architecture, a new link-layer protocol or physical network technology can be deployed subnet by subnet (i.e., one physical network at a time). The Internet was designed to interconnect subnets regardless of their physical network technology. In essence, the Internet layer makes different link-layer protocols functionally independent. To connect a new type of subnet to the Internet, all one needs is a router that can connect this type of subnet to a physical network that uses an existing network technology.⁹⁵

Internet Architecture and Innovation at the Internet and Transport Layers

The original architecture of the Internet lets protocols at the network layer and the application layer change rapidly. By restricting dependencies within the system, the broad version of the end-to-end arguments and the relaxed version of the layering principle with a portability layer isolate the rest of the system from changes to network-layer and application-layer protocols. These design choices make it possible to innovate on application-layer protocols (as long as no other application-layer protocols depend on them) and on link-layer protocols (as long as the IP service interface stays the same) without any system-adaptation costs, both when new protocols are added and when existing protocols are innovated upon.

In contrast, the service interfaces of the Internet Protocol and of the transport-layer protocols are used by a large number of higher-layer protocols. Since the Internet Protocol is the only protocol at the Internet layer, and since all network communication goes through this protocol, all higher-layer protocols depend on it directly or indirectly. As all existing applications use one of the transport-layer protocols, transport-layer protocols are less visible than the Internet Protocol, but they are still highly visible.

By placing a large number of dependencies on the service interfaces of the Internet Protocol and the transport-layer protocols, the architecture constrains the evolution of these interfaces (table 4.1, observation 3). Thus, we would expect little or no change in the service interfaces of existing transport-layer protocols (such as TCP and UDP) or in the service interface of the Internet Protocol.

As we saw above, architects may try to avoid the high costs associated with changing the visible information of highly visible modules by realizing their goals in other ways (table 4.1, observation 3 and accompanying text). For example, they may add new components, add new interfaces to existing modules, or make changes to the service interface in ways that are backward compatible. The evolution of the Internet's architecture at the Internet layer and at the transport layer shows the soundness of these predictions (box 4.5).

Architectural features such as the number of dependencies between components facilitate or constrain the evolution of the architecture by influencing the costs of system adaptation. There is, however, another factor that constrains the evolution of a network architecture's visible information: the structure of the economic system in which the network will be used.⁹⁶ The economic system in which the network is used consists of the actors who use and operate the network, the relationships among them, and the governance structures through which they interact. By providing the context in which innovations are to be deployed, this economic system may constrain the evolution of certain parts of an architecture more than simply considering the dependencies within an architecture would suggest.

A network's architecture affects which components will have to be changed to deploy an innovation and what the costs and benefits of deployment will be. For example, in an architecture based on the broad version of the end-to-end arguments, deploying a new application requires installing the new application on two end hosts. In contrast, deploying a new application in a network that deviates from the broad version may require changing the end hosts and the network's core. The economic system in which the network is used influences who controls the components that must be altered and what incentives these actors may have to make these changes. For example, in the commercial Internet, end hosts are generally controlled by users whereas the network's core is controlled by many different providers. Thus, in an end-to-end network with a deployment context similar to the commercial Internet, the small costs of deploying an application are borne by users who gain the immediate benefit of being able to use the application. In a non-end-to-end network with the same deployment context, network providers may need to make substantial investments to enable an application to run on their network when it is not clear whether users will value the application enough for the providers to recoup their investments. As we will see in chapter 7, these differences make it much more likely that new and unproven applications or

Box 4.5

Evolution of the Internet Architecture at the Internet Layer and the Transport Layer

At the transport layer, the Internet Engineering Task Force, the organization that develops most of the protocol standards for the TCP/IP protocol suite, has standardized several new transport-layer protocols^a; the service interfaces of the existing transport-layer protocols, TCP and UDP, have stayed the same.

At the Internet layer, the IETF has standardized several protocols that would augment the Internet layer's functionality with services such as Quality of Service, IP Multicast, and support for mobility.^b These protocols add to the Internet Protocol's existing service interface but do not change it.

The IETF has also standardized IPv6, a new version of the Internet Protocol.^c This protocol is intended to replace IPv4, the existing version of the Internet Protocol.^d At its core, IPv6 introduces a new format for IP addresses that provides a greater number of addresses than the current version. With the current address format, the total number of unique IP addresses on the Internet is limited to 4,294,967,296 (that is, 2^{32}). Owing to the rapid growth of the Internet and the advent of mobile laptop computers, personal digital assistants, mobile phones, and devices that are always connected to the Internet (such as cable or DSL modems), the Internet is expected to run out of IPv4 addresses soon.^e Since the IP address is used in IP's service and peer interfaces,^f the replacement requires changing both IP's service interface and its peer interface. Owing to the large number of higher-layer protocols that directly or indirectly depend on IP, this change will result in substantial system-adaptation costs. All applications that directly store or transmit IP addresses will have to be adjusted to accommodate the new address format. Without the increase in IP addresses, the Internet would not be able to continue to operate with a universal address space. Thus, in the case of IPv6 the innovation's expected benefit (enabling the continued operation of the Internet) is large enough to justify the high costs of system adaptation.

In sum, in line with the predictions of the real-options analysis developed above, the designers of transport-layer protocols and the Internet Protocol have generally added new protocols (at the transport layer) or new service interfaces (at the Internet layer) instead of changing the existing service interfaces of TCP, UDP, and IP. In the one case in which the service interface of IP was changed, the improvement is necessary for the Internet's successful operation and growth.

a. The new transport-layer protocols are not yet widely deployed. The problems with deployment are attributable to deviations from the broad version of the end-to-end arguments and will be discussed in the concluding chapter.

Box 4.5

(continued)

b. The current version of the Internet Protocol provides a single “best-effort” service, which does not provide any guarantees as to the level of service a data packet receives. In contrast, Quality of Service solutions provide different types of service (for example, some offer guaranteed bandwidth or delay; others give some traffic priority over other traffic without giving absolute guarantees). See Peterson and Davie 2007, pp. 499–525. IP Multicast is a service that simultaneously delivers information to a potentially large number of receivers without creating unnecessary copies of the information. After the sender sends one copy of the data packet into the network, the network takes care of delivering the packet to all receivers, with routers in the network making copies of the packet as needed. See Peterson and Davie 2007, pp. 313–334. Mobile IP allows a mobile end host to move from one network to another without changing its IP address. See Kurose and Ross 2008, pp. 564–569.

c. For short overviews of IPv6 and its history, see Peterson and Davie 2007, pp. 318–329; Kurose and Ross 2008, pp. 360–366

d. Version number 5 was assigned to an experimental protocol, ST-2, that was never standardized. See Kurose and Ross 2008, p. 360.

e. Huston (2008) predicts that the Regional Internet Registries (RIRs), which are responsible for allocating IP addresses within a certain region, will allocate the last unallocated IP address in the year 2011. In the summer of 2007, the RIRs of North America (ARIN) and Latin America and the Caribbean (LACNIC) and the local registry of Japan (JPNIC) issued statements noting the likely exhaustion of unallocated IP addresses in the next few years and strongly recommending transition to IPv6 in their respective geographic areas. See American Registry for Internet Numbers 2007; Latin American and Caribbean Internet Addresses Registry 2007; Asia Pacific Network Information Centre 2007.

f. The transport-layer protocol hands the IP address of the destination to the Internet Protocol through IP’s service interface; the IP addresses of the two end hosts that are sending and receiving the datagram are parts of IP’s peer interface.

applications serving small, widely dispersed audiences will be deployed in an end-to-end network than in a non-end-to-end network.

To see how a network's deployment context constrains the evolution of its architecture, consider two examples. First, the peer interfaces of the existing transport-layer protocols have evolved much more slowly and incrementally than real-options analysis would predict. Second, many changes involving the Internet Protocol's visible information have been standardized and implemented (as real-options analysis would predict), but they have not yet been widely deployed.

Consider first the evolution of the transport-layer protocols' peer interfaces. By separating the vertical service interface through which higher-layer protocols access a protocol's services from the horizontal peer interface through which the protocol peers cooperate to produce the protocol's service, a layered network architecture makes it possible to change the peer interface without changing the service interface. Recall the real-world analogy introduced in chapter 2: the protocol at the second-highest layer, the mailroom layer, provided person-to-person delivery of documents between company employees. Its service interface required the higher layer to provide the documents to be delivered and the name of the receiver, while its peer interface described the format and meaning of the messages that the mailrooms at the sending and receiving location exchange to implement the mailroom services and actions. In the example, the internal mail code used by the mailroom employees to identify the location of people within the company was part of the mailroom protocol's peer interface. This design lets parts of the peer interface (e.g., the internal mail code) change without changing the mailroom protocol's service interface: Since the service interface asks for the name of the receiver (not her internal mail code), it leaves the job of looking up the internal mail code to the mailroom layer, and the service interface is not affected by the change in the mail code.⁹⁷

The ability to change how the various entities belonging to a protocol communicate and cooperate without affecting the protocol's service interface makes it possible to change a protocol's internal operation without affecting any higher-layer protocols that may use it. The ability to avoid any system-adaptation costs by changing the peer interface in ways that do not affect the service interface is particularly valuable for protocols with highly visible service interfaces (e.g., transport-layer protocols or the Internet Protocol).

Since transport-layer protocols are implemented in higher-layer software at the end host, they have relatively low core costs of innovation. A real-options analysis would suggest that changes without system-adaptation

costs (e.g., changes to these protocols' peer interfaces that do not affect the service interface, or changes to the internal implementation of the protocols that do not affect any interface)⁹⁸ would attract a significant amount of innovation. As a result, we would expect the peer interface of the transport-layer protocols to evolve much more rapidly than their service interface. (See table 4.1, observations 3 and 4, and accompanying text.)

But although there have been changes to TCP's peer interface, they were all designed to be backward compatible, and they could be deployed incrementally.⁹⁹ Thus, the changes to TCP's peer interface were much more incremental than we would expect. Instead of being constrained by characteristics of the architecture alone, the evolution of TCP's peer interface was constrained by the deployment context provided by the economic system that uses the Internet.

Before a new peer interface can be used in the operational Internet, the changes have to be implemented and the new implementations have to be deployed. Higher-layer protocols such as application and transport-layer protocols operate end-to-end between end hosts; in general, protocol implementations using the new versions of the peer interface have to be installed on two end hosts to be used between these two end hosts. Unless the new version of the peer interface is backward compatible, protocol implementations based on different peer interface versions will not cooperate. Thus, in the absence of backward compatibility, it will be necessary to deploy the new version of the protocol on all end hosts using that protocol to ensure that they will be able to communicate.

If a protocol is used by only a small number of end hosts, or if the instances of that protocol that currently run in the network are controlled by a small number of actors, it may be possible to switch all protocol instances in use to the new version simultaneously. In this case, the new version of the peer interface does not have to be backward compatible and can evolve freely from one generation to the next.

However, if there are many different instances of the protocol controlled by a diverse set of actors, it is not likely that all protocol instances will be upgraded at the same time. To ensure that the protocol continues to operate under these circumstances, its designers must let the protocol's different generations co-exist and interoperate.¹⁰⁰ This need for backward compatibility may severely constrain whether and how quickly a protocol evolves. Protocol designers can alleviate the problem by designing peer interfaces that can later be extended. For example, they can require two different protocol instances to negotiate which version should be used,

allow the use of options, or reserve fields in the protocol header that can be defined at a later stage.¹⁰¹ Still, a solution that uses options and allows different protocol instances to negotiate the use of these options may be more cumbersome than a solution that does not need to adhere to the existing specification of the protocol header.¹⁰² This is problematic not only for transport-layer protocols (such as TCP) but also for application-layer protocols (such as HTTP) that are widely deployed and whose instances are under the control of a large, diverse group of actors.

Thus, the context in which a higher-layer protocol is deployed constrains how its peer interface can evolve. If the protocol is used widely and the protocol instances that run in the network are controlled by many different actors, its peer interface will change in ways that are backward compatible, are incrementally deployable, or take advantage of extensibility mechanisms that were designed into the protocol from the start. If the protocol is used by only a small number of end hosts, or if the instances of that protocol that currently run in the network are controlled by a small number of actors, its peer interface can evolve in more radical ways that require a coordinated deployment.

For a second example, consider the changes to the Internet layer standardized by the Internet Engineering Task Force (IETF). The IETF has created a number of protocol standards to augment the existing Internet Protocol, for example by providing Quality of Service, Multicast, or support for mobility. In addition, they also standardized a new version of the Internet Protocol, IPv6. By providing a larger address space than the existing version of the Internet Protocol, the new version of the protocol is designed to prevent the Internet from exhausting its supply of IP addresses (box 4.5). As was noted above, these innovations confirm the predictions made in real-options analysis. Although these protocols were standardized by the IETF and implemented by major vendors of routers and operating systems, they have not yet been widely deployed.

This lack of deployment is attributable to the deployment context provided by the economic system that uses the Internet. The Internet Protocol is deployed on all end hosts and routers in the operational Internet. As a result, adding functionality at the Internet layer or replacing the existing version of the Internet Protocol requires that end hosts and routers be changed. To ensure that the new functionality can be used within a network, it must be deployed on all end hosts and routers on that network. If the functionality is intended to work between *any* two end hosts connected to the Internet, it must be universally deployed on *all* Internet end hosts and routers.¹⁰³

The commercial Internet consists of a large number of providers who operate networks for profit. The original versions of Quality of Service and IP Multicast were not well suited to this multi-network, for-profit deployment context.¹⁰⁴ There was rarely a clear business model that would have let network providers recoup their investments in upgraded infrastructure. The technologies were very complex, and the tools needed to manage them in an operational network were not available. Although data packets often cross multiple network boundaries on their way from the sending to the receiving end hosts, the technologies did not address problems associated with using the technologies across network boundaries.

Thus, while these technologies required a large initial capital investment and significant ongoing costs, their benefits were uncertain. As a result, most network providers did not have an incentive to deploy them. While the technological complexity would still have been an issue with one nationwide, non-profit network, the lack of a business model and the problems associated with inter-domain operation would have been less significant. (For a more detailed discussion of the interaction between architecture and deployment context in the case of multicast, see box 4.6.)

In addition to these technology-specific issues, the structure of the commercial Internet seems to create a more general problem. To make new Internet-layer functionality universally available, all network providers must deploy it in a coordinated way. The need for coordinated group effort creates coordination problems or collective-action problems.¹⁰⁵ Moreover, since network providers have to act together to deploy Internet-layer functionality, deploying such functionality does not create a competitive advantage.¹⁰⁶ This is particularly problematic if there is no clear business model that would let network providers recoup their investments.

There were real problems with the original Quality of Service or IP Multicast proposals that prevented their adoption. Thus, it is not clear to what extent the more fundamental coordination or incentive problems just described helped prevent the deployment of these technologies, or what would have happened if the technologies had better addressed the needs of network providers in the commercial Internet by offering clear benefits for network providers, providing management tools, and tackling issues associated with using the technologies across network boundaries. Some of these shortcomings have been addressed since the protocols were first standardized, so the technologies may still be deployed.

Based on the experience with the deployment (or lack thereof) of Internet-layer technologies, networking researchers have developed two sets of strategies.

Box 4.6

Multicast and the Interaction between Architecture and Deployment Context

Architecture affects which components must be changed in order to deploy an innovation; it also affects the costs and benefits of deployment. The economic system in which the network exists and is used influences who controls the components that must be changed and what incentives these actors may have to make these changes.

Consider multicast, a service that simultaneously delivers information to a potentially large number of receivers without creating unnecessary copies of the information. This service is especially useful for applications used to webcast live events that simultaneously transmit audio or video streams to a potentially large number of viewers, or for video conferencing and online games that transmit data from one participant to all other participants.^a In the absence of a multicast service, the sender would have to send the same message to every receiver. Multicast lets the sender send only one version of the message, with the multicast service making copies as needed. In comparison with the traditional model, multicast can drastically reduce the number of messages the network has to transmit.^b

IP multicast realizes the multicast service at the Internet layer. The sender sends the message into the network; the network then takes care of delivering the message to the receivers, routers making copies of it as needed.^c An alternative architectural solution, application-layer multicast, implements its solution at the application layer, embedding the multicast functionality within a particular application. In this solution, the different instances of the application cooperate to deliver messages to everybody, while minimizing the number of messages.^d

The two solutions offer different deployment benefits and costs.^e IP multicast is more efficient than application-layer multicast; for example, application-layer multicast may transfer the same packet over the same link several times, using non-optimal trees to distribute the packet. Thus, the bandwidth savings of application-layer multicast are lower than those of IP multicast. Deploying IP multicast requires changing applications that want to take advantage of IP multicast as well as altering both the service interface and the peer interface of IP at the end hosts; it also requires changing every router on the way from the sender to the receiver. Today, major end-host operating systems and routers are capable of supporting IP multicast. To make IP multicast universally available (i.e., to enable it to be used between any two end hosts attached to the Internet), it would have to be deployed on every router in the Internet.^f In contrast, to allow a specific group of senders and receivers to use application-layer multicast, only the sending and receiving applications that belong to this group have to support it.

Box 4.6

(continued)

Now consider two different deployment contexts: the commercial Internet and an enterprise network. Consider first the commercial Internet. The bandwidth savings from IP multicast primarily benefit senders who send information to many different receivers.^g While these benefits accrue to end users, there are no good business and charging models which would enable network providers to transfer at least part of that benefit to themselves. While the end hosts also have to change, the network provider must invest in an upgraded network and incurs considerable operational costs of managing the increased technical complexity with poor tools.^h Thus, for many network providers, the expected benefits do not justify the expected costs.ⁱ While application-layer multicast provides comparably lower benefits, it only requires those users who would like to take advantage of it to install the corresponding application, giving them an immediate benefit at a comparatively low cost.^j With this distribution of deployment costs and benefits, it is not surprising that IP multicast isn't widely deployed in the commercial Internet, while application-layer multicast has been relatively successful.^k

Now consider an enterprise network. Enterprise networks typically have limited bandwidth. If the company intends to use the network to stream video to its employees (e.g., to distribute enterprise communications or educational offerings), sending the same packet to a large number of employees may overwhelm the network.^l In this case, the bandwidth savings from introducing IP multicast may justify the costs. Since the company operates both the applications and the network, it benefits from the bandwidth savings. Thus, in this context, the costs and benefits of using IP multicast are aligned in a way that they are not aligned in the commercial Internet. At the same time, some of the operational issues associated with managing multicast (e.g., ensuring the security of the sessions) are easier to tackle within a single, tightly controlled administrative domain.^m Finally, since the single enterprise controls end hosts and routers on its network, it does not face the coordination difficulties that help prevent deployment in the commercial Internet. In view of the differences between the two deployment contexts, it is not surprising that deployment of IP multicast has been quite successful in enterprise networks, whereas IP multicast has failed to take off in the commercial Internet.ⁿ

a. Diot et al. 2000, p. 80. It is also useful for sending messages to more than one receiver to synchronize copies of information.

b. In addition, contrary to the traditional unicast model, a multicast sender does not need to know the addresses of the potential receivers (Garber 1999, p. 78).

Box 4.6

(continued)

- c. As multicast is not needed by all applications, the provision of multicast in the Internet layer violates the broad version of the end-to-end arguments. According to Reed, Saltzer, and Clark (1998, p. 69, note 1), the authors of the original end-to-end paper, this is one of the rare cases in which the deviation is justified by the cost savings. See also Peterson and Davie 2007, p. 697. For an overview of IP multicast, see Peterson and Davie 2007, pp. 329–343; Kurose and Ross 2008, pp. 408–415.
- d. For an overview, see Hosseini et al. 2007.
- e. See Hosseini et al. 2007, pp. 60–61; Thaler and Aboba 2008, pp. 20–22.
- f. It may be possible to work around this requirement using overlays; see Peterson and Davie 2007, p. 696. This strategy may be difficult to execute, though. For example, MBone, the overlay network that was intended to enable the use of IP multicast until all routers were able to support IP multicast directly, ran into problems as the number of users increased. See Almeroth 2000, p. 14; Thaler and Aboba 2008, p. 21.
- g. Diot et al. 2000, p. 80; Thaler and Aboba 2008, p. 20.
- h. Diot et al. 2000; Sharma, Perry, and Malpani 2003.
- i. Diot et al. 2000, pp. 82–83; Thaler and Aboba 2008, p. 20.
- j. Hosseini et al. 2007, pp. 60–61; Thaler and Aboba 2008, p. 20.
- k. Hosseini et al. 2007, pp. 60–61; Thaler and Aboba 2008, pp. 20–22.
- l. Eubanks 2006, pp. 22–23.
- m. Ganjam and Zhang 2005, p. 160; Sarac and Almeroth 2005, p. 85; Hosseini et al. 2007, p. 60.
- n. Ganjam and Zhang 2005, p. 163; Sarac and Almeroth 2005, pp. 85, 86; Eubanks 2006, p. 66.

Some advocate changing how Internet-layer protocols are designed to make them more deployable in the commercial Internet. They accept the multi-provider, for-profit nature of the Internet and the fundamental incentive problems that impede the deployment of new Internet-layer technologies. According to them, protocols and other solutions at the Internet layer should be designed in a way that increases their chances of being deployed in this context.¹⁰⁷ This entails exploring business models, developing management tools, and addressing problems associated with the use of Internet-layer protocols across network boundaries.¹⁰⁸ It entails creating solutions that better align adoption costs with their benefits, so that the actor who has to make changes to deploy the innovation also gains the benefits.¹⁰⁹ And it entails designing solutions that can be deployed

incrementally (one network at a time) while still providing benefits to those deploying them.¹¹⁰ These guidelines severely constrain the range of potential solutions that can be considered at the Internet layer. Because these guidelines have been developed to enable networking researchers to innovate more successfully within the deployment context provided by the commercial Internet, they show how the Internet's deployment context constrains the evolution of the Internet layer.

Others suggest solving the deployment problem in a more fundamental way: by changing the architecture to give network providers more incentives to deploy new Internet-layer technologies.¹¹¹ They argue that network providers lack incentives to deploy new technologies because the provider who first implements a new Internet-layer technology does not gain a significant competitive advantage. They advocate for introducing mechanisms into the Internet's architecture that let all users attached to the Internet upgrade to a new Internet-layer protocol even if their network provider does not yet offer it. By driving traffic and increasing revenue (through increased settlement charges from interconnection) to the provider offering the new protocol, this solution would reward first movers among network providers and would put pressure on competitors to upgrade. Such an architectural solution would enable several potentially incompatible Internet-layer protocols to coexist until everyone had migrated to the new technology. Proponents of this solution view the coexistence of several potentially incompatible architectures as a transitory occurrence. Some researchers advocate a more radical approach. According to them, the coexistence of several potentially incompatible architectures should become the norm. Instead of providing a mechanism that enables the Internet to transition from one Internet-layer protocol to the next, the infrastructure and the mechanisms necessary to support different coexisting architectures would form the new core of the Internet's architecture, with different coexisting network architectures on top of it.¹¹²

In the first set of approaches, the deployment context directly constrains the Internet layer's evolution; these approaches accept the constraints resulting from the structure of the commercial Internet and react by constraining the Internet layer's evolution in ways that make it better suited to the deployment context. The approaches in the second set also react to the deployment challenges posed by the commercial Internet. However, instead of accepting the constraints, these approaches (whether they view the coexistence of multiple architectures as a transitory mechanism or a permanent feature of the architecture) aim at changing the constraints by changing the architecture.¹¹³ These two sets of approaches

mirror the different ways in which, according to North (2005), organizations and individual actors may react to a given set of constraints. They can play by the “rules of the game” and look for the best strategy given the existing set of constraints (effectively adjusting their behavior to the set of constraints); this is the strategy pursued by the approaches in the first set. Or they can seek to change the “rules of the game” and change the constraints under which they operate; this is the strategy pursued by the approaches in the second set. Though it is too early to judge the merits of these approaches, they show that the economic system in which the Internet exists and is used can significantly influence the evolution of its architecture.

5 Architecture and the Organization of Innovation

This chapter explores how a system's architecture affects the organizational structures in which the development and production of the system, and subsequent innovations, can take place.¹ Product architectures constrain the options for organizing the initial development and production of a system's components, and for organizing subsequent innovation, by enabling or disabling arm's-length relationships. Once an architecture has been created, detailed design can begin. During detailed design, designers must decide how each component will be structured to provide the functionality assigned to it by the architecture; subsequently, each component must be implemented or manufactured according to this specification. Features of the architecture influence whether this work should take place within the boundaries of a single firm or whether independent economic actors can develop various components. This link has important economic implications. First, an architecture created by a single firm affects how that firm can organize the subsequent development and production of components for that architecture. Thus, an architecture influences the organizational options available to individual firms. Second, the constraints imposed by the architecture constrain the potential structure of the industries that develop and produce components for the architecture. Finally, these mechanisms determine whether economic actors other than the system's architect can participate in the development and production of the system's components, both initially and later. Thus, architectural choices have important implications for the range of potential innovators who can innovate with respect to a particular architecture. As we will see in chapter 8, the size and diversity of the innovator pool may, in turn, influence the amount and the type of innovation that can occur with respect to an architecture.

The first section of the chapter explains how exactly an architecture may influence a firm's organizational choices. On the one hand, the

architecture of a system provides a framework for dispersing work among design teams, the smallest organizational units involved in the design of the system. As the first subsection shows, the boundaries between components in the architecture determine the boundaries between design teams in the economic system that emerges to develop and produce system components. The second subsection examines how the architecture of a system constrains the feasible organizational structures for the development of the system. The analysis focuses on two types of costs—coordination costs and transaction costs—that economic research has identified as relevant for determining the vertical boundaries of firms. It explains how coordination costs and transaction costs affect firms' boundary choices, and it shows how different product architectures influence these costs. It then uses these insights to examine how modular and integrated product architectures affect the feasible governance structures for innovation at the level of individual components and at the architectural level, both initially and later.

The second section explores what the link between architecture and feasible governance structures in modular and integrated architectures means for the organizational options available to individual firms, the potential structure of the industries for the development and production of system components, and the ability of independent economic actors other than the system's architect to innovate.

The third section applies the insights gained in the preceding sections to the question of how the original architecture of the Internet affects the organization of innovation. It explores the impact of the Internet's original architecture on the organizational options available to individual firms, on the potential structure of the industries for the development and production of system components, and on the ability of economic actors to innovate, particularly in applications.

Architecture and Organization in Modular and Integrated Architectures

There is a close relationship between a product's architecture and the economic structure of the system that emerges to develop and produce the product's components. The architecture of a system constrains the feasible organizational structures for developing and producing the system. First, an architecture provides a framework for dispersing tasks among design teams, the smallest organizational units involved in the design of a product. Second, features of the architecture determine whether the different design teams must be within the same firm or can belong to different, indepen-

dent firms. Thus, the architecture determines whether independent component development is feasible.

How Architecture Affects Design Teams

Designing a complex system requires a large number of designers. In development projects, the designers involved in the design of a system are organized in several design teams, each of which is responsible for designing one component. Thus, the organizational structure of the project mirrors the technical architecture of the system: components within the architecture correspond to design teams in the development project.² This division of labor conforms to the predictions of classic organizational theory,³ is described by the literature on product and software design,⁴ and is widely observed in practice.⁵

According to classic organizational theory, rational organizations assign tasks to organizational units such that overall coordination costs are minimized.⁶ In particular, a set of reciprocally interdependent tasks will be allocated to a single group.⁷ Two tasks are reciprocally interdependent if they mutually affect each other; they must be executed coherently in order to create a correct overall outcome.⁸ If the number of reciprocally interdependent tasks is too large to be managed within a single group, the tasks will be grouped so that task interdependencies within a group are strong and task interdependencies across different groups are weak.⁹ This division reflects the insight that coordinating reciprocally interdependent tasks across group boundaries is more difficult than coordinating these tasks within a group.¹⁰ Reciprocally interdependent tasks cannot be performed independently; they require ongoing mutual adjustment. As a result, people performing these tasks must continually communicate and coordinate their decisions. As empirical studies have shown, communication across group boundaries within an organization occurs less frequently than communication within a group.¹¹ Members of a group communicate personally and frequently on an informal basis (“high bandwidth communication”); the resulting common knowledge base facilitates the communication of new information and the coordination of mutually dependent decisions. Communication across group boundaries tends to be more formalized and is often inhibited by the lack of familiarity with members of other groups and their responsibilities.

The architecture of a system affects the division of responsibility between design teams because decomposing a system into components determines the nature and strength of the interdependencies between the tasks involved in the detailed design of the system.

An architecture implies both a set of tasks and interdependencies between them.¹² An architecture contains an inherent set of design problems which have to be solved to create a workable design. Thus, each design problem implies the task of solving it. If design problems are interdependent, their solutions must be coordinated to ensure a coherent outcome. As a result, an interdependency between design problems implies an interdependency between the corresponding design tasks.

To limit complexity, any product architecture places strongly interconnected design problems within the same component. As a result, interdependencies between design problems inherent in the design of a single component are stronger than those between design problems shared among different components. Whereas the tasks involved in designing a single component are strongly interdependent, there are fewer interdependencies with design choices in different components. Consequently, component boundaries implicitly separate strongly interdependent tasks from less interdependent ones. That, in turn, leads rational organizations, which group tasks so that task interdependencies within a group are strong and task interdependencies across different groups are weak, to use them as boundaries between design teams.

How Architecture Affects Firms' Boundary Choices

Apart from implying the boundaries between the different design teams involved in the design of the system, a system's architecture constrains the number and the types of organizational structures that can feasibly develop the system. In particular, features of the architecture determine whether the various components have to be developed by a single firm or whether they can be developed by independent firms. In the terminology of economics, the product architecture determines a firm's feasible vertical boundaries.

The vertical boundaries of a firm separate the activities it performs itself from those it purchases from independent firms in the market. To define its vertical boundaries, a firm must decide whether to organize transactions internally or across a market. The corresponding decisions are called *make-or-buy decisions*.¹³

Economic theory has long investigated how firms decide whether to internalize certain transactions or to purchase them in the market. Initially, different strands of economic theory focused on different factors—for example, on transaction costs or on capabilities, which were viewed as mutually exclusive explanations for the vertical boundaries of firms. As theoretical and empirical analyses have shown, it is more appropriate to

view these theories as complementary approaches that jointly explain the vertical boundaries of firms.¹⁴ According to this view, firms and markets constitute alternative means of economic organization (so-called governance structures), with distinct advantages and disadvantages. To define its vertical boundaries, a firm must decide whether to internalize a transaction and perform the activity itself, or to use the market and purchase it from an independent firm. Owing to the different characteristics of firms and markets, the costs and benefits of organizing the transaction may vary depending on governance structures. Economic theory assumes that firms base this decision on the comparative costs and benefits associated with the transaction under both governance structures.¹⁵ As this chapter will show, product architecture affects the feasibility of different organizational choices by influencing the relative costs of using the market.

The analysis in this chapter focuses on two types of costs that economic research has identified as relevant to a firm's vertical boundary choices: coordination costs and transaction costs (box 5.1). Whereas transaction costs result from the needs to economize on bounded rationality and to safeguard against opportunism, coordination costs arise from the need to ensure consistent treatment of interdependent transactions. According to economic theory, the firm is the superior governance mode for dealing with bounded rationality and opportunism and for ensuring the coordination of interdependent transactions. As a result, economic theory assumes that firms will choose to internalize transactions if transaction costs or coordination costs are high and are not offset by larger benefits of using the market.

Coordination costs and transaction costs are not the only factors affecting the boundary choices of firms. As was noted above, firms base their decision on a joint evaluation of the different costs and benefits associated with the transaction under the different governance structures. Thus, a comparative advantage of internal organization with respect to transaction costs may be offset by a larger comparative advantage of using the market, for example because the market may provide access to superior capabilities or enable firms to take advantage of economies of scale.¹⁶ As a result, an analysis based on coordination-cost and transaction-cost considerations alone is necessarily incomplete and cannot predict the actual choices of specific firms.¹⁷ It does, however, provide an indication of the feasibility of different organizational choices with respect to these costs: If the firm is the superior governance mode with respect to transaction and coordination costs, the analysis of these costs indicates how large countervailing advantages of the market would have to be to eliminate this superiority. If internal organization and arm's-length relationships are equally feasible

Box 5.1

Coordination Costs and Transaction Costs

Whereas transaction costs have long been used to explain the vertical boundaries of firms, coordination costs have only recently been identified as a distinct cost category affecting firms' governance choices.^a

Historically, the two types of costs have been explored by different strands of economic research.^b Transaction costs have been analyzed by economics and strategy research, particularly by transaction-cost economics, property-rights theory, and agency theory. Coordination costs have been examined by organizational research such as structure contingencies theories, organizational learning theory, and parts of the literature on product development in engineering science.

Conceptually, transaction-cost reasoning and coordination-cost reasoning are complementary, distinct approaches to the analysis of governance choices. Transaction costs and coordination costs describe different aspects of a relationship between economic actors and represent different concerns influencing the governance choices of firms.

Transaction costs capture the costs involved in the contractual governance of transactions, such as the costs of drafting, negotiating, monitoring, and enforcing contracts in the presence of bounded rationality and opportunism. According to transaction-cost reasoning, governance choices are driven by appropriation concerns. Coordination costs focus on the procedural aspect of the transaction (that is, the coordination of interdependent activities through the day-to-day interactions of the employees involved in executing the transaction); this approach traces firms' boundary choices to the desire to ensure efficient coordination among interdependent activities.

The theoretical differences between the two approaches are also illustrated by the differing roles played by opportunism.^c Opportunism—which is defined as “self-interest seeking with guile,” and which includes lying, cheating, and subtler forms of deceit, such as violating agreements^d—is one of the main assumptions on which transaction-cost reasoning is based; without it, transaction-cost differences between different governance structures would disappear. In contrast, coordination-cost differences arise even in the absence of opportunism. The magnitude of these differences, however, may be intensified by opportunism. Thus, coordination-cost analysis does not depend on opportunism in the way that transaction-cost theory does.

Since coordination costs and transaction costs arise at different places in the organization, they can be separated empirically.^e Whereas responsibility for the contractual governance of a relationship is usually located in the purchasing department, the day-to-day communication and decision making that are necessary to coordinate interdependent activities take place among employees involved in the execution of the transaction.

Box 5.1

(continued)

- a. The argument that coordination costs are a distinct cost category affecting firms' governance choices that cannot be explained by transaction-cost reasoning and therefore must be studied separately has been advanced by Gulati and Singh (1998), Gulati (1998), Sobrero and Schrader (1998), Sobrero and Roberts (2002), Jacobides (2002), Gulati, Lawrence, and Puranam (2005), and Jaspers and van den Ende (2006).
- b. See Sobrero and Schrader 1998, pp. 587–595.
- c. See Gulati and Singh 1998, pp. 782–783; Jacobides 2002, pp. 15–18.
- d. Williamson 1985, pp. 47–49.
- e. See Sobrero and Schrader 1998, p. 586.

from a transaction-cost and coordination-cost perspective, other factors will determine firms' boundary choices. For example, firms and markets differ in their incentive intensity.¹⁸ Whereas firms in an arm's-length transaction directly appropriate the gains of the transaction, the allocation of gains to recipients in an internal transaction is usually less precise and subject to potential manipulation. As a result, in an arm's-length transaction, actions and consequences are often more tightly linked, resulting in higher incentive intensity and, consequently, higher efficiency of market-based transactions. In contrast, internal organization is characterized by lower incentive intensity and added bureaucratic costs. Other things being equal, this shifts firms' preferences toward market-based organization. Firm's governance choices may also be influenced by specific capabilities¹⁹ within the firm or in the market or by the ability to exploit economies of scale.²⁰ Firms may also base their decisions on such strategic considerations as the desire to maintain or create strategically important capabilities "in house" or the desire to maintain exclusive control over critically important or particularly profitable components.²¹

In the following analysis, we will look at how coordination and transaction costs affect firm's boundary choices and how different product architectures influence these costs. We will focus on a firm's ability to insert an organizational boundary between the definition of the architecture and detailed component design by organizing the detailed design of particular components in an arm's-length relationship (figure 5.1). This interface is particularly relevant to the present book, since detailed component design is an important locus of innovation. If arm's-length relationships across this interface are possible, economic actors other than the system's

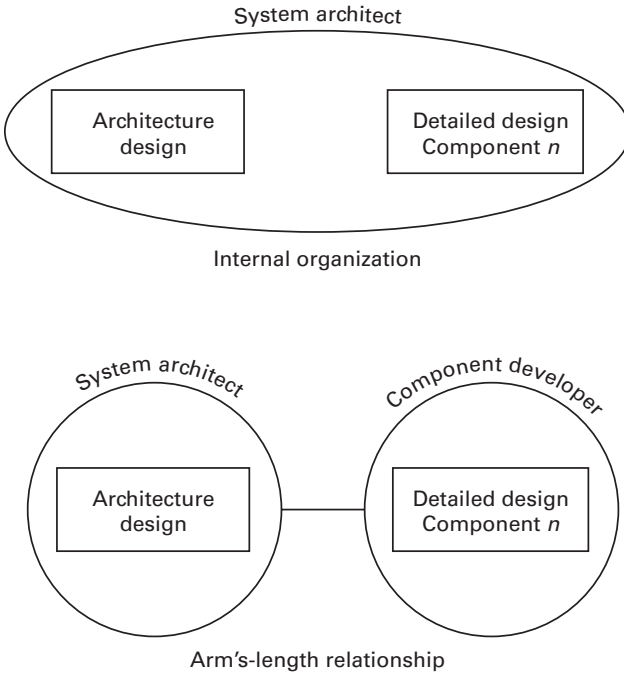


Figure 5.1
Options for organizing detailed design.

architects may be able to develop new or better components, affecting the amount and the type of innovation that may occur.

The analysis assumes that the detailed design and the production of a component are performed by the same firm. In reality, an organizational separation of detailed design and production is quite common. Whether such a separation is feasible can be determined by applying the analysis outlined below to the interface between detailed component design and production.²²

How Architecture Affects Coordination Costs Coordination costs are the costs of coordinating interdependent tasks or sets of tasks. Two tasks are interdependent if the execution of one task may affect the execution of the other in a way that influences the correctness of the overall outcome. In this case, their execution must be coordinated to ensure a correct outcome.

Coordination costs have two elements: the costs of communicating relevant information and the costs of ensuring that the task in question is executed coherently.²³

Coordination-cost reasoning is based on the insight that firms possess distinct characteristics with respect to communication and decision making that enable them to coordinate certain types of interdependent tasks at lower costs within the firm than in arm's-length relationships across firm boundaries.²⁴ For these tasks, firms will internalize both tasks unless their relative advantage with respect to coordination costs is offset by countervailing advantages of using the market.

The relative advantage of firms over markets depends on the amount of communication and decision making necessary to ensure that the interdependent tasks in question are executed coherently. Thus, the coordination requirements of two particular tasks determine the coordination-cost differences between internal and external organization, which, in turn, influence the feasibility of the different governance choices. Thus, coordination-cost theory links an observable task attribute—the type of task interdependency—to feasible governance choices.²⁵

Coordination Advantages of Firms over Markets Firms may have advantages over markets with respect to communication and decision making. Firms possess certain characteristics that make it easier to transfer information within a firm than across firm boundaries. The effect of these characteristics may differ depending on the type of information that needs to be transferred: whereas the costs of communicating tacit information are usually lower within a firm than across firm boundaries, the two governance structures may be equally efficient in communicating certain types of codified information.

Tacit information is information that is informal, unstructured, uncodified, and stored in people's heads.²⁶ Since it is closely linked to an individual's specific knowledge and experience, it cannot be written down or shared easily. Instead, transferring tacit information usually requires "rich"²⁷ interactive communication mechanisms, such as face-to-face communication. Rich communication mechanisms are characterized by their capacity to provide feedback, transmit multiple cues, and use natural language. As a result, they are capable of reducing the inherent ambiguity that makes tacit information difficult to transfer.²⁸

In general, the costs of communicating tacit information within a firm are lower than the costs of communicating it across firm boundaries.²⁹ Rich communication mechanisms are more readily available within a firm. Face-to-face communication is easier to achieve within a firm. Members of a single firm are more likely to be co-located and better able to identify suitable communication partners; membership in the same organization

creates a sense of commonality that facilitates establishing new contacts.³⁰ Should co-location be desired, firms can simply mandate co-location without any of the lengthy and costly negotiations that are common in arm's-length relationships.

In addition, members of the same organization usually develop common languages and communication codes that reduce the potential for misunderstanding and the amount of information needed to explain something; this firm-specific standardization of information further reduces the costs of communicating tacit (and other) information.³¹

Codified information is written down and can be transferred easily. Whether firms can transfer this information internally at lower costs than across firm boundaries depends on the information in question. On the one hand, the existence of firm-specific language and communication codes may lower the costs of communicating within the firm. On the other hand, efforts toward industry-wide standardization of information and advances in information technology are reducing the advantages of firms with respect to codified information. Efforts toward industry-wide standardization of information remove potential ambiguity. By eliminating the need for rich, interactive communication media, they make it easier to transfer this information across firm boundaries.³²

Electronic information systems reduce the costs of transferring large amounts of complex information. In the beginning, however, this cost reduction was restricted to information transfers within firms, because most firms originally adopted proprietary, non-interoperable systems. In recent years, information systems that provide standardized ways of exchanging and interpreting data across firm boundaries have become increasingly common. The widespread adoption of interoperable information systems such as CAD/CAM and systems that implement the electronic data interchange (EDI) standard increases the range of codified information that can efficiently be communicated across firm boundaries.³³

Thus, if codified information is standardized or well documented to reduce ambiguity, and if there are suitable mechanisms for transferring the information across firm boundaries, the relative advantage of firms with respect to communication is reduced considerably, if not eliminated.

Firms may also have advantages with respect to decision making. Owing to differences in governance structures, firms can make decisions regarding interdependent problems at greater speed, at lower costs, and potentially with better quality. In comparison with an arm's-length relationship, firms benefit from the existence of command structures and authority systems,

from the availability of administrative dispute-resolution mechanisms, and from superior access to information.

In an arm's-length relationship, the two parties remain independent entities. There are no command structures or authority systems that let one party impose a decision on the other. Instead, coordinated decision making is brought about by negotiating and convincing, a lengthy and costly process. This process may be further complicated by the existence of divergent goals among multiple parties. Until the issue is resolved, the parties also bear the indirect costs of maladaptation. In contrast, firms already have or can easily create appropriate command structures and authority systems that enable hierarchical decision making. Within the firm, an employee enters into an employment relationship in which she agrees to work at the direction and discretion of a superior within general bonds. Thus, coordinated decision making within the firm can be mandated.³⁴

If the parties in an arm's-length transaction cannot reach an agreement, the dispute probably will be resolved by a court. In contrast, courts usually forbear from dealing with disputes within a firm. As a result, such disputes can be resolved within the firm. The firm has access to flexible administrative dispute-resolution mechanisms, such as managerial fiat or informal mediation, that make it possible to resolve conflicts faster and at lower cost.³⁵

When tasks are organized internally, managers within the firm may have access to superior information that provides a better basis for decision making. Whereas an employee is subject to comprehensive disclosure requirements, the parties in an arm's-length transaction are protected by the general contractual non-disclosure clause, which restricts each party's right to demand information unless this right was explicitly specified.³⁶ In addition, the transfer of relevant information in an arm's-length transaction is often impeded by concerns over information leakage. Since they cannot be sure whether the other firm will protect the confidentiality of the information, firms tend to minimize the amount of information they communicate across firm boundaries.³⁷ Finally, if interdependent transactions are organized internally, it may be easier to recognize and take account of any complementarities and externalities among transactions.

Linking Task Interdependency to Governance Choices Based on the differences between firms and markets with respect to communication and decision making, the subsequent analysis develops a framework that links an observable task attribute—the type of interdependency between two tasks—to feasible governance choices.

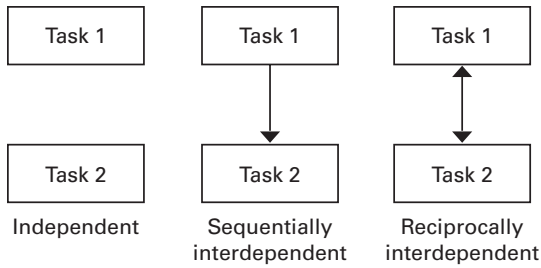


Figure 5.2
Task interdependencies.

Following James Thompson,³⁸ we can distinguish three types of task interdependency that differ in the amount of communication and in the mechanisms needed to achieve coordination: independence, sequential interdependence, and reciprocal interdependence (figure 5.2). As a result, the type of task interdependency directly influences the coordination costs under the two governance structures (internal organization and arm's-length transaction).

Two tasks are independent if they do not affect each other.³⁹ They are sequentially interdependent if the output of the first task is an input into the second task, but the reverse is not true. Thus, the execution of the first task influences the range of feasible approaches to the second task.⁴⁰ Two tasks are reciprocally interdependent if they mutually affect each other; they must be executed coherently in order to create a correct overall outcome.⁴¹

In the order introduced above, the three types of task interdependencies place increasingly heavy burdens on communication and on decision making. As the coordination-cost differences between internal and external organization grow, it becomes increasingly infeasible to organize interdependent tasks across firm boundaries in arm's-length relationships.

The execution of two independent tasks does not require any communication or decision making between those performing them. Thus, task independence creates no coordination costs and does not influence the corresponding governance choices.

To ensure the overall coherence of sequentially interdependent tasks, those performing the second task must comply with the constraints imposed by the execution of the first task. Thus, apart from transferring the relevant information or material from the first task to the second, no communication between those performing the tasks is required. Since the

hierarchy between the tasks is clear, coordinating sequentially interdependent tasks does not require any additional decision making. Instead, the mechanism for ensuring a coherent execution consists of determining whether the execution of the second task complied with the constraints.⁴²

Organizing sequentially interdependent tasks within and across firm boundaries is equally feasible if internal organization does not provide any coordination-cost advantages. This is the case if it is equally costly to communicate the relevant information and to ensure a coherent execution within a firm and across firm boundaries. As was set out above, information can be transferred as efficiently across firm boundaries as within a firm if it is codified, if it is unambiguous, and if efficient transfer mechanisms exist. As will be set out in detail below, the correct execution of a task can be as efficiently monitored in an arm's-length relationship as within a firm if technology is available to measure whether the attributes of the product or service conform to the requirements.

If these two conditions do not hold (i.e., communicating the relevant information or ensuring a coherent execution of the task is less costly within the firm than across firm boundaries), coordinating the tasks internally is less costly than coordinating them across firm boundaries. Unless the relative advantage with respect to coordination costs is offset by countervailing advantages of using the market, a firm will internalize both tasks.

Thus, the effect of sequential interdependence on governance choices depends on the codifiability and transferability of the relevant information as well as on the existence of tests for compliance.

Reciprocally interdependent tasks call for coordination by mutual adjustment.⁴³ Often the circumstances affecting the execution of each task cannot be predicted in advance, and most of the relevant information will be discovered during the execution of each task. As a result, it is not possible to agree on a coherent approach in advance. Instead, those performing the tasks must communicate on an ongoing basis to let each other assess the effect of potential actions on the other task and identify feasible approaches. They must communicate information as it becomes known (a typical case of tacit information) in order to develop common understandings of problems and to design potential solutions. All these tasks require rich communication.⁴⁴

If two tasks are reciprocally interdependent, there may be several pairs of potential actions that are mutually consistent. In this case, each pair of potential actions constitutes a feasible approach; those performing the

tasks must decide jointly which pair to adopt. One approach may be optimal with respect to the first task; another may be preferable with respect to the second. As a result, those performing the tasks may disagree on the joint approach that should be adopted. In this case, hierarchical decision making will be necessary to choose the solution that best meets the overall requirements.⁴⁵

As has been described above, it is less costly to communicate tacit information within a firm than across firm boundaries because rich communication is less costly to achieve within a firm. A firm can also reach decisions regarding interdependent problems at greater speed, with lower costs, and potentially with higher quality. As a result, coordinating reciprocally interdependent tasks within a firm is considerably less costly than doing it across firm boundaries. Owing to the high cost, coordinating reciprocally interdependent tasks across an arm's-length relationship is usually not feasible. Unless the relative advantage with respect to coordination costs is offset by countervailing advantages of using the market, a firm will internalize both tasks.⁴⁶

Internal organization is not the only feasible organizational structure for coordinating two reciprocally interdependent tasks. Other governance structures that constitute intermediate forms of economic organization on the spectrum from firms to markets may provide an alternative to internal organization if the coordination costs of using an arm's-length relationship are too high. For example, two firms can choose to establish a common communication framework and to set up procedures for hierarchical decision making and dispute resolution through a joint venture or (less formally) a joint research project.⁴⁷ The resulting coordination framework may reduce or eliminate the differences in coordination costs between the intra-firm and inter-firm organization of reciprocally interdependent tasks. The result is a quasi-integrated structure that deviates significantly from the typical governance structure of an arm's-length transaction. Negotiating and implementing such a coordination framework is, however, quite costly. Thus, a firm will choose this solution only if the expected gains outweigh the overall costs, as may be the case if the cooperation provides access to capabilities not available within the firm.⁴⁸

Alternatively, a firm may benefit from past investments in a suitable inter-firm coordination framework—for example, in a repeated relationship. Again, such a framework may significantly lower or even make negligible the differences in coordination costs. For example, Japanese automobile manufacturers maintain long-term relationships with major suppliers that are characterized by frequent face-to-face interaction,

co-location of important personnel, and familiarity with the manufacturer's language and procedures. As a result, the costs of coordinating interdependent activities with these suppliers are much lower than in the United States, where most automobile manufacturers deal with their suppliers on a short-term, arm's-length basis.⁴⁹

Thus, though it is not feasible to organize two reciprocally interdependent tasks across an arm's-length relationship (owing to high coordination costs), it may still be feasible to organize them across firm boundaries, using an alternative governance structure.

Empirical Evidence Several empirical studies have explored particular aspects of the relationship between coordination costs and governance choices. If these studies are interpreted in the light of the framework presented above, the results correspond with the predictions of the framework. For example, coordination costs can explain empirically why semiconductor firms choose to internalize design and production if the tasks involved are reciprocally interdependent, whereas production can be outsourced if the relationship between design and production is characterized by sequential interdependency.⁵⁰ A case study of the mortgage banking industry highlights how advances in the industry-wide standardization of mortgage descriptions and the adoption of interoperable information systems eliminated the coordination-cost advantages of internal organization with respect to sequentially interdependent activities, enabling the formation of intermediate markets.⁵¹ According to an empirical study of firms' governance choices for inter-firm alliances, differences in the type of task interdependencies can explain differences in the choice of hierarchical controls in inter-firm alliances.⁵²

The Effect of Architecture From a coordination-cost perspective, product architectures influence the feasibility of various vertical-boundary choices by determining the nature and strength of task interdependencies among component-design teams. As we saw earlier, an interdependency between two design parameters implies an interdependency between the tasks of choosing them. Since component boundaries commonly delineate boundaries between design teams, the design-parameter interdependencies between components determine the task interdependencies between the teams responsible for designing them.

Modular and integrated product architectures differ in the nature and the strength of design-parameter interdependencies between components; the resulting differences in task interdependencies explain why the

feasibility of organizational choices varies between modular and integrated architectures.

Though an integrated architecture identifies interfaces between components, interfaces are complex and not well specified.⁵³ This is because inter-component interdependencies are not resolved during the definition of the architecture, but during detailed design. In fact, “cycling” through parameter interdependencies within and across components to create a globally optimal solution is a crucial part of detailed design. Thus, providing a complete interface specification that resolves all cross-component interdependencies is not possible before the end of detailed design.

An integrated architecture is characterized by complex, unresolved design-parameter interdependencies between components. Since design parameters mutually affect one another without a precedence relationship between them, the corresponding design tasks are reciprocally interdependent across component design teams. As a result, the coordination costs of organizing all detailed component design within a firm are considerably lower than the coordination costs of distributing the design of the different components among independent firms. Unless significant gains are expected from outsourcing the detailed design of specific components or existing inter-firm coordination frameworks can be utilized, the costs of inter-firm coordination are usually prohibitive.

With an integrated architecture, organizing the detailed design of a component in a typical arm’s-length relationship will not be feasible; under the conditions outlined above, a “hybrid” governance structure such as the creation of an inter-firm coordination framework, for example in the form of a joint venture or joint research project, or the use of an existing one may constitute a feasible alternative to internalization.⁵⁴

Characteristics of the architecture also affect how subsequent innovation can be organized. In a system based on an integrated architecture, the various components are linked by numerous interdependencies that have been resolved during detailed design. Due to these far-reaching chains of dependencies within the system, it is usually not possible to change parts of a component without triggering changes in the rest of the system. As a result, isolated innovation at the component level is usually not possible.⁵⁵ Instead, innovation is a system-level activity that is very similar to designing a new system from scratch. Owing to the reciprocal interdependence between parameters in different components that are revisited in the course of this process, both the redesign of the architecture and the detailed design of the different components consist of reciprocally interdependent

tasks that are best organized internally or in a closely integrated alternative governance structure.

In a modular architecture, all cross-component design-parameter interdependencies are resolved during architectural design and documented in detailed interface specifications that are not allowed to change during detailed design. As a result, no unresolved interdependencies between components exist when detailed design starts. Since design decisions within a component are hidden from designers of other components, cross-component interdependencies cannot be reintroduced during detailed design. Design decisions within a component, however, are bound by the visible information defined during architectural design.⁵⁶

Thus, a modular architecture is characterized by two types of design-parameter interdependencies. First, hidden design parameters within a component are sequentially dependent on the visible design parameters contained in the relevant interface specifications. Second, a component's hidden design parameters are independent of the hidden design parameters of any other component.

Consequently, a modular architecture creates two types of task interdependencies (figure 5.3). First, the tasks inherent in the detailed design of each component are sequentially dependent on the tasks required for architectural design. To coordinate these activities, the architecture's visible

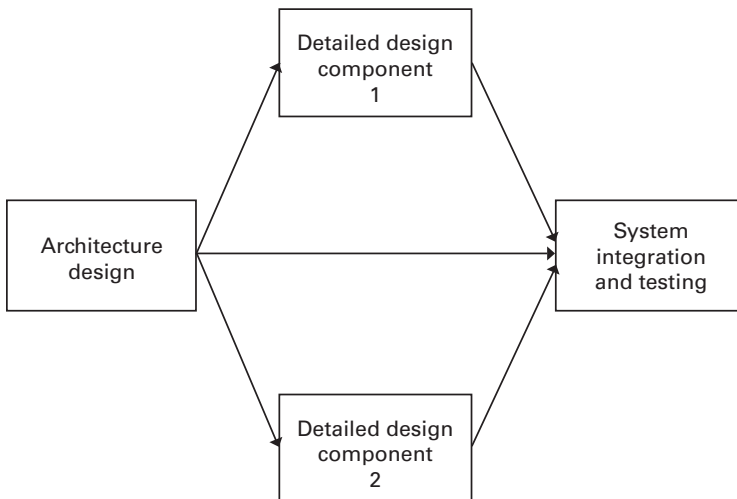


Figure 5.3

Task interdependencies in a mature modular architecture.

information necessary for the design of a particular module must be communicated from the system's architects to the designers of the module. To ensure that the module will function properly within the overall system, the finished component's design must be checked to ensure that it conforms to the relevant interface specifications. Second, the detailed design processes of different components are independent. As a result, there is no need for interaction between the designers of different modules.⁵⁷

Detailed component design can be organized through arm's-length transactions, with different firms responsible for the different components, if (1) the architecture completely captures and resolves all existing cross-component interdependencies, (2) the architecture's visible information is completely specified and well documented to avoid ambiguity, and (3) technology is available to determine whether a module complies with the relevant interface specifications.⁵⁸ The first condition ensures that the detailed design of different components is really independent; the second and the third capture the conditions under which the coordination of sequentially interdependent activities is equally efficient within and across firm boundaries. The second condition eliminates the communication advantages of firms; the third removes the advantage of internal organization in monitoring and enforcing performance. If these conditions hold, a module can be traded in an arm's-length transaction, and each module can be designed by a different firm.

Thus, from a coordination-cost perspective, internal organization and arm's-length relationships constitute equally efficient ways of organizing the detailed component design of a modular system, if the three conditions above are met. Owing to the independence between the detailed design processes of different components, the vertical-boundary choices for the different components are independent of each other; as a result, each module can be designed by a different firm. Whether a firm uses this opportunity and outsources the detailed design of some or all components depends on additional factors not related to coordination costs. For example, a firm may wish to keep certain strategically important components in house to maintain exclusive control of critically important or highly profitable components and may choose to outsource others to leverage the benefits of competitive component markets for innovation and pricing.⁵⁹

Thus, a modular architecture does not determine firms' boundary choices for detailed design, nor does it necessarily result in a modular organizational structure with different loosely coupled organizations developing the different components of the system. It merely gives the firm

controlling the architecture the ability to choose whether to organize the detailed design of specific components internally or in arm's-length relationships without having to worry about coordination costs.

Finally, the tasks of system integration and testing are sequentially dependent on architecture definition as well as on detailed component design.⁶⁰ If the architecture completely captures and resolves all existing cross-component interdependencies and the individual components comply with the interface specifications, no changes to components are required during system integration and testing to ensure that the system functions. In this case, system integration simply involves putting the different components together. Thus, system integrators must know the architecture of the system and must have access to the detailed designs of the components. Apart from that, no interaction among system architects, component designers, and system integrators is required. If the task is not too difficult and does not require technology that isn't available to users, users may be able to perform the task of integrating their own system.⁶¹

The above analysis of task interdependencies applies only if the architecture completely captures and resolves all existing cross-component interdependencies. Owing to the bounded cognitive abilities of the system's architects, architects working on a new or a relatively young architecture are often unable to capture all dependencies when first defining the architecture. This is a common problem in practice.⁶² In this case, resolving unforeseen cross-component interdependencies may require considerable changes in components or in the architecture's visible information during detailed design or during system integration and testing if the system is to function. Thus, if the system's architects have not recognized and resolved all cross-component interdependencies, architectural design, detailed component design, system integration, and testing will be reciprocally interdependent, as will be the detailed design of different components (figure 5.4).⁶³

The organization of subsequent innovation in a modular system depends on the type of innovation. Whereas innovation at the component level can proceed independently as long as the architecture's existing visible information is unaffected, innovation at the architectural level requires close coordination. In economic terminology, innovation at the component level constitutes an *autonomous* innovation that can be pursued independently from other innovations. Innovation at the architectural level is *systemic* innovation whose benefits can be realized only in conjunction with related, complementary innovations.⁶⁴

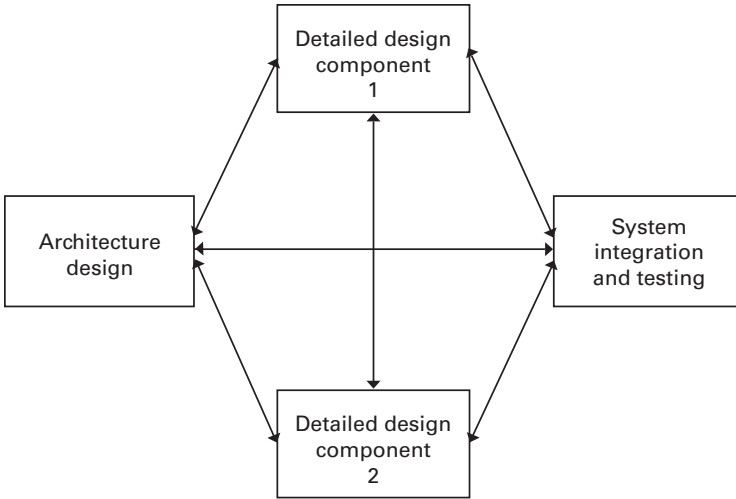


Figure 5.4 Task interdependencies in a modular architecture with unresolved design-parameter interdependencies among components.

As we saw in chapters 2 and 4, changes at the component level that do not affect the architecture’s existing visible information (e.g., improving an existing component or adding a new component that uses existing interfaces) do not require any changes in the rest of the system.⁶⁵ As a result, innovation of this type can proceed independently.⁶⁶ Thus, internal organization and arm’s-length relationships are equally feasible organizational arrangements.

This independence has important implications for the locus of control over component-level innovation. Since the architecture of the system remains unaffected, no technical contribution by the system’s architects is required to enable this type of innovation. If the architecture captures and resolves all existing cross-component interdependencies and the new or improved component complies with the constraints imposed by the architecture, the component will function smoothly within the overall system. In this case, system integration is reduced to putting the different components together and can potentially be performed by anyone who has access to the necessary technology. As a result, innovators do not have to coordinate their activities with the system’s architects or integrators; they do not have to ask for permission to be able to develop an innovation, and, as long as the architecture remains unchanged, the system’s architects have no technical way of preventing innovators from acting. Thus, relative

to an integrated architecture, a modular architecture shifts control of component-level innovation from the system's architects to component designers.⁶⁷ As we will see in chapter 8, this decentralization of control is particularly important if there is uncertainty about the technical or commercial viability of the innovation, since the need to convince another actor decreases the chance that an innovation will be realized.

In contrast, an innovation that involves changing the architecture's visible information (i.e., changes to the interfaces) requires altering different parts of the system. The innovative process consists of two steps. First, those controlling the architecture must agree on the necessary changes and must produce specifications for the refined architecture. This step is similar to the initial design of an architecture. In the new interface specifications, the system's architects must capture and resolve all cross-component interdependencies. Thus, this step consists of reciprocally interconnected tasks and is best organized within a single organization. If the architecture is owned by several firms or is defined in open standards, the economic actors cooperating in the evolution of the architecture will most likely establish an alternative governance structure, perhaps a consortium or a standard-setting body, to coordinate these tasks.⁶⁸ Second, module designers must adapt their component design to the new architecture; that is, they must produce a refined detailed design that complies with the new visible information. Owing to the lack of unresolved cross-component interdependencies in a modular architecture, the designers of various modules can perform this task on their own. As with detailed component design in general, the redesign of the various components can be organized equally well internally or in arm's-length relationships, with different firms designing different components.

How Architecture Affects Transaction Costs An architecture also affects transaction costs. Transaction costs are the costs associated with drafting, negotiating, monitoring, and enforcing contracts in the presence of bounded rationality and opportunism. In addition to the direct costs of managing the relationship, transaction costs include the possible opportunity costs of making inferior governance decisions.⁶⁹

Transaction-cost theory is based on the insight that firms have distinct characteristics that enable them to mitigate problems caused by bounded rationality and opportunistic behavior at lower costs within the firm than in an arm's-length relationship. Thus, if a transaction is afflicted with these problems, the transaction costs associated with internal organization will be lower than the transaction costs of using the market. In this case, and

unless the comparative advantage with respect to transaction costs is offset by countervailing advantages of using the market, a firm will internalize the transaction. Whether bounded rationality and opportunism pose problems for a specific transaction can be traced back to specific attributes of the transaction under consideration.

Thus, transaction-cost analysis links attributes of transactions to feasible governance choices. The subsequent analysis does not take account of differences in incentive intensity between internal organization and arm's-length relationships. Though this factor is often included in traditional transaction-cost analysis,⁷⁰ I treat the differential incentive intensity of both governance structures separately from transaction-cost considerations, along with other factors that may influence firms' boundary choices.

Behavioral Assumptions: Bounded Rationality and Opportunism Transaction-cost theory is based on the assumption that human beings are characterized by bounded rationality and opportunism, an assumption that deviates from the behavioral assumptions of neoclassical economics.⁷¹ Bounded rationality defines human beings as "intendedly rational, but only limitedly so."⁷² Thus, it refers to the limited information-processing and communication capacities that constrain humans' ability to act rationally.⁷³ Opportunism, defined as "self-interest seeking with guile," includes lying and cheating as well as subtler forms of deceit, such as violating agreements.⁷⁴

Linking Attributes of Transactions to Governance Choices Bounded rationality and opportunism pose problems for a transaction if the circumstances of the exchange are uncertain, if it is necessary for one or both of the parties to invest in relationship-specific assets that are less valuable outside of the relationship, or if performance is difficult to measure. If one of these conditions applies, transaction costs will be lower under internal organization. In the absence of uncertainty, asset specificity, and measurement problems, the two governance structures are equally able to manage bounded rationality and opportunism.

Thus, from a transaction-cost perspective, uncertainty, asset specificity, and difficulties in measuring performance are the critical influences on firms' governance choices. This subsection outlines the reasoning that underlies this claim.

The standard way for two parties to organize an arm's-length transaction is to write a contract specifying what is expected from each. In the

absence of uncertainty, measurement problems, and asset specificity, the parties can write a complete, enforceable contract that efficiently safeguards against opportunism.⁷⁵ Since the circumstances surrounding the exchange are clear, bounded rationality is not a problem: the parties can foresee and resolve all relevant contingencies in advance and write a contract that completely specifies each party's rights and responsibilities. They can do so in unambiguous terms that can be applied by a third party enforcing the contract. And since it is possible to measure performance, both parties, and even a third party, can verify contract compliance. As a result, each party can detect opportunistic behavior by monitoring performance and, should it be necessary, can enforce the agreement by going to court. Finally, since neither party has invested in relationship-specific assets, it is possible for the parties to switch trading partners when the contract is up for renewal if opportunistic behavior has occurred. This acts as an additional deterrent to opportunistic behavior.

As will be described below, firms have access to mechanisms that reduce the incentives for opportunistic behavior and enable them to detect opportunistic actions should they occur. As a result, internal organization is an equally efficient way to safeguard against opportunism.

Thus, in the absence of uncertainty, measurement problems, and asset specificity, internal and external organization are equally feasible from a transaction-cost perspective. Under these circumstances, other considerations—such as the different incentive intensity of firms and markets, the desire to gain access to capabilities that may exist within the firm or in the market, or the desire to exploit economies of scale—will determine firms' boundary choices.⁷⁶

The higher the degree of uncertainty and complexity, the more difficult and costly it is to write a complete contract to organize an arm's-length transaction. Owing to the parties' bounded rationality, contracts will necessarily be incomplete. As a result, the contract may have to be renegotiated should unforeseen circumstances occur. As we saw above, this may be a lengthy and costly process. Since the parties to the transaction may try to exploit any existing incompleteness to their own benefit, reaching agreement on how to adapt the contract may be impeded by opportunistic behavior. Disputes will be resolved by litigation, which further delays adaptation. Thus, in comparison with simple transactions without uncertainty, the transaction costs associated with uncertainty and complexity in an arm's-length transaction not only include higher costs of drafting and negotiating the initial agreement; they also include the costs of communicating new information, renegotiating or litigating agreements, or

coordinating activities to adapt to new circumstances. In addition, parties who fail to adapt will incur the opportunity cost of maladaptation.⁷⁷

The problems caused by incomplete contracts are particularly severe if the relationship is supported by specific assets that are less valuable outside the transaction in question.⁷⁸ In particular, the party who has invested in specific assets is in danger of being “held up” by its partner: Since the firm cannot costlessly redeploy the specific assets in other ways, its partner has an incentive to appropriate some of the gains from the specialized investment through post-contractual bargaining or threats of termination. This incentive is particularly high when contracts are highly incomplete, so that proving breach of contract is difficult.

Thus, asset specificity may raise the transaction costs of arm’s-length market exchange in two ways. First, the chance of “hold-up” may lead to more frequent renegotiations as one party tries to change the terms of contract to his own benefit. Apart from increasing costs, the negotiations are likely to disrupt the execution of the contract, leading to productivity losses. If the party succeeds in changing the contract, the other party will not be able to appropriate the expected gains. Second, the fear of being “held up” in the future may lead a party to underinvest in relationship-specific assets *ex ante*, resulting in productivity losses.

A firm is able to deal with the problems caused by uncertainty and asset specificity at lower costs. As was described above, sequential adaptation to problems as they unfold is easier within a firm because it has command structures and authority systems, can use administrative dispute-resolution mechanisms, and can better access relevant information. As a result, firms can adapt to unforeseen problems and resolve disputes more quickly, at lower cost, and potentially with superior outcomes.⁷⁹

In addition, a firm is better able to deal with opportunistic behavior. For one thing, a firm can reduce the incentives to behave opportunistically.⁸⁰ While the interests of two parties in an arm’s-length transaction are not necessarily aligned, sociological influences, such as organizational cultures, may create convergent goals between members of the same organization, enhancing the willingness to act in a cooperative way.⁸¹ And a firm can reduce the payoffs from opportunistic behavior by linking long-term rewards such as promotion prospects to cooperative behavior.⁸² On the other hand, a firm has access to auditing, monitoring, and control mechanisms that enable it to detect opportunistic behavior should it occur.⁸³ As we saw above, in an arm’s-length transaction access to information is usually quite limited. In addition, though the parties in an arm’s-length transaction have the right to test the quality of the transaction’s

result, a standard contract does not confer the right to monitor how the other party executes the contract within its own organization. In contrast, employees of a firm are subject to extensive information disclosure requirements.⁸⁴ Moreover, managers within a firm have access to various monitoring mechanisms that let them closely control how transactions are executed. For example, contrary to parties in an arm's-length transaction, managers can monitor behavior as well as outputs, or can measure the quality of inputs. As a result, behavior is more observable within a firm, making it easier to detect opportunistic actions. Thus, the transaction costs associated with uncertainty and asset specificity are lower under internal organization.

Transaction costs are also influenced by measurement problems. As principal-agent theory has shown, difficulties in measuring performance damage the performance of markets and firms.⁸⁵ Under both governance structures, imprecise measurement constrains the incentive intensity of rewards, which limits performance. Incentives are most efficient when rewards are directly linked to performance. Such incentives encourage actors to work harder to meet performance goals. If performance cannot be measured accurately, it cannot be correctly linked to rewards; since the economic actor does not bear the full costs of his or her actions, the rate of productive effort will be lower and shirking more common than if accurate measurement were possible.⁸⁶

Thus, if it is impossible or difficult to measure performance in an arm's-length relationship, it is impossible or difficult to verify whether a party has complied with the agreement. As a result, the party will receive the same compensation whether it shirks or not. Because of this, some economic actors will invest too little effort or diligence when executing such a contract. This problem is particularly profound in an arm's-length relationship if relevant attributes of the object of the transaction cannot be measured: since one party in an arm's-length relationships does not have the right to monitor how the other party executes the contract internally, the two parties cannot use alternative measuring mechanisms to assess performance. As a result, they have to endure the worse performance that results from not being able to accurately measure and reward performance.

Firms can address measurement problems at lower costs than parties in arm's-length relationships. As we saw above, firms have access to superior auditing and monitoring mechanisms that let them closely control how tasks are executed internally. For example, if measuring the quality of the output becomes difficult, managers can use behavioral monitoring to

better assess performance.⁸⁷ In addition, internal organization may attenuate the incentives for opportunistic behavior.⁸⁸

Thus, in the presence of uncertainty, asset specificity, or measurement problems, the transaction costs associated with internal organization are considerably lower than the transaction costs associated with an arm's-length relationship. As a result, it will rarely be feasible to organize a transaction afflicted with these problems in an arm's length relationship.

While the transaction costs of using an arm's-length relationship to organize a transaction that is subject to uncertainty, asset specificity, or measurement problems will usually be prohibitive, firms may still be able to organize such a transaction across firm boundaries. Firms may be able to reduce the transaction costs associated with executing such transactions by setting up alternative governance structures that better mitigate the problems resulting from bounded rationality or opportunism.⁸⁹ For example, to manage uncertainty, firms may agree on specific mechanisms for adaptation and conflict resolution that let them, if adaptation is required, make decisions and resolve disputes faster. To avoid the risk of expropriation *ex post*, firms facing the risk of "hold-up" may invest in contractual safeguards during the initial contract negotiations, or they may make other investments to improve their post-contract bargaining position. To address the risk of shirking, parties can agree on auditing and monitoring methods that help them better measure performance. For example, the parties can replace output measurements with more costly and complicated measurement methods, such as monitoring the behavior of the other party's employees. Although these solutions lower the transaction costs associated with executing the transaction, they require considerable set-up costs, such as drafting and negotiating the original agreement and implementing dispute-resolution or monitoring mechanisms.⁹⁰ As a result, a firm will use alternative governance structures only if the benefits of partnering with another firm justify the total transaction costs. Further, firms may be able to organize such transactions across firm boundaries if they benefit from relationship-specific factors that reduce transaction costs. For example, research has shown that trust can considerably reduce transaction costs by reducing the incentives for opportunistic behavior and the perceived need for safeguards.⁹¹ Other firms may benefit from past investments in alternative governance structures.⁹²

The Effect of Architecture From a transaction-cost perspective, product architectures influence how detailed component design can be organized by determining the nature of the tasks that are the object of the transac-

tion. As we saw above, the boundaries among components determine the boundaries among the design teams involved in the detailed design. As a result, the nature of the component specification determines the nature of the tasks design teams are supposed to perform. The nature of the tasks, in turn, determines the attributes of the transaction.

Detailed component design under modular and integrated architectures differs with respect to the existence of uncertainty, asset specificity, and measurement problems.⁹³

Under an integrated architecture, detailed component design is characterized by uncertainty, asset specificity, and measurement problems. The object of the transaction (i.e., the design of a component) cannot be specified in advance, since important characteristics of the component (e.g., the exact form of its interfaces) have yet to be determined. In addition, the process necessary to arrive at a suitable design is inherently uncertain: the detailed design of a particular component involves resolving internal design problems and cross-component interdependencies in ways that let the component function within the overall system. In the course of this process, chains of interdependencies through the system must be recognized and resolved. Finding a high-quality solution requires many trade-offs across components, and determining how to best make these trade-offs is an interactive, iterative process. The amount of iteration required depends on contingencies that cannot be planned for. Thus, both the detailed design process and its result are afflicted with considerable uncertainty and cannot be specified in advance. As a result, a contract will be highly incomplete, leaving considerable room for opportunistic behavior.

Detailed component design in an integrated architecture also requires designers to invest in specific assets. To recognize and resolve cross-component interdependencies, designers of a particular component must interact closely with other design teams. If the design of this component is organized in an arm's-length transaction, the other design teams will belong to a different firm. Efficient coordination across firm boundaries requires considerable investment in learning the responsibilities, language, and routines of the other firm.⁹⁴ In addition, to be able to assess the effect of their design decisions, designers must acquire knowledge about other components. Thus, to successfully execute the transaction, designers must acquire skills, know-how, and information that are specific to the particular trading partner and system architecture and therefore more valuable inside this relationship than outside it—a typical case of human asset specificity.⁹⁵ Thus, an independent firm that designs a component in an integrated architecture faces the risk of “hold-up” by its partner.

Finally, detailed component design in an integrated architecture suffers from measurement problems. Since the parties cannot contractually specify the desired characteristics of a particular detailed component design in advance, they cannot measure performance by simply comparing specifications and results. Though it is possible to determine whether the component design correctly interacts with the rest of the system, it is difficult to assess the quality of the design. If the component has problems interacting with the rest of the system, the many interdependencies among components make it difficult to determine what causes these problems and which party is responsible for them. As a result, it is difficult to measure how well the transaction was executed. At the same time, because organizations rarely have the right to monitor across firm boundaries, it is not possible to determine whether the other party worked hard enough to acquire relevant transaction-specific information and skills and to solve the actual design problems.⁹⁶ Thus, under an integrated architecture, it is difficult, if not impossible, to assess the quality of the product and the quality of the service. This results in lower incentive intensity and, potentially, lower performance.

Owing to the uncertainty, the asset specificity, and the measurement problems associated with detailed component design under an integrated architecture, organizing the detailed design of individual components in an arm's-length transaction is not feasible. A firm may nevertheless choose an alternative governance structure if the benefits of using another firm are high enough to justify the total costs.⁹⁷

As we saw above, innovating on an existing integrated architecture is very similar to developing a new system from scratch. As a result, organizing subsequent innovation is subject to the same constraints as initially developing a system.

Under a modular architecture, detailed component design is not subject to uncertainty or measurement difficulty. The degree of asset specificity depends on additional factors.

A modular architecture completely specifies the relevant characteristics of a component, both in terms of expected behavior and in terms of the interfaces governing its interaction with the rest of the system. Module designers must develop a component that complies with these constraints. Since the architecture completely captures and resolves all existing cross-component interdependencies and is not allowed to change during detailed design, the constraints imposed by the architecture are stable and well defined. In addition, module design is independent of the detailed design of other components; this removes another potential source of

uncertainty.⁹⁸ As a result, it is possible to specify the object of the transaction (i.e., the design for a module) with certainty by referring to the module's architectural specification.

Under a modular architecture, parties can assess minimum contractual performance by checking whether a component's design conforms to the component's interface specifications. A component that does not meet the interface specifications cannot function with the rest of the system. Since the design of each module is independent of the design of other modules, responsibility for a component's inability to correctly interact with the rest of the system can be clearly assigned to the designers of the module under consideration. Determining the quality of the design requires additional knowledge and technology. If metrics exist that summarize the performance of a module and tests are available that can capture these metrics, performance can be measured by using these tests to measure the quality of the design. And since it is possible to measure the quality of the output, it is not necessary to determine the quality of the service. Whereas it is more difficult to monitor the quality of a service across firm boundaries, applying a test to the object of a transaction is equally feasible within and across firm boundaries. Thus, if appropriate metrics and tests are available, a modular architecture does not present any measurement problems. This requirement is usually met after an architecture has existed for some time.⁹⁹

Whether designers must invest in relationship-specific assets depends on the architecture. Since the designers of different modules do not have to interact with one another, they do not have to learn about other firms' rules and procedures, or about the internals of other modules. They only have to acquire knowledge about the architecture and the relevant interface specifications. Whether this knowledge can be used outside of the particular relationship depends on the architecture. If the architecture constitutes a standard (open or proprietary) that is followed by a number of firms, knowledge of the architecture probably will be valuable in other relationships too. If a firm is being "held up," it probably will be able to find an alternative trading partner. Thus, for standards-based architectures, asset specificity is low. Asset specificity is particularly low in the case of an open standard, whose evolution no single firm can control.¹⁰⁰ In contrast, if the architecture is proprietary and the relevant architectural information has been disclosed to the firm in an exclusive relationship, asset specificity is higher. Since the idiosyncratic knowledge is restricted to knowledge about the architecture, asset specificity is still considerably lower than under an integrated architecture.

In sum, whether the detailed design of a particular module can be organized in an arm's-length transaction depends on whether metrics and tests for the performance of that module exist and on the degree of asset specificity. Thus, relative to coordination-cost reasoning, a transaction-cost perspective introduces an additional criterion (asset specificity) that influences the feasibility of arm's-length transactions in modular architectures.

A modular architecture also affects how subsequent innovation can be organized. If the architect envisions a particular function for a module and is able to specify its visible behavior and relevant interfaces, the corresponding detailed design process is very similar to the initial designing of a module.

A modular architecture also allows for component-level innovation that the architect may not have foreseen. For example, as long as the architecture's existing visible information is unaffected, new components with completely new functionalities can be added using the services of existing modules. In this case, an important part of the innovative process is coming up with new and creative ways of enriching the existing system. As a result, the parties cannot specify the object of the transaction in sufficient detail *ex ante* to enforce the contract in court *ex post*. In addition, whether the innovator will actually find a useful new component is highly uncertain, which makes this task difficult to contract upon. Not being able to specify the object of the transaction in advance also creates measurement problems similar to those encountered in an integrated architecture. Moreover, metrics and tests for an object yet unknown will not be available. The degree of asset specificity depends on the factors discussed above. Thus, owing to the high level of uncertainty and the existence of measurement problems, organizing this type of innovation in a typical arm's-length transaction will usually not be feasible.¹⁰¹

Nevertheless, internal organization is not the only possible governance structure for this type of innovation. This is because innovation at the component level that does not require changing an architecture's existing visible information can proceed independently. The innovator requires neither permission nor any form of cooperation from the system's architect. Thus, she does not have to reach a contractual agreement with the architect before starting to innovate. Once the innovative process has been successful, a considerable source of uncertainty is reduced. If intellectual-property rights safeguard the innovator against appropriation of her innovation, she can disclose the idea and sell it, or she can contract for the corresponding detailed design in an arm's-length transaction.

Alternatively, she can develop (and produce) the complete component and trade the finished product in an arm's-length transaction.¹⁰²

Implications of the Link between Architecture and Feasible Governance Structures

As this chapter has shown, product architectures constrain how the initial development and production of system components can be organized as well as how subsequent innovation might occur by enabling or disabling arm's-length transactions. The constraints imposed by the architecture have implications for the organizational options available to individual firms, for the potential structure of the industries for the development and production of system components, and for the ability of independent economic actors other than the system's architect to innovate.

Effect on Firms' Organizational Options

From the perspectives of transaction costs and coordination costs, detailed component design under an integrated architecture must take place within a vertically integrated firm or a closely integrated alternative governance structure. Organizing the detailed design of individual components in an arm's-length relationship is not feasible. Since gaining access to other firms' capabilities requires closely integrated alternative governance structures and usually comes at considerable cost, the firm's strategic flexibility with respect to organization is severely limited.

Under a modular architecture, organizing the detailed design of the different components internally and in arm's-length relationships is equally feasible if the architecture completely captures and resolves all existing cross-component interdependencies, if the architecture's visible information is completely specified and well documented to avoid ambiguity, if technology is available to determine the relative quality of a module, and if asset specificity is not too high. If these conditions are met, a modular architecture lets a firm exploit the benefits of using the market by outsourcing the detailed design of one or more components in arm's-length relationships—an important source of strategic flexibility. Whether a firm uses this opportunity depends on additional factors, two of which are the existence of differential capabilities and the desire to maintain exclusive control over critically important or highly profitable components.¹⁰³ Thus, a modular architecture enables, but does not determine, organizational choices.¹⁰⁴

Effect on the Structure of the Industries for the Development and Production of Components

The effect of product architecture goes beyond constraining the organizational options available to individual firms. By enabling or disabling arm's-length relationships, product architectures also constrain the potential structure of the industries that design and produce the components of a system.

If integrated architectures prevail, an industry will be populated by vertically integrated firms, or by closely integrated networks of firms, each of which produces its own proprietary system. In such an industry, competition takes place among different proprietary systems.

Although modular architectures can support the same industry structure, they tend to facilitate vertical disintegration by enabling a firm to outsource the detailed design of one or more components in arm's-length relationships. Moreover, the feasibility of arm's-length relationships is an important prerequisite for forming markets.¹⁰⁵ As a result, a specific modular architecture can provide the basis for a "modular cluster"¹⁰⁶—that is, a horizontally stratified industry in which firms in different sub-industries design and produce components compatible with the architecture. In a modular cluster, firms in one sub-industry specialize in providing a particular module or set of modules, and markets arise at the interfaces among different sub-industries. In such an industry structure, firms operating in different sub-industries are complementors,¹⁰⁷ whereas firms in the same sub-industry are competitors. Thus, competition shifts from competition between proprietary systems to competition between components within the same architecture.

In addition to independent component developers, a new type of company, a so-called system integrator, might emerge to specialize in assembling components into a complete system.¹⁰⁸ If assembly is not too difficult and does not require technology unavailable to individual consumers, users might put together their own systems.¹⁰⁹

Although vertically integrated firms that produce and assemble their own components can operate in such an industry, small specialized firms may be at a competitive advantage: they are driven by the high-powered incentives of market-based organization, they can focus on areas where their superior capabilities give them a competitive advantage, and they may be better able to exploit economies of scale or learning economies.¹¹⁰

The formation of a sub-industry specializing in a particular module requires that many different firms have the architectural information

needed to design compatible components. Without this information, designing a compatible component will not be possible. Firms can learn this information from the original architect or through reverse engineering. Since reverse engineering is usually very difficult and costly, and sometimes illegal,¹¹¹ a system's architect can influence the industry's future structure by deciding whether and to what extent to disclose the relevant information. Thus, the architectural and contractual choices made by a system's architect interact with the existing legal framework to shape the economic environment for innovation.¹¹² By determining the strength of intellectual-property protection for interface specifications, intellectual-property laws can make it more or less difficult for independent third parties to design and produce a system's components if the architect does not voluntarily provide access to the interface information.

The vertical disintegration of an industry into a modular cluster following the emergence of modular architectures has been documented in a variety of industries. For example, this phenomenon has been observed in the stereo-component industry,¹¹³ the disk-drive industry,¹¹⁴ the computer industry,¹¹⁵ the mortgage industry,¹¹⁶ and the microprocessor industry.¹¹⁷ At some stage, modular architectures might become integrated again, for example, to realize performance gains.¹¹⁸ When and how architectures evolve toward more modularity or more integration is an area of active research.¹¹⁹

Effect on Ability to Innovate

By enabling or disabling arm's-length relationships, modular and integrated architectures also affect how easily actors other than the system's architect can innovate, for example by developing new system components.

Innovation on an integrated architecture must occur within a single vertically integrated firm or within a closely integrated governance structure. Thus, although economic actors other than system architects can participate in such an innovative effort, they must closely integrate and coordinate their activities with other participants. They cannot innovate independently.

In a modular architecture, the ability to innovate depends on the type of innovation.

Innovation at the component level that leaves the architecture's existing visible information unchanged (e.g., improving existing components or adding new components that connect through existing interfaces) can be realized by economic actors other than the system's architect if they have access to the architectural information necessary to design a

compatible component. For example, an independent application developer can create a new application for the Windows operating system that uses Windows' existing application programming interfaces if she has access to the architectural specifications of these interfaces.

The ability of independent economic actors to innovate at the component level has important implications for the type of actors who can innovate as well as for the nature of the innovation that can occur.

First, developing a single component does not require a great amount of investment. Whereas small firms may not be able to raise enough capital to develop a complete system, they may be able to raise enough for component-level innovation. In particular, new firms may be able to obtain financial support from venture capitalists.¹²⁰ Thus, a modular architecture probably increases the range of potential innovators at the component level.¹²¹

Second, as we saw above, a modular architecture shifts control over component-level innovation from the system's architect to developers of components. They can innovate independently as long as they abide by the architecture's visible information. As a result, whereas under an integrated architecture the system's architect completely controls a system's evolution, a modular architecture enables decentralized innovation at the component level.¹²² As we will see in chapter 8, if there is uncertainty, this increases the chances that potential innovators will be able to realize their innovative ideas, since they do not have to convince any other decision maker.

Innovations requiring changes to the architecture's existing visible information (e.g., changes to the interfaces) occur in two steps. First, those controlling the architecture must agree on the necessary changes and produce the specifications for the refined architecture. Second, module designers must adapt the design of their component so that the component works with the new architecture; that is, they must produce a refined detailed design and, ultimately, a component that complies with the new visible information.

Who can participate in this type of systemic innovation and how difficult it is to realize depends on who owns the architecture and how detailed component design is organized.

A vertically integrated firm that owns the architecture and designs all components itself can mandate the adoption of the new architecture and efficiently coordinate the necessary activities internally. Thus, it can realize such an innovation on its own. Originally, IBM was in this situation.¹²³ Apple often follows this approach.

If a firm controls the architecture but depends on independent firms to provide compatible components, it can change the architecture on its own, but it must induce independent component providers to produce components to the new specifications. For example, Microsoft owns and can unilaterally change the application programming interfaces of the Windows operating system, but it depends (at least somewhat) on independent software vendors to produce software that takes advantage of new specifications. Getting independent component designers to adopt new interface specifications is not easy.¹²⁴ Adapting a component to new interface specifications is an innovation in its own right. In a platform system in which different firms design, produce, and sell the various components, however, incentives to innovate will usually be too low. First, owing to the complementarity between components, innovation in one component has a beneficial effect on the demand for the other components that is not appropriated by the innovator. Because of this externality, it is usually assumed that component providers' incentives to innovate are too low if different firms design, produce, and sell the various components.¹²⁵ The same externality affects decisions by producers of complementary components to adopt new interfaces. Second, additional factors may further reduce independent component developers' incentives to innovate—for example, a platform monopolist may vertically integrate into a complementary market in which multiple independent component makers operate.¹²⁶ Thus, it is usually assumed that the architect of a system must exert considerable effort to induce independent component designers to adopt new architectural specifications.

If ownership of the architecture is distributed among different firms, or if the architecture is an open standard, firms must cooperate to change the architecture. For example, ownership of the current architecture of the personal computer is distributed among a variety of firms and standards organizations: Microsoft owns the application programming interfaces to its operating systems; Intel owns the interfaces to its processors, and so on. Often, changes in processor interfaces must be complemented by changes in the operating system's application programming interfaces before improvements in the processor can make a difference for the user. Coordinating such architectural changes across firms is much more difficult than coordinating them within a single firm. The distributed ownership of architecture may give rise to serious coordination problems, strategic problems, and collective-action problems. First, as we saw above, the costs of coordinating reciprocally interdependent tasks across firm boundaries

are significant, especially those necessary for gaining agreement on new architectural specifications across firm boundaries. Second, strategic behavior by some or all of the firms involved may further complicate the decision making.¹²⁷ Finally, there may be collective-action problems if no firm feels responsible for evolving the architecture, even though all firms would benefit from the evolution. (This problem plagued the personal-computer industry in the early 1990s.¹²⁸) To overcome the coordination problem at the architectural level, firms often form separate consortia or standards-setting bodies, which then coordinate and specify architectural changes.¹²⁹ As in the other scenarios where firms other than the system's architects develop components, the firms owning the architecture also must motivate unaffiliated component designers to adapt their components to the new specifications. For example, to realize the benefits of changes to processors or operating systems for users, independent software vendors or independent hardware producers must change their products to accord with the new specifications.

The greater the number of parties involved in the design and production of a system, the more difficult it is to realize systemic innovations. Thus, those controlling the architecture face a trade-off between the ability to benefit from third parties' creativity and capabilities and the ease with which systemic innovations can be realized.

In sum, even in a modular architecture, independent economic actors other than the owner of the system's architecture cannot realize a systemic innovation on their own. They may, though, be able to influence the evolution of the architecture by participating in the corresponding innovative efforts—for example, by working with the system's architect or by participating in standard setting. They can also participate in systemic innovation by developing and producing components according to the new specifications if they have access to the necessary architectural information. As was noted above, this activity is an autonomous innovation that can proceed independently.

The link between architecture and organization provides system architects with a number of strategic choices. In particular, they can design their architecture strategically to invite or preclude third-party innovation. If they choose a modular architecture instead of an integrated one, they can allow third-party innovators to innovate on some or all components (though opening the system to third-party innovation makes it more difficult to change the architecture's visible information).¹³⁰

If they have chosen a modular architecture, system architects can influence how easy it is for independent economic actors to innovate by

strategically distributing functionality to components. For example, if functions that contribute to a specific feature are distributed across multiple components, innovating on this feature will require systemic innovation. In contrast, if functions are allocated to a single component, innovation will be autonomous. Similarly, by incorporating specific functionality in a larger component, a system's architect makes it impossible to specialize in providing that functionality alone. Instead, firms must provide the complete component. For example, in luxury cars, cell-phone functionality could be integrated with the audio system, or could be allocated to a separate subsystem. A luxury car's audio system is usually developed by the car maker. Owing to the strong degree of integration, the first design would most likely require that the complete audio-and-cell-phone system be developed in house, eliminating the opportunity to involve independent economic actors. In contrast, under the second design, independent economic actors could develop the cell-phone subsystem on their own.¹³¹

Finally, companies competing within an existing modular cluster can change the nature of the competition by making a modular architecture more integrated. If one competitor starts offering a product based on an integrated architecture, this may force competitors to offer integrated systems too.¹³²

Architecture and Organization in the Original Architecture of the Internet

The above insights into the links between architecture and organization in modular and integrated architectures can be used to understand the effect of the Internet's original architecture on firms' organizational options, on potential industry structures for developing and producing components, and on independent actors' ability to innovate.

Effect on Firms' Organizational Options

The Internet is based on a modular architecture. Thus, from the perspective of transaction costs and coordination costs, organizing detailed design (and production) of different components internally and in arm's-length relationships is equally feasible if the architecture completely captures and resolves all existing cross-component interdependencies, the architecture's visible information is completely specified and well documented to avoid ambiguity, technology is available to determine a module's relative quality, and asset specificity is not too high.

The architecture of the Internet meets these requirements.

First, it has a mature, modular structure that captures and resolves all existing cross-component interdependencies in interface specifications.¹³³

Second, ownership and maintenance of the Internet architecture's visible information are spread over a number of organizations.¹³⁴ For example, the protocols that make up the TCP/IP protocol suite have been standardized by the Internet Engineering Task Force in open standards. The IETF's standard-setting process is open to all participants, and the standards documents are available to all at no cost.¹³⁵ The IETF standards for a particular protocol often specify only the services provided by the protocol and the peer interface, i.e., the format and meaning of messages between two components of the particular protocol. To leave sufficient flexibility for protocol implementors, the exact form of the service interface, i.e., the protocol's interface to higher-layer protocols, is not specified by the standard. Instead, the exact form of the service interface is determined by the protocol implementor.¹³⁶ For example, since IP, TCP and UDP are implemented in the operating system, the service interface of these protocols is specified by the operating system vendor. Thus, whereas the services and the peer interfaces of these protocols are defined in open standards, the corresponding service interfaces are often part of a proprietary standard maintained by its owner.

Although the ownership and the nature of the standards may differ, the services and interfaces of the Internet's core components are completely specified, well-documented, and publicly available. For example, vendors of operating systems usually publish the specification of the service interfaces to IP, TCP, and UDP so as to enable third-party application programmers to access the services provided by these protocols.

Third, over time, metrics have been developed that make it possible to compare the quality or performance of different implementations of the same component.

Fourth, many of the relevant standards, including those developed by the IETF and the World Wide Web Consortium, are open standards, and their asset specificity is low. In particular, the core protocols at the Internet layer and the transport layer (IP, TCP, and UDP) have been developed by the IETF and are open standards.¹³⁷

Thus, the architecture of the Internet fulfils the conditions under which arm's-length relationships and vertical integration among providers of different components are equally feasible. As a result, firms' governance choices are driven by factors other than transaction costs or coordination costs—for example, strategic considerations or differential capabilities.

Effect on the Structure of the Industries for the Development and Production of Components

The architecture of the Internet forms the basis of a modular cluster, with firms in different sub-industries developing and producing the different components or subsystems.¹³⁸ The emergence of a modular cluster has been facilitated by the public availability of the relevant interface specifications that give any interested economic actor the information necessary to design and produce compatible components.

In particular, the design of the Internet layer as a technology-independent and application-independent portability layer populated by a single protocol has enabled the formation of separate, independent sub-industries for providing Internet-infrastructure software and hardware and application software.¹³⁹

Owing to the different types of investment required, the sub-industries have very different costs of entry.¹⁴⁰ Developing Internet-infrastructure components usually requires investments in hardware and software. Products that include hardware and software must be manufactured and physically distributed, requiring a considerable up-front investment in physical capital. In contrast, developing components for the application and the transport layer requires only programming knowledge and access to an end host. The resulting software products can be copied and distributed almost costlessly over the Internet. Thus, entry into the corresponding sub-industries requires almost no physical capital and is considerably less costly. As we will see in the next subsection, this cost structure greatly increases the size and diversity of the pool of innovators in these industries.

Owing to the large number of components making up "the Internet system," no one firm can provide all the necessary components itself. Some firms specialize in one or more separate components; others provide a complete subsystem with a set of adjacent components. For example, Microsoft produces operating systems for end hosts, which include implementations of IP, TCP, and UDP and expose the corresponding service interfaces in so-called application programming interfaces.¹⁴¹ Thus, Microsoft's operating systems incorporate components belonging to the Internet layer and to the transport layer. Cisco specializes in routers; Cisco's routers contain the various hardware and software components necessary to operate a router, combining components belonging to the link layer and the Internet layer.¹⁴²

Firms of another kind specialize in assembling components into products that are ready to use. For example, Dell assembles the hardware and software components that make up an end host. Other firms may specialize

in assembling complete physical subnets. Alternatively, users can assemble their own systems by buying and installing the necessary hardware and software. In addition, users can exchange or add components later.¹⁴³

Effect on Ability to Innovate

The Internet's architecture also affects independent actors' abilities to participate in autonomous and systemic innovation.

Autonomous Innovation As we saw above, an architecture influences independent actors' ability to participate in innovation in two ways. First, it affects whether and when they can innovate independently, without requiring cooperation or permission from the system's architect. Second, by influencing the costs of innovation, it affects which type of actor can participate in innovation.

Owing to its modular structure, the Internet's architecture provides many opportunities for autonomous innovation by independent economic actors. Any innovation that does not affect the architecture's existing visible information can be realized independently. As we saw in chapter 4, because of the use of the design principle of relaxed layering with a portability layer and the broad version of the end-to-end arguments in the design of the Internet's architecture, the architecture enables three specific types of autonomous innovation: developing new application-layer or transport-layer protocols, new link-layer protocols or physical network technology, and new implementations of existing components and sub-systems. Innovations belonging to these categories can be deployed incrementally. Owing to architectural differences, the three categories differ considerably in the type of investment required to develop, produce, and distribute the innovation.¹⁴⁴ Thus, by influencing the costs of innovation, the Internet's architecture influences who can engage in these different types of innovations. To see this more clearly, let us focus on how the Internet's architecture affects innovators' ability to develop applications.

The architecture of the Internet influences who can develop applications by allowing application-level innovation to proceed independently and by inducing a particular cost structure: due to the end-to-end design of the Internet, developing components for the application layer requires only access to an end host that is connected to the Internet, programming knowledge, programming tools, and access to the application programming interfaces necessary to let the application send data over the Internet. The resulting software products can be copied and distributed almost costlessly over the Internet.¹⁴⁵ Since many of these resources are often already

available (e.g., many people have a computer or access to one) or can be acquired at relatively low costs, this cost structure lets a wide range of innovators with diverse motivations and funding models develop new applications.¹⁴⁶

In particular, an innovator does not have to be an employee of a firm or have outside funding to realize his idea for an application. Because the biggest investment often is the design and programming of the application itself, potential innovators can develop an application in their free time or as a “side project,” with the opportunity cost of the time spent the most important cost factor.

Under these conditions, an application does not have to produce a profit in the future to cover the costs of developing it. Instead, a wide range of benefits may be sufficient to cover the development costs. Some innovators develop new applications because they love to program. (This seems to be an important motivation of contributors to open-source projects.¹⁴⁷) Others develop a new application to meet their own needs¹⁴⁸; after the fact, they may discover that others may want to use it too. For example, Yahoo grew out of a list of links to research papers in electrical engineering that Jerry Yan and David Phyllo, two graduate students in Stanford University’s Electrical Engineering Department, used to keep track of research papers that were relevant to their PhD theses. Yan and Phyllo then began to add other, non-technical categories that were of interest to them in their free time. The list quickly became popular, and Yan and Phyllo spent more and more time on it. In the fall of 1994, the site, which still ran on their student computer workstations attached to Stanford’s network, got a million page views per day, which translated to more than 100,000 unique visitors. Recognizing the business potential, Yan and Phyllo decided to turn the website into a business and incorporated Yahoo in March 1995.¹⁴⁹

Del.icio.us, the online bookmarking service that enables users to share their bookmarks with others, emerged from an application that Joshua Schachter had developed to keep track of his collection of 20,000 bookmarks while he was working at Morgan Stanley as a quantitative analyst. The original version, which Schachter finished in 2001, could be used only by him, but other people could view the website that showed his bookmarks. In 2003, Schachter began working on a version that other people would be able to use. It went online in December 2003, and by the end of 2004 it had 30,000 users. In early 2005, with the user base growing, Schachter decided to leave Morgan Stanley and focus on Del.icio.us full-time.

Innovators may also be driven by a mix of different motivations. For example, throughout the 1970s and the 1980s, most of the programs used to read and send electronic mail were developed by researchers on the ARPANET. Often they started to develop a new program because they were unhappy with the existing programs. Solving their problems involved researching what they enjoyed. And in the days of the ARPANET, people who developed successful applications were highly regarded, so the desire to improve their reputation may have motivated these researchers as well.¹⁵⁰

The low costs of application innovation under an end-to-end architecture allow innovation based on a variety of funding models. For example, since realizing an application does not require major capital investments, innovators can pursue their idea in their free time. This option is particularly valuable if there is uncertainty about the technical feasibility of, or the user demand for, the application, since innovators can test their ideas risk-free without giving up their day job. For example, Pierre Omidyar developed the online auction website AuctionWeb, which later became eBay, at home over the Labor Day weekend of 1995. At first Omidyar viewed it as a hobby, working on it nights and weekends. Auction Web ran on his personal website, which he operated at home under his existing Internet-service account that cost \$30 per month. Throughout 1995, the website attracted thousands of auctions, and more than ten thousand individual bids. When his Internet Service Provider switched Omidyar's account to a commercial account arguing that AuctionWeb's traffic was slowing down their system, Omidyar started charging sellers a small commission for sales enabled through the website. He operated AuctionWeb as a hobby until June 1996, when monthly revenues surpassed \$10,000.¹⁵¹

Mark Zuckerberg developed the original version of Facebook, an online social network for students at elite universities, over the course of a few months in his dorm room at Harvard College. He launched the site on February 4, 2004. It was an instant success. By the end of February, about three-fourths of Harvard's undergraduates had registered. Over the next few months, Zuckerberg extended the site to 40 other universities. By mid June, 150,000 users had registered. Since Zuckerberg and Dustin Moskovitz (a roommate who had begun to assist him) were working for no pay, their biggest operating costs were the costs of renting server space at \$85 per month. They continued to work on Facebook over the summer in a house they had rented in Silicon Valley, and in September 2004 they decided to focus on Facebook full-time. At the time, the site had 250,000 users.¹⁵²

The low level of investment required for developing and operating an application lets small companies and start-ups fund the development themselves or get the necessary funds from small, less formal financing sources. As we saw in the preceding chapter, even applications with servers operated by the innovator can now be financed in this manner.¹⁵³ Some innovators use personal funds; others get money from family members, friends, or “angel” investors.¹⁵⁴ Jeff Bezos funded the first six months of Amazon’s development himself. His father invested \$100,000 in February 1995. In July 1995, when Amazon.com went live, his mother invested \$145,553. From summer 1995 to the end of that year, Bezos got \$981,000 from 20 private individuals, most of them contributing \$30,000.¹⁵⁵ Catarina Fake and Stuart Butterfield, the founders behind Flickr, financed their company Ludicorp using personal funds, money from friends and family members, a loan from the Canadian government, and, later, some angel investment for Flickr.¹⁵⁶

Others finance the development of applications with income from consulting or from programming jobs, or, at a later stage, with income from existing applications. Evan Williams and Meg Hourihan, the founders behind Blogger, financed their company Pyra Labs for its first year by having one of the three members of the team develop Web applications for Hewlett-Packard.¹⁵⁷ A company called 37signals—a small software company that offers Web-based productivity applications—has used this approach beyond the development of its first product (box 5.2).¹⁵⁸

By freeing potential innovators from depending on outside funding, an architecture based on the broad version of the end-to-end arguments increases the number of innovative projects that might be realized. Many innovators will not be able to get outside funding, even if they want to, because they are not able to convince others of the viability of their idea. Many innovators whose applications later became very successful could not get venture capital or corporate funding when they first tried.¹⁵⁹ For example, early in 1997 the founders of eBay tried, without success, to sell the company to Times Mirror, a media company that owned the *Los Angeles Times*, the *Baltimore Sun*, and some other newspapers. The decision makers at Times Mirror thought that “strangers would never trade with strangers over the Internet in large numbers,” and complained that eBay did not own any assets such as buildings or trucks. In 1997 and 1998, before they decided to commercialize the technology themselves, Sergei Brin and Larry Page tried, without success, to license Google’s search technology to existing search engines or portals; they also were unable to get venture capital at the time. In 1996, when the founders of ICQ tried to

Box 5.2

37signals

37signals, a small, privately held software company, financed the development of its first product—Basecamp, a Web-based project-management application for individuals and small teams—by doing design work for outside clients.^a Over the course of six months, three designers spent about one-third of their time on designing Basecamp. The programming was done by a computer science student in Denmark who worked on the project ten hours per week.^b After Basecamp went live, in February 2004, the four kept working on it along with their other projects. The biggest operating expense was the cost of a server (\$150 a month). After a year, fees collected from users allowed them to focus on Basecamp full-time. Since then, 37signals has developed five more applications following the same model. At the time of this writing, the company is profitable and has no debt. Its revenues have doubled every year since 2004.

a. In 2006, the company took a minority private equity investment from Bezos Expeditions, a personal investment company of Amazon founder Jeff Bezos (Fried 2006). The account in the text is based on Fried 2008a, Hammock 2006, Hedlund 2005, Heinemeier Hansson 2008, Livingston 2007, and Park 2008.

b. He is now a partner in the firm.

get funding for their instant-messaging application (the very first instant-messaging application), venture capitalists were not interested, pointing out that ICQ had no revenues and no business plan.

In addition, many innovators will not be able to get venture capital for more general reasons that go beyond disagreements over the viability of an innovator's specific idea. First, as venture capital funds grow larger, the venture capitalists managing them may become less interested in the small investments typically needed for early seed funding.¹⁶⁰ Second, most projects that a venture capitalist invests in are not successful. Venture capital funds only work because the gains from a few highly successful projects more than offset the losses from the majority of projects. This investment model requires a few projects in a venture capitalist's portfolio to produce enormous profits. As a result, venture capitalists look for projects that have the potential to realize high returns; they are not interested in projects that, although economically viable, produce smaller returns.¹⁶¹ Finally, there may be periods in which venture-capital funding is not available—for example, during economic downturns or times when less money

flows into venture capital funds. For example, when the initial-public-offering market collapsed in 2000, many venture capitalists drastically reduced their investments in new companies and instead used their remaining funds to save their existing companies. In early 2001, even though Blogger—with 86,000 registered users—had become the most popular weblog publishing tool, its founders were unable to raise any funding after the stock-market bubble burst. In January 2001, Evan Williams, who with Meg Hourihan had founded Pyra Labs (the company that developed Blogger) in 1999, laid off the other members of the team and moved the servers running the Blogger software into his apartment. And in the fall of 2008, venture capitalists began to shift their focus to maintaining their existing venture investments again.¹⁶²

Thus, many application developers will not be able to get funding from venture capitalists or corporate investors. Under an architecture based on the broad version, however, innovators are not dependent on venture capital or corporate funding. Since the funds needed to develop a new application are limited, innovators can finance their innovation themselves or through smaller, less formal funding sources such as family members, friends, or angel investors. More potential sources of funding make it more likely that at least one funder will find the application attractive, increasing the chance that the application can be realized. Joe Kraus, a serial entrepreneur who co-founded the search engine and portal Excite in the early 1990s, has put it this way: “A lot more people can raise \$100,000 than [can] raise \$3,000,000. . . . The sources of funding capable of writing \$100,000 checks are a lot more plentiful than those capable of writing \$3,000,000 checks.”¹⁶³ For example, though ICQ and Google were unable to get venture-capital funding at first, they did manage to attract some angel investors.¹⁶⁴

In addition to enabling potential innovators to pursue projects for which they cannot get outside funding, independence from outside funding increases innovators’ organizational options. Some innovators value venture capital for the financial and entrepreneurial support it provides, and for the potential of a very high return, and are willing to cede some control over their start-up ventures to venture capitalists in exchange; other innovators prefer to stay in control of their business, even if this implies lower profits or slower, more organic growth.¹⁶⁵ In a network architecture in which application innovation requires a significant amount of outside funding, innovators do not have this choice.

Innovators’ independence from outside funding also has benefits for investors. Since innovators are able to realize an application on their own, potential investors can wait until the uncertainty surrounding an

application has been resolved before deciding whether to invest. Throughout the history of the Internet, venture capitalists and corporate investors often withheld investing until after an application had attracted a significant number of users. Although this does not guarantee the ultimate financial success of the investment,¹⁶⁶ it reduces the risk of investing in applications that are not technically viable or attractive to users. For example, in June of 1997, when eBay got its first venture-capital funding from Benchmark Capital, it had more than 45,000 auction listings per day, and in that year it was to earn \$4.3 million.¹⁶⁷ In the spring of 1996, when the online bookstore Amazon.com got its first venture-capital funding from Kleiner Perkins Caufield and Byer, Amazon had been operating since July of 1995 and was on track to earn \$5 million revenue in 1996.¹⁶⁸ In the second half of 1998, when Google got its first angel investments, it was already getting more than 10,000 search queries a day; it passed 500,000 searches in early 1999. At the time of its first venture-capital investment, in June of 1999, Google, still in beta mode, had been included in *PC Magazine's* list of the Top 100 Web Sites and Search Engines for 1998; its search traffic was growing at a rate of 50 percent per month.¹⁶⁹ When venture capitalists invested in Napster, in Friendster, in MySpace, in Del.icio.us, and in Facebook, the applications already were popular and had large and growing numbers of users. Similarly, although none of them had venture-capital funding, ICQ, Blogger, Flickr, and Bloglines were popular applications with a growing number of users when they were acquired.¹⁷⁰ Venture capitalists and corporate investors could not have pursued this approach to investing if innovators had required significant funding in order to realize and offer these applications.

Network architectures that deviate from the broad version of the end-to-end arguments have a different effect on independent application developers' ability to innovate. As we saw in chapter 4, developing a new application in a non-end-to-end network that implements application-specific functionality in the network may require changing the network's core, which in turn may require changing higher-layer protocols and applications that depend, directly or indirectly, on the changed feature.¹⁷¹ Thus, in a non-end-to-end network new applications are often *systemic* innovations. Unless a single economic actor owns the network infrastructure's visible information, can change it unilaterally, and can change affected higher-layer components (or doesn't care about them), no single economic actor can realize such an innovation alone. Instead, the developer of an application must reach agreement on the necessary changes to the network's visible information and must convince the developers of affected components to adapt their components to the new specifications. In addition, higher-layer protocols and applications that relied on the changed

network functionality or interfaces must be adapted so they will continue to function over the network, and that requires more coordination.

Thus, an independent economic actor cannot simply develop an application on her own; she must incur considerable coordination costs. In addition, an application developer cannot use the operational network infrastructure to test her application. She needs access to an experimental network in order to implement the necessary changes in the network's core and then test the application in this new environment. If some of the affected functionality is implemented in hardware, the innovator cannot use the network to copy and distribute the innovation. Instead, the new hardware must be manufactured and physically distributed. Thus, developing and (potentially) producing and distributing the innovation is considerably more costly in a non-end-to-end network than in an end-to-end network, and it requires a very different type of investment.

The physical capital required to develop an application will usually prevent non-commercial innovators without additional funding from engaging in this type of innovation. In addition, non-commercial innovators may not have the resources to coordinate development across affected component providers or to reach deployment. Owing to the higher investment required, the level of funds necessary may go beyond what angel investors can provide, forcing innovators to acquire the funds from venture capitalists or corporations. Thus, relative to an end-to-end network, a smaller and significantly different set of innovators may be able to develop applications.

In sum, the architectural differences between architectures based on the broad version of the end-to-end arguments and architectures deviating from it result in two main differences. First, in a network based on the broad version, application developers can innovate independently, without permission or help from the network provider or other component producers. In a non-end-to-end network, application developers usually have to coordinate their innovation with the network provider and, potentially, with other component producers. Second, the set of innovators who can develop applications in an end-to-end network is larger and more diverse. In particular, application developers in an end-to-end network do not necessarily need outside funding. (Some of the benefits associated with this have been discussed above.) How these differences affect the amount and the type of application innovation that will occur under the two architectures will be explored in chapter 8.

We already saw that in the past innovators' ability to realize new applications without (or with only minor) outside funding was important. Throughout the Internet's history, many successful applications, including eBay, Yahoo, Google, and Facebook, were developed by innovators in this category. (Chapter 8 will highlight additional examples.) Many of these

innovators were unable to get funding from venture capitalists or corporate investors when they first tried. Everything else except for the network architecture being equal, they could not have realized their applications under a network architecture that required a larger amount of funding to realize the same application.

Thus, innovators with no or little outside funding really can produce successful innovations; this is not just a theoretical possibility. To make this point, the examples above focused on successful applications by innovators in this category. This does not mean that these are the only types of potential innovators. Innovators whose resources go beyond the minimum set of resources required by the architecture can innovate too, and this contributes to the diversity of the innovator pool. A large and diverse set of innovators, in turn, is very important for innovation under uncertainty, as will be shown in chapter 8.

The chapter's focus on successful innovators with no or little outside funding is not meant to imply that innovators from this category will necessarily be successful. This book's overall argument does not require such a claim. As we will see in chapter 8, new technologies are often afflicted with fundamental uncertainty, so no one can identify successful projects in advance. Under these circumstances, exploring a wide range of options is better than exploring only some. Some will succeed, and some will fail, but trying is the only way to find out. Thus, failure is an integral part of the mechanism that produces successful innovation under uncertainty. Failed innovations are consistent with the book's overall argument; they do not invalidate it.

Finally, although this chapter focuses on the minimum requirements for application innovation established by the Internet's architecture, these are not the only factors influencing innovation. As we saw in chapter 1, the ultimate effect of an architecture on innovation at a specific place and time depends not only on the set of architectural and non-architectural constraints, but also on the characteristics of the actors exposed to the architecture and their existing and expected relationships with other actors. For example, Yahoo and Google started at Stanford University, a prestigious and well-connected institution with many formal and informal connections to Silicon Valley and with an infrastructure that encourages and supports student-led and faculty-led entrepreneurship. Mark Zuckerberg's success was incubated in the student social networks of elite universities (Harvard, Yale, Princeton, Stanford) and then developed more maturely only after Zuckerberg moved to Silicon Valley to get access to the capital and human resources available there. Thus, while these innovators were acting independently, using a small set of resources to create worth-

while innovations, they benefited from the organizations, cultures, and networks in which they were embedded. Other innovators with similar personal resources who faced the same architectural constraints, but who were embedded in different organizations, cultures, or networks, may have been less successful in realizing the same innovation.¹⁷² Whereas these differences may matter under an end-to-end architecture, they would not have been relevant under a network architecture that deviates from the broad version and requires a significant amount of outside funding to realize the same application, since none of these actors had the resources necessary to pursue their innovation under that architecture. Thus, the fact that the Internet's architecture enabled independent innovators with no or little outside funding to realize their ideas for an application was a necessary but not necessarily a sufficient condition for their success.

Systemic Innovation Owing to its systemic nature, innovation that requires changes to the Internet architecture's existing visible information—that is, to the specification of services and interfaces of the various components—cannot be performed by an independent economic actor alone. Realizing and deploying such an innovation involves three coordination problems. First, economic actors in affected sub-industries must coordinate their activities to agree on new specifications of the relevant visible information. For example, to change the specifications of the HTTP protocol that governs the interaction between browsers and Web servers, interested parties must cooperate within the Internet Engineering Task Force to adopt new specifications. Second, component providers in the affected sub-industries must adopt the new specifications and must design and produce new components according to the specifications. For example, organizations that develop browsers and servers (for example, Microsoft, Apple, the Mozilla Foundation, the Apache Foundation) must produce new versions of their programs that incorporate the changes. Third, depending on the position of the affected components in the architectural hierarchy, a large number of users and network operators may have to deploy the new components to enable their use in the operational Internet.¹⁷³ For example, end users must replace the browsers on their PCs, and content providers (such as the *New York Times*) and providers of other Web-based services (such as Flickr) must replace their servers.

Who can participate in the standard-setting part of these innovative efforts depends on who owns the standard and on how the standard is set. In open standards, any interested party can participate in the standard-setting process. This applies, for example, to the core Internet standards

developed and maintained by the Internet Engineering Task Force, and to the Web protocol standards developed by the World Wide Web Consortium. In contrast, industry consortia often limit participation, because of efficiency considerations or for strategic reasons. Finally, a firm that owns a proprietary standard may evolve it independently. For example, Microsoft independently develops the application programming interfaces through which applications access transport-layer protocols such as TCP and UDP.

Once the new specifications are finished, independent economic actors can design and produce components to the new specifications—as long as they have access to the information necessary to design a compatible component. Again, small new firms funded by venture capitalists, and (depending on the component's position in the layered hierarchy) non-commercial innovators, may be able to engage in this activity.

6 Architecture and Competition among Makers of Complementary Components

This chapter focuses on an architectural feature that influences the benefits that component developers can expect from their innovation: the ability of one component to control how other components are executed. By describing how components cooperate to provide the overall functionality of a system, an architecture defines interdependencies between components. As we will see, the nature of these interdependencies at the architectural level affects the economic relationships among the makers of components.

For example, if one component can affect how other components function, the maker of that one component can favor certain versions of the rest of the components. In this case, the architecture tips the balance of power toward the maker of the controlling component, who gains the power to tilt the competitive environment of other components in favor of selected component makers. Thus, by enabling one component to control others, an architecture can affect the range of competitive strategies available to companies.

For-profit innovators usually benefit from an innovation by exploiting it commercially. How much profit they can expect depends on the competitive environment and, if the market is imperfectly competitive, on the economic strategies available to the innovator and the innovator's competitors. Thus, by changing the range of strategies available to component makers (or to those who later use these components), an architecture can influence the benefits that innovators may expect from an innovation.

Suppose that a system's architecture enables the owner of one component to block, or degrade the performance of, complementary components from another innovator. An independent innovator with an idea for a complementary component may anticipate that her new component might be discriminated against. This would lower her profits, and it might render an otherwise profitable innovation unprofitable. Expecting

discrimination in the future, she does not innovate today. In contrast, an architecture that provides no opportunity to discriminate against components provides a level playing field. Since innovators will not have to account for a potential profit reduction caused by discrimination, they can expect higher profits than under the “discriminatory” architecture. Thus, more innovations by independent innovators will be profitable under an architecture that precludes discrimination. Conversely, the owner of a controlling component may have a greater incentive to innovate under a discriminatory architecture, since he may be able to realize higher profits by blocking or degrading the performance of complementary components.¹

Consider the example of Skype. Widespread adoption of Internet-telephony applications reduces network providers’ profits from conventional telephone service. If network providers had been able to prevent their users from using Internet-telephony applications in 2002, when Niklas Zennström and Janus Friis were thinking about building an Internet-telephony application based on a peer-to-peer architecture, the innovators might have dropped the idea, fearing that network providers would block their application. But because Internet service providers were not able to detect Internet-telephony applications in 2002,² Zennström and Friis did not have to take into account a possible profit reduction due to discrimination or exclusion.

The line of reasoning presented so far assumes that the owner of a controlling component *will* use the discriminatory power conferred by the architecture to discriminate against other components—or, in economic terms, that he will have an incentive to discriminate against them. If the owner of the controlling component has no incentive to use the architecture’s discriminatory functionality, component developers will not be affected by the difference in architecture.

This chapter explores the effect of the ability to control components in the context of the Internet’s original architecture; the underlying theoretical framework, however, is general enough to apply to other architectures. The first section explains how architectural differences between the original architecture of the Internet and a hypothetical architecture that deviates from the broad version of the end-to-end arguments alter network providers’ strategic options. The second section explores the conditions under which a network provider might have an incentive to discriminate against some applications. The third section explores the effect of architectural differences on network providers’ pricing strategies.

Architectural Differences—Application Awareness and Application Control

To explore how the broad version of the end-to-end arguments affects network providers' strategic options, I will compare the original end-to-end architecture of the Internet with the architecture of a hypothetical discriminatory network. This discriminatory network has two characteristics. First, it is application-aware: it can distinguish between different applications that are using the network. Moreover, it can determine not only the general type of application (e.g., e-mail or the World Wide Web), but also the specific application (e.g., Firefox versus Internet Explorer) and the content (e.g., the website of the *New York Times* versus that of the *Wall Street Journal*). Second, the network can control the execution of applications. For example, the network provider can discriminate against, or exclude altogether, specific applications or content offerings (e.g., blocking Firefox but not Internet Explorer, or limiting the speed at which content from the *New York Times* website downloads). As we saw in chapter 2, a network with these characteristics violates the broad version of the end-to-end arguments.³

Apart from that, the discriminatory network provides the same general functionality as the Internet; it enables computers attached to distinct, interconnected physical networks to communicate. Users will not notice the difference unless the network provider begins to block or slow down the application they are using.

In the original architecture of the Internet, the network is application-blind and unable to control the execution of applications. The Internet protocol receives data packets and transports them to their destination without differentiating between applications. The Internet protocol knows nothing about the application whose data it carries,⁴ and the treatment of data packets at the network's routers is independent of the application that sent the data.⁵ Thus, as long as the network provider respects the layering principle and does not look at the application information within a packet, the network cannot discriminate against applications; it sees only data packets traveling to a particular destination.

For network providers, the ability to control applications creates a set of strategic options that differs from that available under the Internet's original architecture. First, a discriminatory network lets the network provider discriminate against specific applications or exclude them altogether. Second, the owner of a discriminatory network can set different Internet transport prices for different applications (e.g., charge more for an e-mail

packet than for a packet of Web content) or exclude applications to price discriminate between customers (e.g., allow the use of video conferencing only for users of its premium Internet service, not for users of its basic service). Third, under a discriminatory network, the owner can set an access charge for providers of complementary products. None of these three options are available in an end-to-end network.

Today's Internet architecture partly follows this latter "discriminatory" design. Whereas the original Internet was application-blind and unable to control the execution of particular applications, technologies such as deep packet inspection now enable Internet service providers and the providers of physical networks to distinguish between, and control, applications that use their networks.⁶ However, the Internet does *not* currently enable network providers to set fine-grained, application-specific prices or sophisticated access charges.⁷ Thus, today's Internet enables network providers to employ all the discriminatory strategies described in the next section but only some of the pricing strategies described in the final section.

Effect of Ability to Discriminate or Exclude

By enabling a network provider to discriminate against specific applications, a discriminatory network enables a network provider to engage in noncooperative strategic behavior. That is, a network provider may maximize its profits by improving its position relative to potential rivals rather than working with them to maximize overall profits and functionality. This behavior generally increases the profits of one firm while reducing the profits of competing firms.⁸ In contrast, in the original end-to-end architecture of the Internet, the Internet layer shields innovators at the transport layer and the application layer from discriminatory or exclusionary actions by actors operating at the Internet layer or the link layer.

Of course, the architectural differences affect independent application developers' decisions to innovate only if a network provider actually makes use (or independent application developers believe it will make use) of the opportunity to discriminate against complementors⁹—in economic terms, if the ability to discriminate is accompanied by an incentive to discriminate.

The effect of the difference in architecture may be moderated by the structure of the market for the controlling component (in our case, the market for Internet services) or by the laws that govern exclusionary and discriminatory conduct by the provider of the controlling component. For example, a network provider may not have an incentive to discriminate if

it competes with other Internet service providers. And even if it has an incentive to discriminate, existing rules and regulations may still prevent the network provider from acting on this incentive.¹⁰

In view of the interaction between architecture, regulations, and market structure, it is not surprising that network providers' deployment of discriminatory technology has triggered a heated debate over whether their ability to discriminate should be regulated. Though the debate has broadened over time (box 6.1), a rule that would prevent network providers from blocking independent applications, content, or portals is central to every regulatory or legislative proposal supporting "network neutrality."¹¹ Whether network providers have an incentive to use the discriminatory functionality available to them is highly relevant to this debate. Proponents of regulation base their calls for regulatory intervention on the threat of discriminatory or exclusionary behavior by broadband network operators. According to them, regulation is needed to mitigate that threat. If, however, a network operator has no incentive to discriminate against independent portals, content, or applications, regulation is not necessary.¹²

Most commentators believe that the threat of discrimination against independent providers of complementary products can be mitigated by competition in the market for Internet services.¹³ They assume that the market will restrict a network operator's ability and incentive to discriminate. Their rationale is that competition has a disciplining effect. For example, if Comcast's Internet-service customers want to access YouTube, but Comcast blocks YouTube, they will switch to another Internet service provider (ISP) that lets them use YouTube. Anticipating this, Comcast, wanting to keep its customers, will not block YouTube. This assumption about how market forces will operate is the basis for two common policy proposals. The first proposal assumes that fostering competition between operators of different physical networks will mitigate a network provider's ability and incentive to discriminate.¹⁴ This proposal would increase competition for Comcast by encouraging entry by other providers of physical networks—for example, by broadband wireless access providers, or by a local electricity company offering broadband service over power lines. The second proposal seeks to restore competition at the ISP level by requiring the owners of broadband networks to allow independent ISPs to offer their services over these networks. This proposal would increase competition for Comcast by giving independent ISPs, such as Earthlink, a right to offer their services over Comcast's network. Depending on the commentator's point of view, this regulatory response is called *open access*, *multiple access*, or *forced access*.¹⁵

Box 6.1

What Is Network Neutrality?

Every current network-neutrality proposal includes a rule that would prevent network providers from blocking specific applications or content.^a But beyond this, proposals for network neutrality differ in several ways. For one, should the rule ban only blocking, or both blocking and discrimination? If the rule does prohibit discrimination, what does it mean to not “discriminate” against an application? Must a network provider treat every packet identically? Or should providers be allowed to treat packets belonging to different classes of applications differently as long as they treat applications with similar requirements alike (e.g., a network provider could treat e-mail packets different from Internet-telephony packets, but could not treat Vonage packets differently from Skype packets)? A proposal’s approach to this question determines whether, and in what form, network providers would be able to offer Quality of Service under a network-neutrality regime.^b Next, if a network-neutrality regime lets network providers offer Quality of Service (or other forms of better transport, such as a higher bandwidth), who should the network providers be allowed to charge for it? There are four potential answers, each supported by at least some proponents of network neutrality: the network provider (1) can offer Quality of Service, but is not allowed to charge anyone for the use of the service (though it can increase the general price for Internet service),^c (2) can charge its Internet-service customers,^d (3) can charge its Internet-service customers and/or application and content providers, but is required to offer the service to application and content providers on a non-discriminatory basis,^e or (4) can charge its Internet-service customers and/or application and content providers. Some proposals would ban any access charges to application and content providers, not just access charges in return for better transport.^f Finally, proponents of network neutrality disagree over whether a potential non-discrimination rule should be implemented as an *ex ante* rule or as an *ex post* rule, and what agency should be tasked with enforcing it.^g

a. In some proposals, such a rule takes the form of users’ rights to use the (lawful) applications and (legal) content of their choice. Usually there is an exception that allows network providers to block malicious applications and content, such as those involved in denial-of-service attacks.

b. A network that provides “Quality of Service” (QoS) offers different types of service to different data packets. For example, it may guarantee a minimum bandwidth or maximum delay, or it may give some traffic priority over others without giving absolute guarantees. See, e.g., Peterson and Davie 2007, pp. 499–525. For network-neutrality proposals banning QoS, see, e.g., Internet Non-Discrimination Act of 2006 (2006), §4(a)(6); Crawford 2007,

Box 6.1

(continued)

pp. 403–404; for network-neutrality proposals allowing QoS, see, e.g., Internet Freedom and Nondiscrimination Act of 2006 (2006), §3(2); Internet Freedom Preservation Act (2007), §12(a)(5); Lessig 2006, p. 10.

c. E.g., Internet Freedom and Nondiscrimination Act of 2006 (2006), §3(2); Network Neutrality Act of 2006 (2006), §4(a)(7); Internet Freedom Preservation Act (2007), §12(a)(5).

d. E.g., Lessig 2006, pp. 8–10. For criticism of this proposal, see Frischmann 2005, pp. 1009–1012.

e. E.g., Lessig 2008b, p. 9. For criticism of this proposal, see Sidak 2006.

f. E.g., Internet Non-Discrimination Act (2006), §4(a)(3); Lee and Wu 2009. For criticism of this proposal, see Hahn and Wallsten (2006) and Hemphill (2008).

g. E.g., Lessig 2008b, pp. 8–9 (favoring a legislative, *ex ante* rule enforced by the FCC); Shelanski 2007, pp. 101–105; Weiser 2008 (both favoring an *ex post* regime enforced by the Federal Communications Commission or by the Federal Trade Commission).

Both the analysis in this chapter and the current policy debates turn on the question of whether, and under what circumstances, a network provider has an incentive to discriminate against, or to exclude, applications or content. After all, if network providers do not have an incentive to use the discriminatory functionality available in the discriminatory architecture, the differences in architecture do not affect the economic environment for innovation. Because the answer may differ depending on the structure of the market for Internet services, the following analysis proceeds in two steps. In the first step, we will suppose that the network provider is a local monopolist.¹⁶ I don't make any assumptions about the size of the network provider's network relative to the nationwide network. Thus, the network provider's network may be much smaller than the nationwide network. In the extreme case, the network provider owns the nationwide network and has a nationwide monopoly on the provision of Internet services. It may also offer products in the market for applications, content, or portals.¹⁷ In the second step, we will suppose that the network provider competes with at least one other provider. To isolate the effect of architecture, we will assume, in both situations, that there are no legal rules preventing discrimination or exclusion.¹⁸ The analysis focuses on the competitive interactions between the "network" and the "applications."

Architecturally, the network consists of the network layer and the Internet layer, while the application domain consists of the transport layer and the application layer.¹⁹ Economically, the network comprises two layers of economic activity: the operation of physical networks and the provision of Internet Protocol (IP) access and transport services. In reality, these activities may (or may not) be provided by different actors with separate economic interests. The resulting competitive interactions between network operators and Internet service providers have featured prominently in the debate over whether independent ISPs should have “open access” to broadband cable networks in the United States.²⁰ To highlight the effects of architectural differences, the following analysis treats these players as a single economic entity, the “network provider,” that provides Internet access and transport services and operates the network infrastructure of a given physical network. The resulting service will be called *Internet service*.

As we will see, a network provider may want to discriminate for one of three reasons: to increase or protect its profits, to control the bandwidth use on its network, or to exclude unwanted content.

The Network Provider Is a Monopolist in the Market for Internet Services

Economic theory predicts that a network operator with a monopoly in the market for Internet services will not generally have an incentive to discriminate against independent applications. According to the “one monopoly rent” theory, a monopolist has no incentive to monopolize the market for a complementary product if the complementary product is used in fixed proportions with the monopoly good and is competitively supplied.²¹

In this case, there is only one final product and therefore only one monopoly profit available in the market. The monopolist can extract the complete monopoly profit through its pricing of the monopoly good, and gains no additional profit from monopolizing the complementary good. (For an example, see box 6.2.)

This line of reasoning suggests that the monopolist need not monopolize the secondary market to extract the entire monopoly rent, and will therefore have no incentive to drive rivals from that market. In the context of the Internet, this would imply that an ISP with a local monopoly in the market for Internet services does not have an incentive to monopolize the market for a particular application, because it can extract the complete monopoly profit simply through the pricing of its Internet service.

Moreover, economists note that the monopolist may benefit from the presence of independent producers in the market for complementary products; if it benefits from their presence, the monopolist will welcome, not

Box 6.2

"One Monopoly Rent" Theory

To understand the reasoning behind the "one monopoly rent" argument, imagine two complementary products, A and B, that are used in fixed proportions.^a Consumers are willing to pay amount v for a system consisting of one unit of A and one unit of B. The marginal costs of producing component A and B are c_A and c_B , respectively. Product A is supplied by monopolist M. Under these circumstances, the maximum profit from the sale of a single system will be $v - c_B - c_A$. Monopolizing the market for B would not help the monopolist realize a higher profit.

If the market for product B is perfectly competitive, B will be sold at a price equal to its marginal cost, c_B . Because consumers pay c_B for product B, the monopolist M can charge $v - c_B$ for product A, and will make a profit of $v - c_B - c_A$.

If M has monopolized the market for B, it can sell systems consisting of A and B. Because v is the most consumers are willing to pay for the system, M charges v for the system and again makes a profit of $v - c_B - c_A$. Thus, M cannot increase its profits by extending its monopoly from A to B.

a. For similar examples in the context of tying, see Whinston 2001, p. 70; Gilbert and Katz 2001, pp. 33–34.

exclude, independent producers of complementary products. This argument has been called *internalizing complementary efficiencies*.²²

If the presence of these independent producers generates additional surplus, the monopolist may be able to capture some of that surplus through its pricing of the primary good. In this case, the monopolist will earn greater profits when its rivals are in the market than when they are not in the market. The monopolist does not want to take sales from its rivals in the secondary market, but derives its profits by charging a higher price for the primary good.²³ Customers may find Internet service more attractive if they can access a broader range of applications and content through this service. Thus, increasing the number of applications and the amount of content available online may enable the network provider to charge a higher price for Internet service.

Whether the presence of independent producers generates additional surplus depends on consumers' preferences, as well as on such things as the intensity of competition and the degree of differentiation in the complementary market (box 6.3).

Box 6.3

Independent Producers as a Source of Surplus

The presence of independent producers of complementary products may benefit the primary-good monopolist. As the intensity of competition increases, prices are driven down to marginal costs. Owing to the complementarity between both products,^a the monopolist benefits from lower prices in the complementary market. The lower that prices are in the complementary market, the higher the demand (if demand is responsive to price) or the consumer surplus (if demand is inelastic) will be—and, consequently, the higher the profits will be that can be extracted in the primary market.^b

If the producers of a complementary component are not equally efficient, the monopolist has an incentive to let the most efficient supplier make the sales. This maximizes the consumer surplus available for extraction in the primary market.^c

Given complementarity between markets and appropriate consumer preferences, an increase in the quality or variety of complementary goods will increase consumers' valuation of the primary good. Consumer surplus rises if a rival with a differentiated complementary product enters the market and some consumers prefer the new product.^d The value consumers derive from greater variety may differ depending on the type of complementary product. For example, consumers may value a fifth teleconferencing application less than they value a fifth multi-player online game.

Sometimes, innovation and lower prices may be mutually exclusive goals. Behavior by a monopolist that forces independent producers to lower the quality-adjusted price of their products and price closer to marginal cost may improve *ex post* efficiency (the rivals price closer to marginal cost, increasing output and efficiency). However, this may inefficiently destroy their incentives to innovate by reducing or eliminating their returns to innovation.^e

a. In general, two goods are complements if a decrease in the price of one increases the demand for the other. See Varian 1999, p. 112.

b. See, e.g., Farrell and Katz 2000.

c. See, e.g., Ordovery, Sykes, and Willig 1985, p. 117.

d. See, e.g., Whinston 1990, pp. 850–852; Carlton and Waldman 2000, pp. 10–11.

e. For a model demonstrating this effect, see Farrell and Katz 2000.

Whereas the “one monopoly rent” theory argues that exclusionary conduct in the complementary market will not increase the monopolist’s profits, the “internalizing complementary efficiencies” theory suggests that such conduct may even reduce its profits.

Economic research shows that this line of reasoning is incomplete. In some cases, the logic underlying the “one monopoly rent” argument breaks down and the monopolist profits from monopolizing the complementary market. In these cases, the monopolist may profit from the presence of independent producers in the complementary market, but the loss of these profits amounts to a cost that may be more than offset by the gains associated with monopolizing the secondary market. In other words, although the monopolist may profit from the presence of independent producers by enjoying higher profits in the primary market, it may profit even more by excluding them from the market. The monopolist will engage in exclusionary conduct only if the associated profits are larger than the associated costs.²⁴

In each of the exceptions described below, the network provider faces the same trade-off: the exclusion of rivals in the market for a specific application, content, or portal will increase the network provider’s profits. At the same time, the network provider’s Internet-service customers may find its Internet service less attractive if they cannot use the excluded application, content, or portal. This will make the network provider’s Internet-service offering less profitable. The network provider will have an incentive to discriminate against a particular complementary product only if the gains from excluding it exceed the profits lost in the market for Internet services.

The following analysis sets out the theoretical framework for these exceptions, highlights the conditions under which they apply, and demonstrates that in the context of the Internet, these conditions may be met. In the first set of exceptions, the exclusionary conduct increases the monopolist’s current profits. In the second set of exceptions, it preserves the monopolist’s current profits.

Exclusionary Conduct Increases Current Profits (Monopoly Extension)

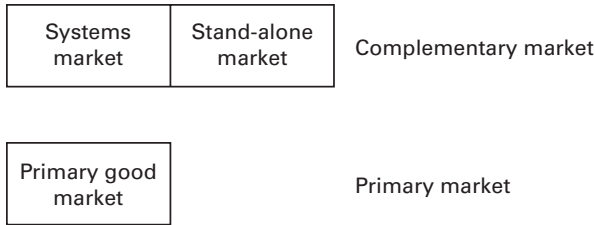
The “one monopoly rent” argument is based on the assumption that the monopolist can extract all potential monopoly profits through sales of its primary good, so monopolizing the second, complementary market does not increase its profits. But research has identified a variety of circumstances in which this assumption does not hold true. Under these circumstances, the monopolist can increase its profits by extending the monopoly to the complementary market.

Four exceptions are particularly relevant in the context of the Internet. In the first exception, rate regulation in the primary market prevents the monopolist from gaining the complete monopoly profit through its pricing of the primary good. In the second exception, the primary good is not necessary for some uses of the complementary good, which makes it impossible for the monopolist to extract all monopoly profits through its pricing of the primary good. In the third exception, the complementary product is a source of outside revenue (such as advertising revenue or sales commissions) that the monopolist cannot extract in the primary market. In the fourth exception, which is a variant of the third, only the monopolist's complementary product is a source of outside revenue, and this revenue is lost when rival producers make the sales. This exception is particularly relevant to Voice over Internet service.

Rate Regulation in the Primary Market Rate regulation limits the price a monopolist can charge for its product, forcing the monopolist to set a price lower than the monopoly price. If a monopolist is subject to rate regulation in the primary market, this will prevent it from realizing monopoly profits in the primary market. By extending its monopoly to a second, unregulated complementary market, the monopolist gains the ability to recover forgone monopoly profits through sales of the second good.²⁵

In the context of the Internet, if regulation restricts the price that a network provider with a local monopoly in the Internet-services market can charge for Internet services, then monopolizing the market for one or more applications may enable the network provider to make up for lost monopoly profits. This may give the network provider an incentive to exclude rivals from that market, or to discriminate against rivals in order to shift sales to its own application product. The profitability of this strategy will depend on several factors, including the extent of product differentiation and the intensity of competition in the application market, the relative efficiency of the different application providers, and the size of the profit margin in the regulated market for IP services.²⁶ In the United States, network providers are not subject to rate regulation in the market for Internet services, so this exception is not applicable.

Primary Good Not Essential In this exception, uses of the complementary good that do not require the primary good make it impossible for the monopolist to extract all monopoly profits through its pricing of the primary good. For this exception to apply, three conditions must be met²⁷: the monopolized product is not essential for all uses of the complementary good (i.e., some uses of the complementary good do not require the primary

**Figure 6.1**

The “systems market” and the “stand-alone market.”

good), the monopolist can prevent its rivals from selling their version of the complementary good to users of the primary good, and the complementary market is subject to economies of scale or network effects. The first condition explains why the monopolist would want to extend its monopoly to the complementary market in spite of its monopoly in the primary market. The second and third conditions provide the mechanism that enables the monopolist to extend its monopoly to the complementary market.

Suppose there is some use of a complementary good that does not require the primary good. As a result, the complementary market is divided into a “systems market” for uses that require the primary good and a “stand-alone market” for uses that do not require it (figure 6.1). Consumers in the “systems market” desire both the primary and the complementary good, whereas consumers in the “stand-alone market” desire only the complementary good.

Now suppose there are rival producers of the complementary good. The monopolist can extract all monopoly profits in the “systems market” through its pricing of the primary good. But since consumers in the “stand-alone market” do not buy the primary good, the monopolist derives no profit from its rivals’ sales in that market. Moreover, the presence of rivals constrains the monopolist’s ability to price its version of the complementary good in the “stand-alone market.” Thus, the monopolist cannot earn monopoly profits in the “stand-alone market” unless it has a monopoly in that market too. Consequently, the monopolist needs to drive its competitors from the complementary market to be able to charge monopoly profits in the “stand-alone market.”²⁸

The second and third conditions provide the mechanism that enables the monopolist to extend the monopoly to the complementary market: in the presence of economies of scale or network effects, the monopolist may be able to drive potential rivals from the complementary market by excluding them from the “systems market.”

When the second condition is met, the monopolist can deprive rival producers of complementary products of any sales in the “systems market.” This behavior does not exclude rivals from the “stand-alone market.” Given economies of scale²⁹ in the complementary market, the remaining sales to customers in the “stand-alone market” may not suffice to reach an economically efficient scale. Thus, rivals excluded from the “systems market” may be forced to exit the “stand-alone market” too.

Similarly, in the presence of network effects (box 6.4) in the complementary market, exclusion from the “systems market” may suffice to drive competitors from the complementary market or into a niche existence.

If the benefits derived from a bigger network are large relative to the benefits of product differentiation in the network good,³⁰ competition between two incompatible technologies^{31, 32} usually will result in domination of the market by a single technology.³³ This is because network effects give rise to strong positive feedback loops in technology adoption. If a technology is subject to network effects and the competing technologies are incompatible, a larger network is more valuable, so more consumers will join that network. The resulting increase in network size will make that network even more valuable, which, in turn, will motivate even more consumers to join it. Once this positive feedback loop sets in, the affected technology will quickly pull ahead of its rivals in market share, and will ultimately dominate the market. This phenomenon is known as “tipping.”³⁴

Because small initial advantages may quickly get magnified, minute differences (in either perception³⁵ or reality) may determine who wins the competition for the market. Therefore, establishing an early lead in installed base (i.e., in the total number of customers who already bought the network good) large enough to start the positive feedback loop is an important strategy in network markets.³⁶ Thus, if a monopolist blocks its rivals from access to its primary-good customers in the “systems market,” the monopolist can capture *all* customers who want to buy the complementary product in that market. If the “systems market” is large enough, the monopolist’s advantage in that market may enable it to reach a critical mass of the complementary product’s customers that is so attractive to other customers that the positive feedback loop sets in. Once this happens, rivals will not be able to catch up.

If the presence of rivals increases consumer surplus, the exclusion of rivals may reduce the monopolist’s profits in the “systems market.”³⁷ In this case, monopolizing the complementary market increases the monopolist’s profits if the gain from monopolizing the “stand-alone market” is

Box 6.4

Network Effects

Network effects exist if the utility that an individual customer derives from a good depends on, and increases with, the number of other customers who buy that good (or goods compatible with it).^a

If the benefits of consuming a good directly result from the size of the network,^b the network effects are “direct.”^c Communication services such as the telephone, e-mail, or instant messaging are examples; the more people who can be reached by using the service, the more valuable the service becomes. Direct network effects exist for some software applications as well. The greater the number of people who use a particular word-processing software, the easier it is to exchange documents. The greater the number of people who own a particular multi-player game, the easier it is to find someone with whom to play it.

Network effects of another kind arise if consumers’ demand for the primary good increases with the variety of complementary goods and services.^d These “indirect” network effects arise from supply-side economies of scale in the complementary market. A larger installed base for the primary product (i.e., a larger number of consumers who have already bought the primary product) allows application developers to spread sunk development costs over a larger potential sales base. Thus, in the presence of economies of scale and free entry into the complementary market, a larger customer base leads to lower costs and greater variety of complementary products. For example, application developers find it more attractive to develop applications for an operating system with a larger number of customers. The higher number of applications available for the operating system with the larger number of customers, in turn, makes this operating system more attractive to customers who consider buying a new operating system. Thus, the value consumers derive from the operating system increases with the number of other users of the operating system. Contrary to direct network effects, consumers do not derive benefits from the larger network as such, but from the increased variety of complementary products that results from the larger network.

a. See the definition by Katz and Shapiro (1985, p. 424). For an analysis of network effects in the context of information goods, see Shapiro and Varian 1999, chapters 7–9. For an analysis of the legal implications of network economic effects, see Lemley and McGowan 1998. For critical voices, see Liebowitz and Margolis 2001; Kolasky 1999.

b. The size of the network is the number of consumers owning compatible units of the network good.

c. See, e.g., Katz and Shapiro 1985, p. 424.

d. See, e.g., Katz and Shapiro 1985, p. 424; Katz and Shapiro 1994, p. 99.

larger than the loss resulting from excluding rivals from the “systems market.”³⁸ Thus, the monopolist has an incentive to exclude its rivals only if the gains in the “stand-alone market” are larger than the losses in the “systems market.”

If the complementary market is subject to network effects, two factors make it particularly likely that the gains in the “stand-alone market” are indeed larger than the loss in the “systems market,” so that the monopolist will have an incentive to discriminate. First, the potential profits from winning the competition between incompatible technologies are huge. Higher profits from monopolizing the complementary market make it more likely that the benefits of exclusion are larger than the costs. Imagine a competition between incompatible technologies both of which are subject to indirect network effects. If the winning standard is protected by intellectual property, the winner can make money on any primary or complementary product that uses the standard. In view of the potentially large number of complementary products in markets with indirect network effects, licensing fees can lead to substantial profits.³⁹ For example, in a competition between media-player technologies, a winner with a proprietary standard protected by intellectual property will not only dominate the market for media players, but will also be able to charge licensing fees for every song or video encoded for use with the player. Thus, in markets subject to network effects the benefits of exclusion are particularly high. Second, if the complementary product is subject to network effects, the presence of an independent rival in the complementary market will not necessarily increase the monopolist’s profits in the “systems market.” If the monopolist’s complementary product and the rival’s complementary product are not compatible, sales to the rival decrease the size of the user network for the monopolist’s complementary product. As a result, the value that users can derive from the monopolist’s complementary product (and the profit the monopolist can extract from them in the primary market) is lower than it would be if the rival made no sales.⁴⁰ Thus, although the presence of a rival in the complementary market may increase the network provider’s profits in the primary market (because customers of the rival product value the ability to use that product), it may also reduce the profits that the network provider can derive from selling its own version of the complementary product. Overall, this lowers the costs of exclusion. Higher benefits and lower costs make it more likely that exclusion will be profitable.

The conditions underlying this model may be met in the Internet context. Consider the market for residential broadband Internet access in

the United States.⁴¹ In some areas of the country, the owner of a cable network or a telephone network that provides broadband Internet access through an affiliated broadband Internet access provider may well be a local monopolist.⁴² If this network provider offers a complementary product not only to customers of its Internet service but also to customers nationwide, then its primary product, Internet service, will be essential only to its Internet-service customers. The area covered by the network constitutes the “systems market.” Customers who live outside that area constitute the “stand-alone market.”

This business model is common. For example, where AOL can strike a deal with owners of cable networks, it offers its portal bundled with broadband Internet access. Consumers nationwide can also buy the portal without access; this is known as the “bring your own access” option.⁴³ Other AOL services, including MapQuest and AOL Moviefone, are also offered to all consumers on the Internet. Microsoft’s search engine and Hotmail are available to all, not only to customers of Microsoft’s Internet service.⁴⁴ Similarly, if a provider of narrowband access has a local monopoly on narrowband access but offers its portal to all Internet users, the narrowband-access service will not be essential for customers accessing the portal via broadband-access services.⁴⁵

In the hypothetical network, the monopolist can prevent rivals’ applications, content, or portals from running over its network. As a result, the monopolist’s Internet-service customers—the consumers in the “systems market”—are not able to access or use these products. Thus, rivals are deprived of sales in the “systems market.”

The markets for software applications, Internet content, and portals are subject to significant economies of scale. Whereas development of these products and services is characterized by large fixed costs, the marginal costs of production and distribution over the Internet are small. Thus, the marginal cost of production⁴⁶ is low relative to the average cost of production,⁴⁷ resulting in significant economies of scale.⁴⁸

In addition, many software applications are subject to direct or indirect network effects.⁴⁹ For example, instant messaging and Internet telephony become more valuable as more and more people can be contacted through them.⁵⁰ Viewers of multimedia content are subject to indirect network effects⁵¹: the larger the catalog of content available in a given format, the more highly users will value viewers that are compatible with that format. At the same time, the larger the installed base of viewers compatible with that format, the more likely it is that a content provider will be willing to incur the costs of coding its content in that particular format.

Given economies of scale or network effects, an Internet service provider may be able to force its rivals from the nationwide market for a particular application, content, or portal by excluding them from access to its Internet-service customers. Whether this exclusion will drive competitors from the nationwide market will depend on the size of economies of scale with respect to the complementary product, on the strength of potential network effects with respect to that product, and on the sizes of the monopolist's network (i.e. the number of its Internet-service customers) and the remaining network (i.e. the number of customers of other Internet-service providers).

If a network provider prevents market rivals for a particular application or for particular content from accessing its Internet-service customers, this reduces the number of applications and the amount of content available to these customers; consequently, these customers may not be willing to pay as much for the Internet service. Thus, a network provider will have an incentive to monopolize the market for a particular type of application, content, or portal only if the profit from additional application, content, or portal sales nationwide more than offsets the reduction in revenues from broadband access.⁵² Whether this condition is met is an empirical question.⁵³

In some cases, the network provider may not manage to drive its competitors from the complementary market—for example, because the number of its Internet-service customers may be too small, relative to the size of the market for the application or content in question, to deprive its rivals of minimum efficient scale. As we will see, this does not necessarily imply that the network provider will not have an incentive to discriminate: exclusion may still be a profitable strategy, even if the network provider does not manage to monopolize the complementary market.

Complementary Product Source of Outside Revenue A monopolist in the primary market may also be unable to extract the maximum possible profit through its sales of its primary good if some of the revenue in the complementary market comes from outside sources. For example, a firm might follow the example of the mass media, selling access to its customers to advertisers and other third parties.⁵⁴ In the extreme case, consumers get a firm's product or service for free, while all of the firm's revenue comes from outside sources.⁵⁵ Google's search engine is an example of this. This exception may apply whether or not the monopolist's primary product is essential for use of the complementary good.

If firms in the complementary market derive some of their revenue from outside sources, a monopolist in the primary market may be unable to maximize profit unless it monopolizes the market for the complementary good too.

Consider first the variant of the exception in which all uses of the complementary good require use of the primary good. For this variant to apply, the monopolist must earn less if it extracts the advertising revenue from its rivals than if it excludes its rivals, monopolizes the complementary market, and captures all outside revenue directly. Exclusion in the market for the complementary good may reduce the value to consumers of the primary good and hence reduce profits in the primary-good market. For exclusion to be profitable overall, the net gains in outside revenue that result from exclusion must be larger than the profit reduction in the primary-good market.

Suppose, for example, that firms in the complementary market offer their product or service for free and make all their revenue from selling access to their customers to advertisers and other third parties.⁵⁶ Usually, the primary-good monopolist can use a variety of tactics to “squeeze” revenue from its rivals in the complementary market.⁵⁷ Suppose that these tactics are perfectly effective and extract all profits from the complementary market.⁵⁸ Even in this case, exclusion can be more profitable than employing these tactics if exclusion increases the total profitability of the complementary market.

If the value of the complementary market is greater when foreclosed, the net gains from exclusion will be larger than the net gains from extraction no matter how well the monopolist is able to extract rents from the complementary market.⁵⁹ The crux of this argument is whether and how foreclosure will increase the value of the complementary market. This is where outside revenues are crucial. First, outside revenues may increase with monopolization, insofar as a single provider is able to exert market power and increase prices. Second, the value of access to customers may increase disproportionately with the number of customers being accessed. For example, an advertiser may value access to 100,000 customers at \$10,000 (10 cents per customer) but may value access to a million customers at \$200,000 (20 cents per customer). Third, the monopolist may have information that allows it to increase the value of access to consumers in the complementary market. Fourth, if all firms in the complementary market incur fixed costs, the reduction in fixed costs resulting from the reduction in the number of firms may increase the overall value of the complementary market. Fifth, employing the tactics used to extract rents

without foreclosure may involve higher administrative and negotiation costs than foreclosure.

By monopolizing the complementary market, the primary-good monopolist will also monopolize the side of the market that consists of selling third parties access to the users of the complementary good.⁶⁰ Before foreclosure, third parties who wanted to buy access to users of the complementary good had a variety of platforms they could use. After foreclosure, they must go through the monopolist's complementary product. If providers of the complementary product and third parties negotiate advertising fees, the primary-good monopolist probably will be in a stronger bargaining position than suppliers of the complementary good previously were in, and may be able to negotiate a higher price. Alternatively, if advertising prices are determined through an auction, the auction is likely to be more competitive when the third parties all have to bid in the same auction.⁶¹ Finally, if suppliers of the complementary good directly set prices for advertisers, the monopolist may be able to charge higher (per customer) prices directly as a result of the reduced competition than suppliers of the complementary good could previously charge.⁶²

For various reasons, third parties may value access to customers disproportionately such that with access to more customers they have a higher value per customer. The value of access to a customer can be thought of as the expected additional profits made from additional sales to that customer less the cost of access to that customer (e.g., delivering an advertisement to that customer). There are then three ways to increase the value of access to a given customer: increase the profit margin on additional sales to that customer, increase the additional sales to that customer, or decrease the cost of accessing that customer. For example, if an advertiser incurs a one-time cost of dealing with a provider of a complementary good (for example, because the advertising prices are negotiated), accessing a large block of customers through a single provider of a complementary good rather than accessing several smaller blocks through several providers will reduce the per-customer costs of accessing customers and will increase the value of accessing a specific customer in the sense defined above. In addition, access to different customers can result in different values through the three components identified above. For example, a customer who buys more items with higher profits margins and is cheaper to access will be more valuable. Ultimately, how buying access to more customers at once affects the value per customer is an empirical question.

Owing to its relationship with consumers in the primary market, the monopolist may have information about its customers that enables it to

charge higher prices (per customer) to third parties than other firms in the complementary market.⁶³ The third parties may value this information, insofar as it allows them to better target the type of consumers in which they are interested.⁶⁴

If fixed costs are high relative to marginal costs, foreclosure will reduce the total costs of servicing customers by eliminating the duplication of the fixed costs. If the primary-good monopolist chooses to extract outside revenue from complementary-market producers without foreclosing the market, the monopolist will have to leave sufficient rents for the producers of the complementary good to cover their costs. Foreclosure can then increase the total rents that can be extracted from the complementary-good market.

Finally, even if the per-customer prices charged to third parties stay the same, the monopolist's profits may be lower if there are rivals in the market, owing to the costs of negotiating and administering the access fees.

Thus, the monopolist will have an incentive to exclude its rivals from the complementary market if the net gains from capturing the outside revenue directly more than offset the reduction in profits that results from the reduction in variety of complementary goods.⁶⁵

The other variant of this exception may apply if the monopolist's primary good is not essential for all uses of the complementary good. This is a special case of the "primary good not essential" exception outlined above. Since the primary-good monopolist cannot extract the outside revenue generated by consumers who use a rival's complementary good without the monopolist's primary good, the monopolist has an incentive to monopolize the market for the complementary good to capture all the outside revenue available in that market. If the complementary good is subject to economies of scale or to network effects, the monopolist may be able to drive its competitors from the "stand-alone market" (the market for uses of the complementary good that do not require the monopolist's primary good) by excluding them from the "systems market" (the market for uses of the complementary good that require the monopolist's primary good).⁶⁶ Again, the monopolist will have an incentive to discriminate only if the net gains in the complementary market are larger than the profit reduction in the primary market.

The conditions underlying this exception may be met in the Internet context. In the markets for Internet content, portals, and applications, firms often derive at least some of their revenue from outside sources by selling access to customers to advertisers or to online merchants.⁶⁷ For

example, the search engine Google makes almost all of its revenue from advertising fees.⁶⁸ The portals AOL and Yahoo are at least partly financed by advertising fees or online sales commissions.⁶⁹ The social networking site Facebook also is based on this business model.

Consider first the variant of the exception in which the network provider offers the complementary product only to customers of its own Internet service. Since the discriminating functionality in the network enables the monopolist to exclude applications from the network, the monopolist network provider can extract at least some of its rivals' outside revenue. The monopolist can condition the "access" of rivals' complementary products and services on payment of an access fee that captures some or all of the rivals' outside revenue. Some owners of cable networks in the United States use this practice with respect to ISPs. Unaffiliated ISPs who want to offer their service over a cable network have to pay a fixed fee per customer. In addition, the owner of the cable network receives a portion of the outside revenue that the ISP earns per customer.⁷⁰

Thus, the discriminatory technology enables the network provider to choose between two strategies. It can give competing providers of the complementary product access to its network and charge a fee to extract some of their advertising revenue. Under this strategy, the rivals have to keep enough advertising revenue to pay the costs associated with serving the customers of their complementary product. Alternatively, the network provider can exclude all competing providers of the complementary product from its network and sell advertisers access to all of its Internet-service customers who use the complementary product. For example, Comcast could let AOL and Yahoo onto its network, but could require them to pay Comcast at least a portion of their outside revenue, or it could exclude all competing portals from its network, become the only portal available to its Internet-service customers, and directly sell access to its Internet-service customers to all advertisers who would like to advertise to Comcast's Internet-service customers on the portal.

Although a monopolist will be able to capture some or all of its rivals' outside revenue in the complementary market by threatening exclusion, its outside revenue may be higher if it excludes rivals from the complementary market and collects the outside revenue directly.

First, selling access to one large group of customers as a whole may yield substantially more revenue than selling access to subgroups of that group separately. This is obvious if the monopolist network provider manages to monopolize the market in which advertisers buy online access to its Internet-service customers.⁷¹ To do this, however, the network provider

would have to exclude from its network all applications, content, and portals that are financed through advertising and are operated by providers not affiliated with the network provider. This may not be a realistic strategy. However, foreclosure of many applications, content offerings, and portals is likely to generate some market power for the network provider and allow it to charge a higher rate per customer even if the network provider cannot completely monopolize the market in which advertisers buy online access to its Internet-service customers. Alternatively, the network provider is likely to gain some market power over advertisers through exclusion from the complementary market if advertisers view “access to customers through the complementary product” as a differentiated product that they cannot get elsewhere. Products in this category may include “yellow pages” (because they provide advertisers with access to customers looking to buy something), portals (because of their reach),⁷² websites specializing in a particular type of content such as cars or finance, and websites catering to a specific demographic group (because they provide access to customers with specific interests or characteristics).⁷³ If the network provider manages to monopolize the customer side of one of these product categories, it may be able to realize higher advertising rates by becoming the sole provider of this type of advertiser access to its Internet-service consumers.⁷⁴ Even without monopolizing a specific market in which advertisers buy access to the network provider’s Internet customers, selling access to a large block of customers may be more profitable than selling access to subgroups of that block. For example, Google has more users than Yahoo or Microsoft,⁷⁵ and is currently able to realize higher revenue per search than either of these companies.⁷⁶ The ability to realize higher revenue per search is attributed to two factors. First, there are network effects between the advertiser and the user side of the market. More users make a search engine more attractive for advertisers. Even if an ad is relevant to a user, the probability that a user will click on the ad and buy something on the website is relatively low. As a result, advertisers want to show their ads to a large number of users to have a meaningful number of people actually buy their product.⁷⁷ As Google has by far the largest number of users, Google’s advertising space is in the greatest demand. The prices for ads shown to users of search engines are determined at auctions where advertisers can bid for the right to have their ads shown. A higher number of advertisers increases competition in these auctions, which increases the prices for ads shown.⁷⁸ At the same time, users interested in buying something value the ability to view relevant ads; the more advertisers are willing to show their ads on the search engine, the higher

the chance that there will be advertisers whose offerings are relevant to the user.⁷⁹ Second, Google's methods of charging for ads and of placing ads result in more revenue per search than, for example, Yahoo's.⁸⁰ Google determines where ads will be placed not only on the basis of what price the advertiser is willing to pay, but also on the basis of the likelihood that the ad will be clicked on. This lets relevant ads rise to the top, even if the corresponding advertiser has bid to pay a lower price. As advertisers pay only for ads that are clicked on, and the mechanism shows ads that are more likely to be clicked on, this increases the overall revenue per search. Now suppose that Google was a network provider and could exclude competing providers of search engines from access to its Internet-service customers. Google would make more revenue if it was the only search engine its customers could use (and it sold advertising on its search engine directly) than if some of its Internet-service customers used other search engines, even if the other search engines gave all of the resulting advertising revenue to Google.⁸¹ Owing to the network effects operating between the customer side of the market and the advertiser side, the increase in customers probably would enable Google to realize even higher revenue per search.⁸² Furthermore, selling access to a large block of customers as a whole may be more profitable than selling access to subgroups of that block separately, because advertisers value both breadth and depth; they want to reach as many consumers as they can, and they want to reach consumers who have specific needs or characteristics.⁸³ Content, portal, or application providers with a large number of customers can often meet both needs, so they command higher advertising prices per click. In addition, a large provider of applications or content often has an in-house sales team that sells ad space on the company's online offerings directly to advertisers. Smaller providers of content or applications do not have their own sales teams and often sell their advertising space through advertising networks (intermediaries that aggregate ad space from a large number of content or application providers and sell it to advertisers). Ad space on large websites that is sold directly to publishers can be sold at a much higher rate than ad space sold through advertising networks. For example, the investment firm Piper Jaffray estimated that an ad on a typical portal or a large specialized website could cost at least \$8–\$10 per 1,000 page views (for example, an ad on MSN may cost \$45 to \$70 per 1,000 page views, and an add on the website of the *Wall Street Journal*, which reaches a specific and highly valuable audience, may cost \$65 to \$90 per 1,000 page views), whereas an average collection of sites sold through an advertising network may cost about \$1–\$2 per 1,000 page views.⁸⁴ This means that a website belonging to a

larger conglomerate that sells its own advertising space can command much higher advertising revenue per user than an independent website. Imagine a network provider whose website or application belongs to the first category. For example, Yahoo partners with network providers to sell co-branded Internet access services.⁸⁵ In addition, Yahoo has many websites, some with very large reach (such as portals) and some targeted at specific customer groups (such as auto and finance sites). This enables Yahoo to sell its advertising space at very high rates.⁸⁶ Now imagine that Yahoo's finance offering competes with a new independent website that is not part of a larger conglomerate, does not have its own advertising sales force, and sells its advertising space through an advertising network. Yahoo could realize more advertising revenue if the users of the independent site used Yahoo's website than if they used the independent website, even if the independent website transferred all of its advertising revenue to Yahoo.

Second, the network provider's revenue may be higher if it excludes rivals from the complementary market and collects the outside revenue available in that market directly, because through its billing relationship with customers of its Internet service, a network owner has data about customer demographics. This enables it to charge higher advertising fees or commissions for online sales than many of its rivals in the complementary market.⁸⁷

Third, the network provider may be able to reach higher net gains in outside revenue under foreclosure, if its costs of serving the additional customers after exclusion are lower than the costs incurred previously by its rivals.⁸⁸ Without foreclosure, rivals' costs affect the network provider's revenue: the amount of revenue the network provider can extract from applications, content, and portals will be limited, because the providers need to keep enough of their revenue to cover their fixed and variable costs.⁸⁹ Many applications, content providers, and portals have high fixed costs and low marginal costs.⁹⁰ This makes it more likely that exclusion provides a net gain in outside revenue: owing to the low marginal costs, the network provider's costs of serving additional customers after foreclosure will be small, but the high fixed costs associated with a specific application will be paid only once, not several times.

Finally, owing to the potentially large number of complementary products, negotiating and administering the access charges for unaffiliated content, applications, and portals may be prohibitively expensive. In any event, these transaction costs will further decrease the monopolist's profits if rivals are present in the market.⁹¹

Thus, if firms in the market for a particular type of application, content, or portal derive some of their revenue from outside sources, a monopolist in Internet services may have an incentive to monopolize that market in order to capture all the outside revenue available in that market directly. This will be true as long as the net gains in the complementary market exceed the reduction in profits in the market for Internet services.

Now consider the second variant of the exception, in which the network provider offers its complementary product not only to its Internet-service customers but also to customers nationwide. In this case, it may have an incentive to exclude its rivals in the complementary market from access to its Internet-service customers in order to drive the rivals from the nationwide market and to capture the advertising revenue in the part of the market that consists of the customers of other network providers who use the complementary product (this is a special case of the “primary good not essential” exception described above). For example, suppose AOL had a monopoly in a particular local market for Internet services. This would not let AOL capture the advertising revenues made by rival portal providers in areas outside AOL’s local market. Thus, in this example, AOL would have an incentive to exclude rival portal providers from access to its Internet-service customers, in order to monopolize the nationwide market for portals and capture all the advertising revenue in that market, if the resulting net gains in the complementary market would be larger than the reduction in profits in the market for Internet services. The analysis in this case follows the analysis in the subsection on the “primary good not essential” exception above.

Monopolist’s Complementary Product Source of Outside Revenue In the last exception, only the network provider—not its rivals in the complementary market—can realize higher outside revenues. Thus, letting rivals make the sales and then extracting the outside revenue from them is less profitable to a network provider than making the sales directly.

This exception is a variant of this line of reasoning. The network provider’s offering is a source of outside revenue; the rivals’ offering does not provide this revenue. Thus, this revenue is lost if rivals make the sales. As a result, the network provider has an incentive to make as many sales as possible directly.

Consider a local phone company that offers broadband Internet service over its network.⁹² Vonage, Skype, and other independent companies offer Voice over IP (VoIP) services to customers of this network provider. Since the price of long-distance calls made by means of VoIP tends to be lower

than the price of long-distance calls made by means of conventional telephone service, the network provider's customers who use VoIP will place fewer long-distance calls using the network provider's "legacy" telephone service.

To the network provider, conventional long-distance services are a source of outside revenue that is not similarly available to providers of VoIP services. In the United States, local phone companies are paid an "access charge" by long-distance providers for every long-distance call they originate or terminate. Because access charges were traditionally intended to cross-subsidize local telephone service, regulators have mostly set them significantly above the costs of originating or terminating long-distance calls. Thus, for many local phone companies, access charges are an important source of revenue.⁹³ Independent VoIP providers threaten this revenue. When more of the network provider's customers place their long-distance calls using VoIP, the network provider receives fewer access charges. If independent VoIP providers are excluded from the network and the network provider does not offer a VoIP service itself,⁹⁴ customers who want to make long-distance calls will be forced to use the conventional telephone service. Thus, exclusion in the VoIP market protects the network provider's current profits.⁹⁵

It is not surprising that the first publicly documented incident of VoIP blocking in the United States involved a rural telephone company.⁹⁶ In February 2005, the VoIP provider Vonage complained to the Federal Communications Commission that its Internet-telephony application was being blocked by Madison River Communications, a rural telephone company based in North Carolina. In March 2005, after a short investigation, Madison River and the FCC entered into a consent decree. Madison River agreed to pay \$15,000 to the US Treasury and to stop blocking VoIP applications; the FCC terminated the investigation. Access charges make up a substantial portion of rural phone companies' revenue, so the companies have a high incentive to guard them.

In several countries, monopoly providers of telephony services have blocked international VoIP calls in order to maintain the revenue they accrue from settlement charges. For international calls, carriers in different countries negotiate these charges; a settlement charge is split between the two providers. These charges are often set significantly above costs, especially if one of the providers is a national monopolist in its country.⁹⁷ If users in these countries place or receive calls using Internet telephony, the local telephony provider loses the chance to earn revenue from these settlement charges. To prevent this, network providers in Saudi Arabia, the

United Arab Emirates, Mexico, and other countries have blocked Internet-telephony applications altogether.⁹⁸

In the example discussed above, the existence of outside revenue is a result of regulation that requires long-distance providers to pay above-cost access charges to local phone companies. It is more difficult to determine whether local phone companies that are local monopolists in the market for Internet services (this assumption holds throughout the subsection titled “Network Provider Is a Monopolist in the Market for Internet Services”) would also have an incentive to block VoIP in the absence of such regulation (box 6.5).

Exclusionary Conduct Preserves Current Profits (Monopoly Preservation)

In the models described above, the monopolist’s exclusionary conduct is driven by the desire to extend its monopoly to the complementary market in order to increase its current profits. In this section, the monopolist pursues a different goal: by harming rivals in, or excluding rivals from, the complementary market, it seeks to protect its monopoly in that market or in another market. The exclusionary conduct maintains, rather than increases, the monopolist’s current level of profits. In the absence of this conduct, however, the monopolist’s profits would shrink or disappear.⁹⁹

Models in this category share the following features: The primary-good monopolist has a monopoly in one market (call it market X) that is threatened by new entry, by the expansion of fringe competitors, or by other developments. For some reason, the presence of rivals in the

Box 6.5

Incentive to Discriminate in the Absence of Regulation

In the absence of regulation, local phone companies would have no incentive to block provision of Internet-telephony, or Voice-over-Internet-Protocol (VoIP), service by independent providers if they could price their Internet service in a way that allowed them to realize the same revenue that they make by offering traditional telephone service. For example, if someone were to stop using the telephone and begin using an independent provider’s VoIP service, the network provider could increase that person’s Internet-service rate until the additional profit from Internet service equals the profit that the network provider would have realized if the person were still using the telephone (i.e., until the additional profit from Internet service for that customer equaled the lost profit from telephone service for that customer).^a

Box 6.5

(continued)

Replicating current phone profits via Internet-service rates is not trivial, though. Because some telephone charges are based on use, network providers would not only have to introduce use-based elements into the Internet-service pricing structure (a pricing structure currently based on flat-rate pricing); they would also have to price Internet use differently depending on the application being used—for example, charging a different price for transport of VoIP packets than for transport of e-mail packets. Though this is possible in theory, there are many reasons it may not be feasible in practice—for example, customer resistance to deviations from flat-rate pricing, or the cost of implementing a fine-grained accounting and billing system for Internet use. Thus, simply pricing their Internet services differently may not allow network providers to replicate the profits that they generate from telephone use.

Network providers may be able to reach this goal, however, by excluding independent providers' VoIP applications allowing only their own. This was not an option in the example above, because the outside revenue was lost regardless of who provided the VoIP. But if we assume that there is no outside revenue in the case, then, by excluding independent VoIP providers from access to its Internet-service customers, a network provider who is a local monopolist in the Internet-services market also gains a local monopoly in the market for VoIP services. This monopoly, in turn, will give the network provider more flexibility in pricing VoIP, and may enable it to replicate the revenue formerly realized through traditional phone services. For example, it might offer a higher flat rate for Internet services and VoIP services combined. This idea that a firm with a monopoly in a primary market may have an incentive to exclude competitors from a secondary, complementary market in order to get more pricing flexibility (e.g., to facilitate price discrimination) is well acknowledged in the literature.^b

In sum, network providers may not be able to replicate their phone revenues simply by pricing their Internet service differently, and thus they may have an incentive to block the VoIP services offered by independent providers even in the absence of regulation of traditional long-distance access charges.

a. The network provider could also try to replicate traditional telephony profits by charging Internet-telephony applications an access charge that is equal to or greater than the potential lost profit. However, for reasons discussed in notes 28 and 95 to this chapter, this alternative may be less attractive than exclusion.

b. See, e.g., Ordovery, Sykes, and Willig 1985, pp. 119–121; Farrell and Weiser 2003, pp. 107–109.

complementary market is a part of this threat; stated differently, if the presence of rivals in the complementary market continues, the monopolist's position in market X may be weakened. Exclusionary behavior in the complementary market may be able to deter or destroy the emerging competition. Thus, the monopolist harms or excludes its rivals in the complementary market to preserve the monopoly in market X.

The models differ with respect to the market in which the monopolist's position is threatened; they also differ in how they explain why exclusionary behavior in the complementary market can remove that threat. The following analysis highlights two cases that may be relevant in the context of the Internet.

Monopoly Preservation in the Primary Market In this class of models, exclusionary behavior in the complementary market maintains a monopoly in the primary market.¹⁰⁰

In models that fall into this category, the monopolist faces potential competition in the primary market. It can deter entry into the primary market by engaging in exclusionary conduct in the complementary market. By deterring rivals' entry into the primary market, the exclusionary behavior in the complementary market preserves the monopoly in the primary market.

Economists have come up with a number of explanations as to why exclusionary conduct in the complementary market may deter entry into the primary market. One explanation is particularly relevant in the context of the Internet: the exclusionary behavior in the complementary market harms future competitors in the primary market by depriving them of a source of complementary products.¹⁰¹ As a result, in order to make any sales in the primary market, an entrant to the primary market must also enter the complementary market. If this is significantly more difficult or costly than entering the primary market alone, potential entrants to the primary market may be deterred.

For this strategy to succeed, two conditions must be met. First, the monopolist's exclusionary behavior in the complementary market must deprive a potential entrant into the primary market of a source of complementary products. As a result, the entrant cannot enter only the primary market; it must enter both markets at once. Second, simultaneously entering both markets must be more costly than the sum of the costs of entering each market on its own¹⁰²; otherwise, the exclusionary behavior in the complementary market is not likely to have an adverse effect on entry into the primary market.

Economists have identified four reasons why simultaneous entry into both markets may be significantly more costly than the sum of the costs of entering each market on its own: increased cost of capital, differing economies of scale in the two markets, the uncertainty of innovation, and the existence of indirect network effects.

Increased Cost of Capital An entrant forced to break into both markets may face an increased cost of capital if it has experience relevant to operating in only one of the markets. If the skills and knowledge necessary to succeed in both markets differ considerably, the increased probability of failure due to inexperience in one of the markets may lead lenders to charge a higher rate for the necessary capital. The risk premium will be even larger if the entrant has to incur huge sunk costs to enter the market, because sunk costs cannot be recovered if the entrant fails.¹⁰³

Differing Economies of Scale Entering two markets is more difficult than entering one market if the minimum efficient scale in the two markets differs considerably. If it does differ considerably, an entrant must choose between operating on an inefficiently small scale in one market and operating on an unnecessarily large scale in the other. Either strategy may significantly increase an entering firm's operating costs.¹⁰⁴

Uncertainty of Innovation In view of the uncertainty associated with the innovative process, the need to innovate successfully in two markets probably will decrease the chance of successful entry. Let us assume that the probability of innovating successfully in one component is k . In this case, the chances of successful innovation in n components are k^n . Unless k is close to 1, k^n is considerably lower than k .¹⁰⁵ Thus, the probability of successful innovation in n components required to enter n markets simultaneously is lower than the probability of successful innovation and successful entry in one component market.

Existence of Indirect Network Effects If the primary good is subject to indirect network effects¹⁰⁶ and any available complementary goods are offered exclusively with the monopolist's platform, an entrant into the primary market faces a "chicken and egg" problem. Owing to consumers' desire for variety in complementary products, consumers prefer a primary good that already offers a wide array of complementary goods and services. At the same time, owing to economies of scale and sunk costs in the development of complementary products, developers of complementary

products prefer to develop products for primary goods that already have a large number of users. Thus, an entrant into the primary market “either has to offer consumers much lower value or has to incur large sunk costs to develop (or subsidize) a wide range of [complementary goods and services] before there is a large user base to purchase them.”¹⁰⁷

The conditions underlying this theory may be met in the Internet context. For exclusion in the complementary market to deter rivals from entering the market for Internet services, the exclusionary behavior in the complementary market must deprive a potential entrant to the Internet-service market of a source of complementary products. By excluding rival producers of Internet portals, content, and applications from its network, the monopolist network provider may be able to drive rivals from the nationwide markets for these products.

To deprive a potential entrant of a source of complementary products, the monopolist not only must drive rival content and application producers from the market; it also must deny access to its own content and applications to consumers outside its network.¹⁰⁸ Otherwise, a rival network provider’s customers could simply access the monopolist’s content and applications through the rival’s Internet service.¹⁰⁹ Hence, for a particular application or content, this strategy and the “primary good not essential” strategy are mutually exclusive.¹¹⁰

Thus, this theory is applicable only if an Internet service provider offers proprietary content and applications exclusively to customers of its Internet service¹¹¹ and if there are (perhaps owing to the exclusion of rivals) not enough remaining independent applications, content, or portals available that could be used by customers of rival network providers.¹¹² In this case, a new entrant into the market for Internet services must develop (or subsidize the development of) its own content or applications.

One may wonder whether the second condition mentioned in the preceding paragraph will ever be met in the Internet context. After all, portals, content, and applications are available to anyone using the Internet. But the condition may be met in the market for Internet services for mobile phones, or in emerging national markets in countries outside the United States. For example, there may not be enough independent applications or enough content adapted to the limitations associated with accessing the Internet from a mobile phone.¹¹³ And in a country that recently started adopting the Internet, there may not be enough independent applications or content in the country’s national language.

We might also imagine that consumers perceive certain applications and content—for example, search engines or e-mail—as indispensable

elements of Internet use. If these applications and this content are available only with the incumbent provider's Internet service, consumers may not consider an entrant's Internet service an adequate alternative unless the entrant offers a similar set of applications and content. In this case, to deter entry into the market for Internet services, the incumbent does not have to drive all existing independent applications, portals, and content from the market. All it has to do is restrict the exclusionary conduct to applications and content that consumers view as essential. Although there will be independent applications and content left that customers of a rival Internet service could use, the entrant will still be forced to enter the market for specific applications and content in order to compete in the primary market.¹¹⁴

For exclusion in the complementary market to deter rivals from entering the market for Internet services, the exclusionary conduct must not only deprive a potential entrant into the Internet-service market of a source of complementary products, but simultaneously entering the market for Internet services and the market for content or applications must be more difficult or costly than entering the market for Internet services alone. This requirement is fulfilled too. Simultaneous entry into both markets is more difficult or costly than entry into the Internet-services market if the two markets each exhibit at least one of the four characteristics described above. In the Internet context, all four characteristics are present: entry into both markets requires different capabilities, production in the two markets is subject to differing economies of scale, success in the different markets is uncertain, and, owing to the incumbent's exclusionary conduct, the provision of Internet service is subject to indirect network effects with respect to the individual provider's network.

Developing software applications or interesting content requires capabilities that are very different from the capabilities required to operate a network, and a potential entrant to the market for Internet services may not have both sets of capabilities.¹¹⁵ In addition, most of the cost of entry into those complementary industries consists of the sunk costs of developing the offering—costs that cannot be recovered in case of failure.¹¹⁶ Thus, if an entrant to the primary market is required to enter complementary markets as well, the risks associated with entry may be considerably greater than if it only enters the primary market. Taking these risks into account, investors probably will charge higher rates for firms who wish to access both markets simultaneously.

The market for Internet services and the markets for complementary products are subject to very different economies of scale. For example,

McKinsey estimated that assuming an average per-customer revenue of \$18.00 to \$22.50 in 2005, a broadband PC portal in Germany would need more than 8 million users to break even.¹¹⁷ In contrast, the economies of scale of building and operating IP networks are much lower.¹¹⁸

Although network technology is undergoing rapid change, a new entrant into the market for Internet services does not have to innovate, but can take advantage of existing technology. The development of applications and content is less predictable. If a potential entrant to the market for Internet services has to develop several applications and services in order to compete with the incumbent's Internet service, the uncertainty associated with each development reduces the likelihood of successful entry into the market for Internet services.

Internet service is subject to indirect network effects.¹¹⁹ The greater the number of applications and content that are available for users, the more valuable Internet service becomes. At the same time, the development of content and applications is subject to economies of scale.¹²⁰ Having more users allows developers of applications and developers of content to spread the fixed costs of development over a larger sales base, so that the more users they have, the higher the variety of applications and content and the lower their cost will be (under free entry into these markets).¹²¹

Technically, any application based on the Internet protocol can run over any network that supports the protocol, as long as the network is connected to the public Internet. As a result, the relevant network for indirect network effects is not an individual provider's network; it is the global Internet. Thus, Internet service providers compete under conditions of compatibility.

By excluding independent applications from its network and offering its own applications exclusively to its own Internet customers, an Internet service provider changes the nature of the competition from competition among compatible networks (in which the benefits resulting from adding a new user—i.e., the increased number of applications resulting from additional users—accrue to anyone connected to the Internet) to competition between incompatible networks (in which the benefits of adding a new user are limited to the customers of the new user's ISP). Thus, by using this strategy, the ISP reintroduces indirect network effects with respect to its own network.¹²² Application developers and content developers then have to decide whether to offer their product to the customers of the ISP with the "closed" network or to the customers of ISPs with an "open" strategy. Owing to economies of scale in the production of application and content, the developers will base their decision on the sizes of the different networks.

Because an entrant to the market for Internet services does not have any customers yet, it will have difficulty attracting application developers and content developers to write for its network rather than for that of the incumbent, which already has a large number of customers. Owing to the incumbent's strategy, the entrant will face the "chicken and egg" problem described above: consumers will not subscribe to its Internet service unless it has an attractive array of content and applications, and developers will not produce applications and content for its network without an attractive number of users.¹²³

Thus, a monopolist provider of Internet services may be able to deter entry into the Internet-services market by excluding rival producers of applications, content, and portals and offering its own content and applications exclusively to customers of its own Internet service.¹²⁴ This strategy may reduce consumers' valuation of the monopolist's Internet service, since excluding rival producers of applications, content, and portals reduces the variety of complementary products available. Thus, in deciding whether to employ such a strategy, the monopolist must trade off the loss in Internet-service fees against the gains in future monopoly profits.

Monopoly Preservation in the Complementary Market A monopolist may also use its monopoly over the primary good to protect a monopoly in the complementary market against dynamic competition. In this case, the exclusionary conduct in the complementary market preserves the monopoly in that market.¹²⁵

The structure of the model and the underlying reasoning are parallel to the "primary good not essential" case.¹²⁶ The monopolist has a monopoly in the primary market; the complementary market is subject to economies of scale or network effects. Some uses of the complementary good do not require the primary good. As a result, the complementary market comprises a "systems market" and a "stand-alone market." By excluding its rivals in the complementary market from the former, the monopolist can force them to exit the latter.

In the "primary good not essential" case, the monopolist uses this mechanism to extend its monopoly to the complementary market. In the case under consideration here, the monopolist uses the same mechanism to protect a legally acquired monopoly against emerging competition in the complementary market.

Such a scenario may be particularly relevant if the complementary market belongs to an R&D-intensive industry in which there is dynamic or "Schumpeterian" competition.¹²⁷ Owing to intellectual-property rights,

economies of scale, or network effects, R&D-intensive industries are prone to short-run exercises of market power. In other words, competition in these markets often leads to dominating of the market by a single firm. Firms in these industries typically compete “for the market,” not “within the market.” Though firms with market power (the winners of the competition) are an inherent feature of such industries, their dominance may be temporary. Rapid technological change and pivotal innovations may cause demand for their product to collapse quite suddenly. For example, rivals may come up with a vastly superior product or develop a new product that makes the incumbent’s product obsolete. Thus, the primary constraint on incumbents in these industries is dynamic competition—that is, innovation by other firms seeking to replace the existing firm. To avoid being dislodged by rivals, incumbents must innovate.

In the scenario described above, a monopolist could use its market power in the primary market to preserve the legally obtained market power in the complementary market, thereby distorting the dynamic competition for future market power. Instead of innovating to stay ahead of competitors, the monopolist could simply exclude rivals from the systems part of the complementary market, preventing them from ever growing large enough and reaching the scale or network size they must reach to displace the incumbent.

As was explained above, the conditions underlying this model are common in the Internet context. Network providers may be local monopolists in the market for Internet services, but may offer applications, content, or portals to consumers nationwide; these latter markets are usually subject to significant economies of scale—and, potentially, to network effects. In addition, at least some of these markets undergo rapid technological change. Not surprisingly, markets for software applications are the canonical example of R&D-intensive industries subject to dynamic competition.¹²⁸

Now consider a network provider that is a local Internet-services monopolist and has a dominant position in the nationwide market for a particular application. This monopolist has an incentive to exclude rivals from the market for that application to protect itself from dynamic competition.

This theory played an important role in the Federal Communications Commission’s evaluation of the merger between AOL and TimeWarner. TimeWarner owned a number of broadband cable networks, while AOL offered its instant-messaging program to consumers nationwide and held a dominant position in that market. The FCC was concerned that the merged firm could use its control over broadband cable networks to

disadvantage rivals seeking to overturn AOL's legally acquired monopoly in instant-messaging services. To alleviate this problem, the FCC approved the merger subject to a condition (among others) that required AOL Time-Warner to interoperate with instant-messaging competitors before offering "advanced" instant-messaging services.¹²⁹

Profitability of Discrimination without Monopolization When thinking about the profitability of exclusionary conduct, researchers commonly focus on the ability of and the incentives for a primary-good monopolist to monopolize the market for selected complementary products. This line of analysis is based on the implicit assumption that discrimination is profitable only if the primary-good monopolist monopolizes the complementary market. But, as will be shown in this section, this focus may be too narrow. A network provider may have an incentive to discriminate against an application even if the network provider fails to drive the application from the market.

Focusing on monopolization leads researchers to underestimate the likelihood of discriminatory behavior by network providers. If discrimination is profitable only if the network provider monopolizes the complementary market, then a network provider will discriminate only when it can expect to drive its competitors from the complementary market. If discrimination is profitable even if the network provider does not manage to monopolize the complementary market, it is much more likely to occur.

The following analysis will cover four of the six exceptions outlined above,¹³⁰ and is based on the assumption that depriving rivals of access to the network provider's Internet-service customers will increase sales of the network provider's complementary product. At least some of the network provider's Internet-service customers who, absent exclusion, would have used a rival's complementary product will use the network provider's offering instead. Thus, by excluding rival producers of applications or content from its network, the network provider gains additional sales from its Internet-service customers at the expense of its rivals. If the complementary product is subject to economies of scale or to network effects, and the network provider offers its complementary product to customers nationwide, exclusion from access to the network provider's Internet-service customers may force rivals to operate at an economically less efficient scale or with a smaller network of customers. This puts the rivals at a competitive disadvantage in the rest of the market too, potentially leading to even greater success of the network provider's complementary product in that part of the market.¹³¹

Will an increase in sales of the network provider's complementary product increase its profits, even if the network provider does not monopolize the complementary market in question? As we will see, the common theme that emerges from the analysis is that, in all the exceptions discussed below, the marginal benefits of serving an additional customer are larger than the marginal costs of serving the customer. As a result, every additional sale results in a net gain for the network provider, whether or not it monopolizes the complementary market.

This alone does not make exclusion a profitable strategy. The network provider has an incentive to exclude its competitors from a specific complementary market only if the net gains from the additional sales in the complementary market are larger than the associated reduction in profits in the Internet-services market.

More Sales at Market Prices In a perfectly competitive market subject to constant returns to scale, simply increasing the number of sales at the market price will not increase profits. In such an industry, long-run equilibrium prices equal marginal costs, resulting in zero profit per unit sold. As a result, a firm cannot increase its profits by making additional sales at the market price. Instead, it has to gain a monopoly position that enables it to raise prices above marginal costs.

Markets for applications, content, and portals are different. In these markets, exclusionary conduct need not result in a monopoly to increase the network provider's profits; it suffices if exclusion results in a larger number of sales.¹³² This is due to the cost structure underlying the production of applications and content; the production of these goods has high fixed costs and low marginal costs. Though the cost of developing the first instance of an application may be significant, the cost of producing additional copies may be small. Because of the need to cover fixed costs, these products are priced significantly above marginal costs.¹³³ In these circumstances, a firm need not charge monopoly prices to increase its profits; making additional sales at the market price may be sufficient.¹³⁴ More sales enable the firm to spread the fixed costs of production over more units, resulting in lower average costs per unit and a higher profit margin at the same price. Put differently, once a firm has made enough sales to cover the fixed costs, any additional sale at the market price only adds to the profits. For example, gross margins of 80 percent or 90 percent are common for computer software,¹³⁵ so in these markets any additional sales may lead to a significant increase in profits.

By excluding rival producers of complementary products from its network, a network provider gains additional sales of its own complementary product. If marginal profits are positive, these sales increase the provider's profit even if the rivals are not driven from the complementary market completely. This has important implications for the relevance of the "primary good not essential" exception. As we saw above, a network provider's ability to monopolize a nationwide complementary market by excluding rivals from access to its Internet-service customers will depend on a variety of factors, including the exact size of economies of scale with respect to the complementary product, the strength of potential network effects with respect to that product, and the size of both the monopolist's network and the remaining network.¹³⁶

Ultimately, monopolization may not always be a realistic prospect, but this does not automatically remove the network provider's incentive to discriminate, since monopolization is not necessary to increase the network provider's profits. As long as the exclusion of rivals from its Internet-service customers translates into more sales of its complementary product, and the additional profits are larger than the costs of exclusion, exclusion will be a profitable strategy. And because the conditions underlying the "primary good not essential" exception¹³⁷ are often met, exclusion may often be a profitable strategy.

More Outside Revenue As was explained above,¹³⁸ a network provider may have an incentive to monopolize a complementary market if the complementary product garners outside revenue that cannot be extracted in the market for Internet services.

Again, there are two variants of this exception. First, if the network provider offers its complementary product only to its own Internet-service customers, then, as we saw above, its outside revenue may still be higher if it excludes rivals and collects the outside revenue directly than if it tries to capture rivals' outside revenue by threatening exclusion. This increase in profit, however, is not dependent on a monopolization of the market for the complementary product. This is easy to see if the market for the complementary product is subject to network effects between the customer side and the advertiser side of the market, and if demand from advertisers increases faster than the amount of advertising space available. For example, Google collects more advertising revenue per search than its competitors in the search-engine market because of this type of network effect, and because of superior technology—not as the result of a monopoly position in the search-engine market. Similarly, the difference in revenue per

customer between websites that belong to a conglomerate that sells its advertising space directly and independent websites that sell their advertising space through advertising networks does not hinge on having a monopoly position in the respective markets for the complementary product.

In addition, through its billing relationship with customers of its Internet service, the network provider has data on customer demographics that may enable it to charge higher per-customer fees than many of its rivals in the market for Internet content, portals, and applications.¹³⁹ Again, this ability is not dependent on a monopoly in the complementary market.

If, as in the second variant, the complementary product is offered to customers outside the network provider's network, the provider cannot extract the advertising revenue that rivals realize from customers who do not use the provider's Internet service. The only way to get this advertising revenue is to convince consumers to use the network provider's complementary offering.¹⁴⁰ As we saw above, online advertising rates usually are set per 1,000 page views. Having more customers (and therefore more page views) results in higher advertising revenue even if the advertising rate stays the same. In view of the low marginal costs of serving an additional customer with content, applications, or a portal, the marginal advertising revenue from an additional customer in the complementary market probably will be larger than the marginal cost of serving that customer. Moreover, a larger customer base may enable the network provider to raise its per-customer advertising rate, thereby realizing even higher net gains. More customers make an online offering more attractive. If the number of interested advertisers grows more quickly than the amount of advertising space available, the advertising rate per customer may increase. As we saw above, the search-engine market exhibits such a network effect between the customer side and the advertiser side.¹⁴¹

Remember, though, that a network provider will have an incentive to exclude rivals in the complementary market from access to its Internet-service customers only if the increase in advertising profits from the larger number of customers in the complementary product outweighs the reduction in profit in the market for Internet services.

A similar argument applies to the variant of this exception, described above,¹⁴² in which a network provider excludes Internet-telephony providers from access to its Internet-service customers in order to preserve the outside revenue from access charges associated with traditional long-distance calls. This strategy will also be profitable even if the provider does not exclude the Internet-telephony providers from its customers

completely. Access charges are per-call charges set by regulators, and can still be charged even if some long-distance customers are lost. Every long-distance call lost to an Internet-telephony provider reduces profits; thus, the more conventional long-distance calls the network provider manages to keep, the higher its profits.

Monopoly Preservation in the Complementary Market In the “monopoly preservation in the complementary market” exception outlined above,¹⁴³ the network provider excludes rival producers of a complementary product from access to its Internet-service customers in order to preserve a legally acquired monopoly in the complementary market. This analysis assumes that the monopolist will be able to keep its rivals out of the nationwide market by excluding them from access to its Internet-service customers. Even if the monopolist is not large enough to keep its rivals out of the complementary market completely, exclusion from a part of the market may put them at a competitive disadvantage by making them operate at a less efficient scale or with a smaller network. Relative to a world without exclusion, this may slow the erosion of the network provider’s monopoly in the complementary market, extending the length of time that it can charge monopoly profits.¹⁴⁴

Network Provider Faces Competition in the Market for Internet Services

To this point, we have assumed that the network provider is at least a local monopolist in the market for Internet services. This assumption aligns with standard economic thinking about vertical exclusionary conduct in complementary markets: according to economic theory, an economic actor without monopoly power in the primary market will be incapable of excluding competitors from the complementary market by using vertical practices such as tying, vertical mergers, or exclusive dealing.¹⁴⁵ A monopoly in the primary market is therefore considered an indispensable precondition for successful monopolization of the secondary market.¹⁴⁶

On this theory, it is not surprising that most of the literature on vertical exclusionary conduct in markets for complementary products focuses on exclusionary conduct by monopolists. After all, the same conduct is not likely to pose any significant anti-competitive threat if the firm faces competition in the primary market.¹⁴⁷ This theory has also shaped economists’ evaluation of existing firms’ behavior in a complementary market. Allegations of anti-competitive conduct in a secondary market are often countered by evidence that the accused firm does not have monopoly power in the primary market.¹⁴⁸ Alternatively, the analysis of the monopoly case

is used as an argument *a maiore ad minus*: if a monopolist in the primary market lacks the ability and incentive to impede competition in the secondary market, it is argued, then a competitive firm's conduct will pose even less of a threat.¹⁴⁹

Based on this line of reasoning, most commentators believe that the threat of discrimination against independent providers of complementary products can be mitigated by competition in the market for Internet services.¹⁵⁰ Stated differently, it is usually assumed that competition in the market for Internet services will restrict a network operator's ability and incentive to discriminate against independent content, portals, or applications. As we saw above, this assumption is the basis of the proposal to increase facilities-based competition and the proposal to introduce open-access regulation (regulation that would require the owners of broadband networks to allow independent ISPs to offer their services over these networks), both of which seek to mitigate the threat of discrimination by increasing competition in the Internet-services market.

The following analysis shows that the above assumption is not correct. A network provider may have the ability and incentive to exclude rival content, applications, or portals from its network, even if it faces limited competition¹⁵¹ in the market for Internet services.¹⁵² Apart from increasing the number of cases in which unaffiliated providers of complementary products face a real threat of discrimination, this result also implies that neither facilities-based competition nor open-access regulation is an appropriate tool for mitigating this threat.¹⁵³ Three arguments drive this conclusion. First, in the Internet context, the ability to exclude competitors from a complementary market (the market for a particular application, content, or portal) is not dependent on a monopoly position in the primary market (the market for Internet services). Instead, the power to exclude is conferred by network technology. Second, realizing the benefits of exclusion (an increase in profits—or, sometimes, a preservation of current profits) does not require monopolizing the primary market. The absence of a monopoly in the primary market actually increases the network provider's incentive to increase profits through exclusionary conduct in the complementary market, since the provider cannot extract the available monopoly profit by charging higher prices in the primary market. Third, owing to the existence of switching costs or to the network provider's ability to discriminate rather than exclude, the exclusion of rivals will not necessarily cause the network provider's Internet-service customers to switch providers, and that makes the costs of exclusion lower than is commonly assumed. As we will see, these factors confer a degree of market power on

the network provider, making competition in the market for Internet services less effective.

The following analysis assumes that the network provider competes with at least one other network provider.¹⁵⁴ In addition, the network provider may offer content or applications. A particular application or particular content may be offered to all consumers or exclusively to the customers of the network provider's own Internet service.

Ability to Exclude In our hypothetical network, as in today's Internet, a network provider can distinguish between different applications and different content running over its network, and can control their execution. This technology lets network provider exclude selected complementary products or slow their execution.

Although the exclusionary power of the technology does not extend beyond the provider's network, exclusion from the provider's Internet-service customers may drive rival producers of a complementary product from the nationwide market for that product if economies of scale or network effects are operating.¹⁵⁵ Whether rivals will be driven from the market is contingent on the size of the economies of scale with respect to the complementary product in question, on the strength of any network effects with respect to that product, and on the numbers of the monopolist's Internet-service customers and the customers of other network providers.¹⁵⁶ Thus, in this context, the ability to drive competitors from the nationwide market for the complementary product depends on the network provider's nationwide market share in the Internet-services market. Again, a monopoly position in this market is not required.

Benefits of Exclusion In a variety of cases, exclusionary conduct will increase, or preserve, a network provider's profits in the complementary market. This effect does not require a monopoly in the market for Internet services, nor, as we saw above, does it require the network provider to gain a monopoly in the complementary market.¹⁵⁷ Instead, the lack of monopoly in the primary market constrains the network provider's ability to extract profits in the Internet-services market, making the opportunity to realize profits in the complementary market even more attractive. As a result, exclusion may be profitable more often than is commonly assumed.

In general, if a network provider excludes rival producers of a complementary product from access to its Internet-service customers, it will increase the sales of its own complementary product.¹⁵⁸ As was explained above, this increase in sales often leads to an increase in profits. In the

cases outlined above, the increase in profits results from more sales, not from the ability to charge monopoly profits. Thus, to be profitable, the exclusionary conduct need not drive rivals from the complementary market entirely.

More Sales at Market Prices In the “more sales at market prices” exception,¹⁵⁹ the increase in profits resulting from the higher number of sales in the complementary market was driven by the specific cost structure of the markets for applications, content, or portals. These markets are characterized by high fixed costs and low marginal costs and this cost structure is not affected by the existence of market power in the market for Internet services.¹⁶⁰

More Outside Revenue In the “more outside revenue” exception, the increase in profits resulted from the logic of pricing in the advertising markets.¹⁶¹ The network provider realized higher outside revenue by selling access to a large group of its Internet-service customers directly instead of letting rival producers of complementary products sell access to smaller groups of customers and then extracting the outside revenue from the rivals. Again, no monopoly is required.

There is evidence that some Internet service providers do, in fact, try to reduce the amount of time their customers spend on content or portal offerings not affiliated with their service. For example, in the AOL–Time Warner merger proceeding, the FCC found that “[t]he record in this proceeding provides some evidence that AOL already seeks to limit its members’ access to unaffiliated content on the World Wide Web. For example, AOL requires that content appearing on AOL web sites have only a limited number of hyperlinks to unaffiliated content.”¹⁶² This made it more likely that a user who clicked on a link within the AOL portal remained within the AOL portal, where AOL benefited from the advertising revenue or sales commissions generated by this user.

In the variant of this exception,¹⁶³ the network provider was interested in excluding VoIP providers from access to its customers because it could charge access charges only for long-distance calls placed using the conventional telephone service, not for those using VoIP. (Access charges are per-call charges set by regulation; they do not depend on a monopoly in the market for Internet services.)

Monopoly Preservation in the Complementary Market In the “monopoly preservation in the complementary market” exception,¹⁶⁴ the ability to

preserve the monopoly depended on the exact size of economies of scale, the strength of potential network effects, and the nationwide number of both the monopolist's Internet-service customers and the customers of other network providers. A monopoly in the market for Internet services was not required.

Costs of Exclusion Relative to the monopoly case, the existence of other, competing network providers may increase the costs of exclusionary behavior in the complementary market. Owing to the ability to use discrimination instead of exclusion and the existence of switching costs, the costs of exclusion will still be lower than is commonly assumed.¹⁶⁵

If the network provider is the only supplier of Internet service in a particular geographic area, consumers have no other way to get access to the excluded application or content. Here, the costs of the exclusionary behavior are twofold. First, owing to the reduction in the variety of applications and content, consumers may value the network provider's Internet service less without the exclusion, which may force the network provider to reduce the price of its Internet service.¹⁶⁶ Second, without being able to use the excluded application or content, some consumers may not value the network provider's Internet service enough to pay the lower price.¹⁶⁷ Because the pricing of the service already reflects the reduced value, the number of lost transport customers will probably not be high.

If the provider competes with at least one other network provider, consumers who want access to the excluded application may switch providers. As these consumers do not have to forgo Internet services altogether, the number of lost transport customers will probably be higher than if the excluding network provider faces no competition. Thus, competition increases the costs of exclusionary behavior in the complementary market.¹⁶⁸

The market for Internet services, however, exhibits a number of characteristics that make competition less effective than is commonly assumed.¹⁶⁹ These factors may limit the costs of exclusionary behavior in spite of competition in the market for Internet services.

First, if the exclusionary conduct drives the producers of the excluded application or content from the market, switching providers will not give consumers access to the excluded product. As a result, fewer consumers will switch as a result of the exclusion.¹⁷⁰

Second, if all network providers block the same application, there is no provider for consumers to switch to. (For example, in many countries all mobile network providers block Internet-telephony applications to protect

their revenue from mobile voice services, leaving customers who would like to use Internet telephony over their wireless Internet connection with no network provider to turn to.)

Third, the network provider may be able to engage in exclusionary conduct without losing too many of its Internet-service customers by using discrimination instead of direct exclusion. For example, it may slow a specific application instead of blocking it.¹⁷¹ If the network provider blocks a certain complementary product, its Internet-service customers are not able to use the application or to access the blocked content, and they will notice this interference. In contrast, if the network provider merely slows a specific product, its customers may not notice that the network provider is discriminating against specific complementary products, and therefore will see no reason to switch to another provider. If a network provider discriminates against a rival's complementary product by slowing it, the discrimination will work indirectly. Consumers' use of the rival's product will be less satisfactory than their use of the provider's offering, and this will lower their perception of the quality of the rival's product. Because many consumers will not be able to detect the true cause of the lower quality, they may attribute it to poor design. For example, a slow gaming experience might be attributable to bad application programming, insufficient server capacity at the gaming site, or slow Internet transport. The results would look identical to most consumers. Similarly, long waiting times for pages from an online shop could result from bad programming of the underlying databases or from insufficient server speed. Customers may blame the online shop, especially if they do not usually have problems with network speed. With discrimination, consumers can choose the rival's product, but they prefer the network provider's product because they perceive it to be of higher quality. Unlike customers who face direct technical exclusion or tying, customers who face discrimination will not think their choice has been restricted. Thus, by using discrimination, a network provider can exploit customers' incomplete information about the true source of poor performance. That customers' having imperfect information can provide market power to an economic actor who faces competition in the primary market by enabling the actor to impose restrictions in a complementary market that it would not be able to sustain if the primary market was perfectly competitive is well established in the economics literature.¹⁷² Recent behavior by Comcast, the largest provider of broadband over cable in the United States,¹⁷³ indicates that network providers do, indeed, use discrimination in the manner just described. Comcast surreptitiously interfered with the operation of BitTorrent, a popular peer-to-peer file-sharing

application, on its network. Instead of announcing that it did so, or consistently blocking the use of the application, Comcast sent forged data packets (i.e., packets that look as if they were sent by the other end host) to both end hosts involved in a connection, telling them to sever the connection. As a result, both end hosts terminated the BitTorrent connection. In some cases, this merely delayed a connection; in other cases, it prevented a file from transferring.¹⁷⁴ By using forged packets, Comcast hid its interference with the connection from users. When asked (by a reporter, and later by the Electronic Frontier Foundation) whether Comcast was interfering with BitTorrent, Comcast representatives stated that Comcast “isn’t deliberately blocking, degrading, interfering with, or discriminating against particular protocols or kinds of traffic [and] isn’t using network management techniques that are designed to disrupt anyone’s use of BitTorrent (or any other application).”¹⁷⁵ As a result, users, journalists and public-interest organizations had to expend considerable technical effort to understand what Comcast was doing and trace BitTorrent’s unusual behavior back to Comcast’s intervention. As white papers produced by Sandvine (the equipment vendor Comcast was using) show, Sandvine (and Comcast, by using the equipment in this way) chose this method of interference to prevent customers from noticing it.¹⁷⁶ The use of file-sharing applications is an important driver for broadband adoption, and network providers do not want to lose customers who wish to use file-sharing applications.¹⁷⁷ In sum, if the network provider discriminates against rival products instead of excluding them directly, competition in the market for Internet services does not increase the costs of the exclusionary conduct.

Fourth, even if the network provider’s Internet-service customers know that the provider is restricting their access to applications, content, or portals, the cost and time involved in switching Internet services may still prevent many consumers from changing providers.¹⁷⁸

Switching costs are the costs a customer incurs when switching to a competitor.¹⁷⁹ When deciding whether to switch, customers take these costs into account. A consumer will switch if the present value of switching is larger than the present value of staying with his or her current provider. Switching costs reduce the present value of switching, which reduces customers’ willingness to switch (as compared to a world without switching costs). Switching costs make consumers’ demand less elastic, enabling a provider to charge a higher price.¹⁸⁰ Similarly, switching costs allow a provider to impose other restrictions that it could not impose in a perfectly competitive market. Whether these costs will prevent a customer from switching depends on the value the customer places on the excluded

application and on the magnitude of the switching costs. The higher the switching costs, the fewer customers a network provider loses as a result of its exclusionary conduct.¹⁸¹

Switching costs may be substantial in the market for Internet services. Consider first the obvious financial expenses that may be associated with switching providers. A customer who has a long-term contract with a provider will be charged a termination fee.¹⁸² When switching from a broadband-over-cable service to a digital-subscriber-line (DSL) service, a consumer will be charged for installation and will have to buy a DSL modem and other new equipment. If (as is common in the United States) the Internet service is bundled with television and telephony, cancellation of the Internet-service portion of the bundle may result in a loss, or a partial loss, of the bundle discount, and the loss of that discount may then be a significant ongoing financial cost for the consumer.¹⁸³

Further, switching providers may require a customer to invest a significant amount of time and effort. She will have to open an account with the new provider and close her account with her present provider.¹⁸⁴ If she cannot install the access hardware and software herself (which takes time and expertise), she must stay at home for the installation. If she has used other provider-specific offerings available only to her old provider's Internet-service customers (e.g., stock portfolio tracking, instant messaging, or customized news pages), she will have to find alternative offerings and customize them. If she has been using an e-mail address offered by the network provider, she will have to notify various people of her new e-mail address, perhaps have new stationery and business cards printed, update her résumé and her website, and bear the risk of missing e-mail messages sent to the old address.¹⁸⁵ The precise cost of switching e-mail addresses is difficult to measure, but anecdotal evidence indicates that customers view it as substantial. The *New York Times* reported in 2005 that AOL had about 5 million customers who paid \$14.95 per month in order to keep using an AOL e-mail address even though they had switched to another broadband-access provider and paid Internet-service fees to the new provider.¹⁸⁶

In addition, the Internet-service offerings of various providers differ substantially in price, quality, and other characteristics. Therefore, they are not interchangeable. For example, broadband cable networks have different technical characteristics than DSL networks, which differ from broadband offerings provided over fiber networks. Switching in response to the exclusion of a complementary product would require a customer to switch from his most preferred offering to another offering that may meet fewer of his needs.

The trend toward bundling differentiates the market further. Cable customers may not think of the digital or satellite television service offered by phone networks as a perfect substitute for their cable television; on the other hand, customers of a conventional telephony provider may not trust the digital telephony offered by cable companies. Though it is possible to switch only the Internet service and keep the other offerings, this will significantly reduce the bundle discount. The problem is further exacerbated if the network provider offers exclusive content that is valuable to the customer.¹⁸⁷ For example, Deutsche Telekom currently has the exclusive right to transmit games of the Bundesliga, the premier German soccer league, over the Internet.¹⁸⁸ If it offered the streaming of Bundesliga games exclusively to its Internet-service customers, soccer fans would have to value the ability to use the excluded applications at least as much as the opportunity to watch soccer games over the Internet.¹⁸⁹ Thus, product differentiation in the market for Internet services and in the market for bundles of telephony, television, and Internet gives network providers an additional degree of market power over their Internet-service customers.¹⁹⁰

The exact costs of switching may depend on the circumstances. For example, some customers may use provider-independent e-mail services, such as Hotmail or Gmail; others may not subscribe to a bundle at all. The details of contracts vary too. For example, some customers are not subject to a long-term contract, or a competing provider may waive the installation fee. In countries with open-access regulation where regulation allows independent Internet service providers to offer their services over other providers' networks, customers may be able to switch to another provider that offers its services over the same physical network; that removes the need to buy new equipment.

A particular Internet customer may face any combination of the switching costs discussed above. And every customer has to go through the process of choosing an alternative provider and installing and setting up the access software. These hurdles alone may suffice to deter switching. The cost of searching for and comparing alternative offerings is an important switching cost. Moreover, empirical studies show that the decision to switch depends on the perceived costs of switching, which are not necessarily equivalent to the actual costs. Empirical studies in the UK's market for long-distance telephone service have shown that providers were significantly more likely to retain dissatisfied customers who perceived the switching costs as high than dissatisfied customers who perceived them as low.¹⁹¹ According to empirical studies in the long-distance and credit-card industries, the perceived costs of switching are significantly increased if

the product is perceived as complex, if it has a large number of features, or if it is bundled with other products.¹⁹² Thus, customers in an industry whose services are viewed as complex and are characterized by many features and by bundles, such as Internet services, will perceive switching costs as high.

Finally, research in behavioral economics indicates that even very small costs may prevent customers from switching. Individuals exhibit a “status quo bias”—they are much more likely to keep what they already have than rational-choice theory would predict.¹⁹³ For example, this bias is exploited by free trials that automatically convert to a paid subscription at the end of the trial period unless the customer calls or writes to prevent this.¹⁹⁴ If, however, the costs of placing a call or writing a letter are sufficient to prevent people from acting, the significantly higher actual (or perceived) costs of switching Internet service providers may prevent many Internet service customers from switching providers, even if their Internet service provider excludes applications or content they would like to use.

The problem of incomplete customer information is reduced by rules that require network providers to disclose whether, and how, they interfere with applications and content on their networks. But such regulations cannot reduce the perceived or actual switching costs in the market for Internet services and still leave the network provider with substantial market power over its customers—power that enables it to restrict some applications and content without losing too many customers. Similarly, while open access regulations may reduce some of the costs of switching (since customers may not have to switch to another network technology), the costs are still substantial enough to prevent customers from switching in response to exclusion. This suggests that neither disclosure requirements nor open access regulations may be sufficient to mitigate network providers’ incentives to discriminate, even if they are competing with at least one other network provider.¹⁹⁵

Other Incentives to Discriminate or Exclude

Discrimination to Control Bandwidth Network providers may also have an incentive to block applications so as to control the use of bandwidth on their networks.¹⁹⁶ Networks are designed with certain patterns of use in mind. For example, “last-mile” networks (the networks that connect users’ premises to their network provider) usually offer much more bandwidth from the Internet to the home than from the home to the Internet. This design is based on the assumption that users will receive much more data

than they send. As we saw in chapter 2, this reflects the demands of client-server applications (such as Web browsing) that were the dominant application class at the time when asymmetric bandwidth was deployed, but creates problems for peer-to-peer applications such as file sharing, video conferencing, Internet telephony, or online gaming, which send and receive equal amounts of data. The design of today's last-mile networks also assumes that users' patterns of Internet use differ enough to enable a number of users to share the connection. The models assume that people don't all use the Internet at the same time, and that even if they do the traffic will be "bursty"—that is, will exhibit uneven patterns. On the basis of these considerations, the link between the cable modem at the end user's home and the provider's cable modem termination system (which communicates with subscribers' cable modems to give them access to the Internet) is shared by a number of users; the exact number is determined by the access provider on the basis of statistical calculations about expected use.¹⁹⁷ If users behave as expected, they will get the predicted bandwidth. If all these users want to use the maximum advertised bandwidth at the same time, they will not be able to get it—a phenomenon known as "oversubscription."

If Internet use increases, or if the patterns of use differ from the expected patterns, the performance of the network may worsen. To restore lost performance, the network provider may have to invest in additional bandwidth capacity in its network, incurring an increase in capital expenses. Higher-than-expected Internet use may also increase a provider's operational costs. If the traffic leaves the provider's network, the provider may have to pay higher interconnection fees to the networks that further transport the traffic.¹⁹⁸

If use of the Internet increases, or if the patterns of use change, network providers may have an incentive to block or slow certain applications so as to maintain the performance of their network while avoiding the higher costs associated with network upgrades or higher interconnection fees. This incentive may be especially strong if the provider's Internet-service business is based on flat-rate pricing.¹⁹⁹ Because the provider's Internet-service customers pay the same price regardless of the amount of data they send and receive, increased use drives up the provider's costs, but not its revenue.

For example, in 2007 Comcast was shown to interfere with Gnutella, BitTorrent, and other file-sharing applications in a way that slowed or stopped file sharing.²⁰⁰ Peer-to-peer file-sharing applications challenge traditional assumptions about use: they often send and receive continuous

streams of data over longer periods of time (violating the assumption that traffic will be bursty, discontinuous and asymmetric), and they often download or upload data to and from computers on other networks (increasing the network provider's interconnection charges).²⁰¹ The Comcast example offers a number of insights. First, it invalidates the common argument that providers would refrain from blocking or excluding applications or content on the notion that more applications and content make the network more attractive. Peer-to-peer file sharing is one of the most popular applications on broadband networks and an important driver for broadband adoption.²⁰² This did not stop Comcast from blocking these applications; it only motivated Comcast to do so secretly.²⁰³ Second, it demonstrates that discrimination or exclusion may happen even if the network provider does not have a monopoly in the market for Internet services; in many US cities, Comcast competes with the local telephone company that offers DSL. As we saw above, using methods that are difficult to detect reduces the likelihood that customers will switch to another Internet service providers in response to the blocking, since they do not realize that their network provider interferes with the application.

The incident has also fueled the debate over network neutrality in the United States. Many network-neutrality proposals, as well as the Internet Policy Statement adopted by the Federal Communications Commission in 2005,²⁰⁴ contain an exception for reasonable network management. Proponents of network neutrality claim that singling out specific applications to manage bandwidth on a network is not reasonable network management and should be forbidden under a network-neutrality regime. They do not dispute that network providers may have to engage in traffic management to manage bandwidth on their networks, but they argue that traffic-management techniques should be non-discriminatory.²⁰⁵ Opponents of network neutrality would like to exempt any practices employed to manage bandwidth on the network, even if the network provider applies them only to a specific application or even a specific application provider.²⁰⁶ In August 2008, after a complaint and a petition for a declaratory ruling that Comcast's conduct violates the FCC's Internet Policy Statement by public-interest groups and academics (including the author of this book), the FCC ruled that Comcast's conduct did not constitute "reasonable network management" and that it violated the Internet Policy Statement.²⁰⁷ Comcast has appealed the decision.

Content-Based Discrimination A network provider may also have an incentive to block, or discriminate against, certain content. For example,

providers may have an incentive to block messages harmful to the company's interests. For example, in July 2005, Telus, Canada's second-largest Internet service provider, blocked access to the "Voices for Change" website. At the time, Telus and the Telecommunications Workers Union were involved in a contentious labor dispute, and the "Voices for Change" site, run by a union member,²⁰⁸ offered online discussion forums for union employees to discuss strategies during the strike. Telus argued that online posts suggesting that striking workers jam Telus's phone lines hurt the company, and that the site's inclusion of pictures of employees crossing picket lines threatened the privacy and safety of these employees. The company restored access to the site after it had obtained "an injunction that prohibited the posting of photos that might intimidate or threaten anyone connected with the dispute."²⁰⁹ (See box 6.6.)

In April 2006, AOL's e-mail servers temporarily blocked e-mails that contained the URL "dearaol.com."²¹⁰ Dearao.com had been set up by an alliance of nonprofit and public-interest groups organized by MoveOn.org (a liberal advocacy group) and the Electronic Frontier Foundation (an Internet civil-liberties group) in order to protest AOL's plans to adopt CertifiedEmail, a program that would have required high-volume e-mail senders to pay fees in exchange for guaranteed preferred delivery and circumvention of spam filters. The site contained an open letter to AOL and an online petition that users could sign to protest CertifiedEmail. AOL later attributed the blocking to a "software glitch" and removed it within a day.

Network providers may also have an incentive to impose other restrictions on the content transported over their network. For example, in September 2007, Verizon Wireless rejected a request by NARAL Pro-Choice America, an abortion-rights group, to let it send text messages over Verizon Wireless's network using a five-digit code.²¹¹ If Verizon Wireless had granted the request, Verizon Wireless's customers could have sent a short text message to this code to sign up for messages from NARAL; NARAL periodically sends text messages to those who have signed up to inform them about pressing issues or to ask them to take political action. Other wireless providers had complied with NARAL's request, but Verizon Wireless rejected it, arguing that it "does not accept issue-oriented (abortion, war, etc.) programs" and that it would refuse service to "any organization that seeks to promote an agenda or distribute content that, in its discretion, may be seen as controversial or unsavory to any of our users." After the *New York Times* reported the incident in a front-page article, Verizon Wireless reversed its decision, noting that the decision was based on an

Box 6.6

The Telus Incident—A Case of Justified Self-Help?

According to a letter written by a lawyer representing the union member who operated the website Voices for Change, the site contained some content that would have justified an injunction.^a However, this does not justify the blocking of the site. There are very clear legal procedures (such as getting a court order to get the content removed) for realizing that goal. Every person or entity threatened by online information needs to follow these procedures. A network provider should not have different rights just because it is technically able to engage in some sort of self-help. In fact, the Telus case highlights some of the reasons why network providers should not be allowed to engage in self-help through blocking.

First, in comparison with an injunction, blocking is both over-inclusive and under-inclusive. According to investigations by the OpenNet Initiative at the time of the blocking, by banning access to the IP address of the server on which the website was hosted, Telus not only blocked access to Voices for Change; it also blocked access to 766 other websites hosted on the same server.^b And blocking Voices for Change affected not only the illegal content but also the many legitimate discussions that were taking place on the site.^c At the same time, the blocking wasn't an effective way of preventing access to the illegal information; it merely prevented customers of Telus's Internet service (and customers of ISPs who were downstream from Telus) from accessing the site directly. Even Telus's customers were still able to get access through a proxy site or one of the mirror sites that had been set up.

Second, the relevant legal rules assign the right to decide which content is illegal to a court that has no interest in the matter. As the case of Telus shows, an entity whose own interests are at stake may have an incentive to interpret more liberally what should be removed. Telus consistently justified the blocking not just with the threat to employees' safety, but also with the online discussions about jamming Telus's customer service lines that were taking place on the site.^d These additional concerns related to Telus's business interests featured prominently in other statements by the company.^e However, neither the settlement with the operator of the website nor the injunction against the Telecommunications Workers Union that Telus obtained in court addressed these business-related concerns. Though there may be other reasons, this suggests that these concerns were not able to withstand judicial scrutiny.

All this suggests that, while there was some illegal content on Voices for Change, blocking the site was not the right method to get it removed.

- a. Letter by McGrady, p. 2 (Penmachine 2005b).
- b. OpenNet Initiative 2005.
- c. Letter by McGrady, p. 2 (Penmachine 2005b).
- d. CBC News 2005.
- e. See the letters cited in Penmachine 2005a.

“incorrect interpretation of a dusty internal policy” against spam messaging and receiving unwanted messages. Verizon Wireless maintained that it had a right to decide what messages to transmit. Though this incident concerns a service offered over the wireless mobile network, not over the wireless mobile Internet, an analogous motivation could be attributed to Internet service providers.

In August 2007, AT&T deleted some lyrics from a webcast of a Pearl Jam concert at the Lollapalooza festival in Chicago: “George Bush, leave this world alone” and “George Bush, find yourself another home.” The webcast was streamed through AT&T’s website “The Blue Room,” which offers webcasts of live concerts. AT&T, through an agency, hires contractors to monitor the performances. The live webcasts are delayed for a few moments to enable the contractors to censor material that violates AT&T’s content policy. AT&T later called the Pearl Jam incident “a mistake.” According to an AT&T spokesman, AT&T’s “policy is not to edit or censor performances.” The contractors are “told only to edit for excessive profanity, not really for the songs, but the banter going on between band members or band members and the audiences, as well as any nudity that can arise” in order to protect younger visitors to the website.²¹² Though this is not a direct example of a network provider restricting content on its network (AT&T acted in its role as a content provider at the application layer), it is easy to imagine a virtually identical incident in which a provider enacts a content policy and restricts content that flows through its network.

The last two examples illustrate the two types of problems associated with content policies. First, the content policies themselves may prevent end users from accessing certain content and prevent affected content producers from reaching their audience (as in the Verizon Wireless example). In addition, enforcement of the content policy may result in deliberate or inadvertent misclassifications, resulting in the exclusion of additional content (as in the AT&T example).

A number of states require their network providers to block access to content that the state singles out.²¹³ Mainly, this includes political content perceived as posing a threat to the current government or to national security, social content perceived to violate local norms and morals, and content describing or providing access to Internet tools that could be used to circumvent restrictions imposed by the state. For example, Bahrain, China, Ethiopia, Libya, Iran, Myanmar, Pakistan, Saudi Arabia, Syria, Thailand, Tunisia, Uzbekistan, and Vietnam all block websites that report on political opposition to the existing government. Many countries in the Middle East, including Iran, Oman, Saudi Arabia, Sudan, Tunisia, and

the United Arab Emirates, also filter extensively for content deemed to be religiously, culturally, or socially inappropriate (e.g., content related to pornography, dating, homosexuality, or gaming).

Conclusion on Ability to Discriminate

Although a network provider does not generally have an incentive to discriminate against independent providers of applications or content, the analysis above has highlighted a number of cases in which it may have an incentive to do so in order to increase or preserve its profits, manage bandwidth on its network, or exclude unwanted content. A network provider may have an incentive to discriminate even when it faces competition, not only when it has a local monopoly in the market for Internet services.

Whether the conditions giving rise to such an incentive are present in a real-life situation is a question for empirical inquiry. In most cases, however, the network provider need not gain a monopoly in the complementary market to make exclusion a profitable strategy; thus, the threat of discrimination is more relevant than is commonly assumed.

In most cases, a network provider need not exclude all independent developers of complementary products from its network in order to increase profits. Often it will be profitable to exclude only the complementary products that compete directly with one of the provider's own products.²¹⁴ This reduces the costs of exclusion, as the reduction in complementary goods variety is restricted to the products that are actually excluded.

Owing to the specific characteristics of markets for applications and content (such as the cost structure of information goods and—sometimes—the existence of network effects), the exclusion of rivals may lead to gains significantly higher than the gains that can be achieved in traditional markets. As a result, it is more likely that gains from exclusion will exceed associated costs, which makes it more likely that exclusion is a profitable strategy.²¹⁵

Effect on Incentives to Innovate

The preceding subsection highlighted conditions under which a network provider in our hypothetical discriminatory network, and in today's Internet, may have an incentive to exclude independent producers of applications, content, or portals from access to its Internet-service customers. When these conditions are present, independent producers of complementary products face a real threat of discrimination.

The threat of discrimination reduces the amount of application-level innovation by independent producers of complementary products in three ways.

First, when the conditions for profitable exclusion outlined above are present in a particular complementary market, a network provider will discriminate against rivals in that market. As was noted above, discrimination will reduce a rival's profits.²¹⁶ A potential innovator bases its decision to innovate on the expected costs and benefits of realizing the innovation; a potential innovator in an affected market will expect lower profits if it anticipates discrimination.

Second, the profitability of exclusion depends on many factors that may not be common knowledge for all participants in a market. As a result, an economic actor with an idea for a complementary product may not know whether a network provider will have an incentive to discriminate against the innovator's final product.²¹⁷ As a result, potential innovators face uncertainty about their future competitive environment. This uncertainty may reduce a risk-averse developer's incentive to innovate, even when the factual conditions for profitable exclusion are not present.

Third, the above analysis suggests that independent producers of complementary products need not be concerned that a network provider will exclude their products in order to increase its profits if it does not offer a competing product. (As we saw above, a network provider's incentive to exclude applications or content to manage bandwidth or to exclude unwanted content is independent of whether the network provider offers a similar complementary product.) This seems to imply innovation will be harmed only if the provider is already vertically integrated into a complementary market. But economic theory shows that this is not correct.²¹⁸ Even if the provider does not currently offer a competing product, it may be tempted to let the entrant "test the waters" and then—if the entrant begins to achieve success—imitate the entrant and exclude it from the network. Foreseeing this course of events, the independent producer refrains from entering the market. As a result, no one enters the complementary market. (There is a region of forgone invention in which privately and socially beneficial innovations are not realized.)

For the third problem to occur, three conditions must be realized.²¹⁹ First, demand in the complementary market must be uncertain. Second, in the presence of demand uncertainty, entry into the complementary market must be attractive for the independent producer but not for the primary-good monopolist (e.g., owing to cost heterogeneity). Third, the

primary-good monopolist must have a selling advantage in the complementary market.

In the context of the Internet, these conditions often are met. First, in markets for new applications or content there usually is considerable uncertainty in demand. Second, the literature of economics and business strategy highlights several reasons why an incumbent network provider may not have an incentive to enter a complementary market for a new product in the presence of uncertain demand, whereas an independent producer may have such an incentive. (For example, start-ups often have lower entry costs than an incumbent, owing to the different cost structure of incumbents and new entrants.²²⁰) Third, the ability to technically exclude a rival producer of complementary products from its network provides the network provider with a huge selling advantage in the complementary market. Thus, the number of markets in which independent developers' incentives to innovate are reduced will be larger than the exceptions outlined above imply.

In an end-to-end network, technical exclusion is not a strategy available to network providers. As a result, independent developers of complementary products need not fear discrimination or technical exclusion. Consequently, their incentives to innovate will be higher than in a discriminatory network.

Whereas the threat of discrimination reduces independent developers' incentive to innovate at the application level, network providers' incentives to innovate at this level rise as a result of the increase in profit that discrimination brings. As we will see in chapter 8, under the conditions present in today's Internet the increase in application-level innovation by network providers cannot offset the reduction in innovation by independent producers. Thus, the threat of discrimination reduces the amount of application-level innovation.

These insights have important implications for the current policy debate regarding the regulation of broadband network providers. The analysis in this chapter has highlighted a variety of circumstances under which a network provider may have the ability and an incentive to exclude, or discriminate against, independent content, portals, or applications. Thus, there is a real threat that network providers will engage in this kind of conduct, and that threat bolsters arguments made by proponents of regulation. Contrary to common belief, however, this threat is not necessarily mitigated by competition in the Internet-services market. Rather, as the analysis in this section has shown, a network provider may have an incentive to exclude or discriminate even if it faces competition. This finding

suggests that neither increased facilities-based competition nor open-access regulation is an appropriate policy response to that threat. Finally, a network provider may have an incentive to discriminate against unaffiliated applications or content even if they do not compete with one of its own applications—for example, to manage bandwidth on its network, or to exclude unwanted content. This makes it less likely that existing anti-trust rules can protect innovators from all discriminatory conduct that may occur.²²¹

Effect on Pricing Strategies

The difference in application awareness and application control between a network based on the broad version of the end-to-end arguments and the hypothetical network in my analysis enables the network provider in the latter network to pursue three pricing strategies not available in the former network. First, a network provider in a discriminatory network can charge end users different transport prices for different applications. Second, a provider in a discriminatory network can use its power to exclude applications to price discriminate between Internet-service customers. Third, a provider in a discriminatory network may be able to charge application and content providers to have their applications and content function on its network.

Price Differentiation

In a discriminatory network, the network provider can monitor which applications run. As a result, it can set different individual transport prices for each application.²²² In contrast, an end-to-end network is application-blind; a network provider cannot base transport prices on the application being used, only on observable characteristics such as bandwidth. Thus, in end-to-end networks there will be a single transport price²²³ for all applications.

Whereas a provider operating a discriminatory network can set a transport price for each application to maximize transport revenue from that application, a provider in an end-to-end network can only maximize the overall transport revenue from all applications.²²⁴ As a result, the network provider in an end-to-end network may be forced to forgo consumer surplus from use of a single application to capture more consumer surplus from all applications. For this reason, a provider in an end-to-end network can capture less (or, at most, the same) consumer surplus from a particular application than a provider in a discriminatory network.²²⁵

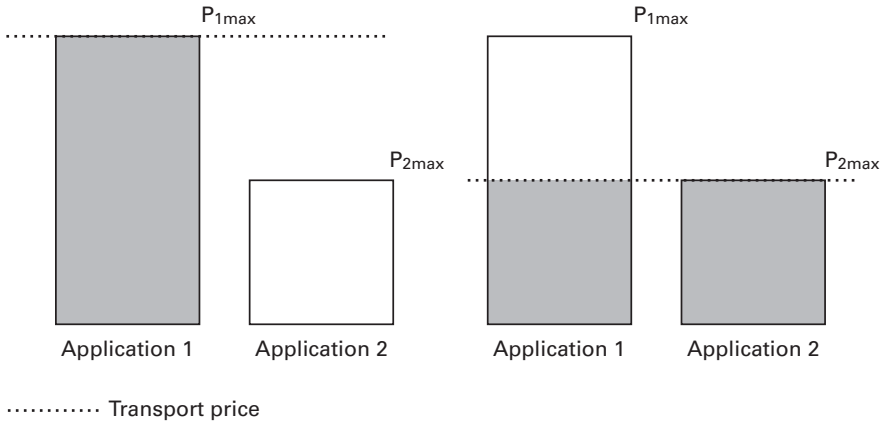


Figure 6.2
Options for setting a uniform transport price.

Consider a network with two applications that have equivalent bandwidth requirements. The network provider is a monopolist in the Internet-services market. The applications are competitively supplied at a price equal to marginal cost, which is assumed to be zero. Assume that consumers value the first application more than the second (figure 6.2).²²⁶ In this example, the network provider in an end-to-end network has two options. One option is to forgo transport revenue from the second application and choose the highest possible transport price at which consumers are willing to use the first application. Let us call this price p_{1max} . At this price, the network provider captures all consumer surplus from the use of the first application but earns no transport revenue from the second application (figure 6.2, left). Alternatively, the network provider can set the transport price low enough to earn transport revenue from both applications. The resulting price is the highest possible transport price at which consumers are willing to use the second application. Call this price p_{2max} . This price enables the provider to earn transport revenue from both applications (figure 6.2, right). While the provider extracts all consumer surplus associated with the second application, it can only capture a fraction of the surplus that consumers derive from using the first application. As a result, consumers keep a surplus of $p_{1max} - p_{2max}$ with respect to the first application. The exact form of consumers' preferences will determine which strategy maximizes the network provider's profits.

In contrast, the network provider in a discriminatory network can choose different transport prices for the two applications: it sets the transport price

for the first application at $p_{1\max}$ and the transport price for the second application at $p_{2\max}$, gains transport revenue from both applications, and captures all consumer surplus. The resulting transport revenue is greater than the transport revenue under each of the options described above.

The difference in available pricing strategies has implications for application developers' incentives to innovate.²²⁷ Consider the surplus a consumer would derive from a particular application if it paid for transport but got the application for free. If the application is offered in the end-to-end network at all,²²⁸ this surplus is higher (or at least the same²²⁹) in an end-to-end network than in a discriminatory network. This surplus, however, denotes the maximum surplus an application developer can extract through its pricing of the application.

Thus, in an end-to-end network, more consumer surplus is available for extraction by application developers. Consequently, relative to a discriminatory network, an end-to-end network may earn application developers higher profits, thereby increasing their incentives to innovate.²³⁰

This analysis applies only if the network provider has some market power in the market for Internet services. If it does not have the ability to profitably raise the transport price above marginal cost, then the bandwidth-adjusted transport price—and, consequently, the consumer surplus available for extraction by application developers—will be the same in both networks.

Price Discrimination

Available Strategies The owner of a discriminatory network can also use its power to exclude applications from its network to price discriminate between customers. A provider in an end-to-end network lacks this option, since it cannot detect which applications run over its network and cannot control their execution.

Price discrimination can maximize profits by setting prices charged to different consumers on the basis of their individual willingness to pay.²³¹ Imagine two groups of potential customers. Group 1 has a low willingness to pay for the product; group 2 has a high willingness to pay. If the seller has to set a uniform price for all customers, it has two options: it can set a low price and sell to all customers, or it can set a high price and sell only to customers with a high willingness to pay. Which option is more profitable will depend on the sizes of the two groups and on their respective willingness to pay, but regardless of this the seller cannot capitalize on the differences between individual consumers' willingness to pay. In contrast,

if it was able to charge members of group 1 a lower price than members of group 2 it could sell to each group of customers at their maximum willingness to pay, thus extracting more consumer surplus and earning more profit than if it had to charge a uniform price.

To practice price discrimination, a seller must have some market power (i.e., ability to profitably set price above marginal costs).²³² Otherwise, it would not be able to charge any consumer more than the competitive price. Thus, the network provider must be an oligopolist or a monopolist to use the following strategy.²³³

By excluding certain types of applications from its network and enabling their use for a higher price, a provider in a discriminatory network can practice “second-degree” price discrimination.²³⁴ Under this type of discrimination, a seller knows that there are groups of consumers willing to pay different amounts, but the seller cannot identify which group an individual consumer belongs to. To deal with this, the seller offers products that differ in features and in price and that target different groups of consumers. The differences between products are designed to induce an individual consumer to choose the product intended for his or her group; the price for a product is based on the targeted group’s willingness to pay.²³⁵ As a result, the seller can capture more of each group’s consumer surplus than if it sold only one good at a uniform price.

To practice second-degree price discrimination, a network provider may use consumers’ eagerness to use specific types of applications as an indicator of consumers’ willingness to pay for Internet services.²³⁶ The provider offers two types of Internet service: a low-priced version that enables consumers to use the Internet but blocks a particular type of application, and a high-priced version that enables consumers to use all applications. If this strategy is to be used as a method of price discrimination, consumers’ desire to use the blocked application must be positively correlated to their willingness to pay for Internet services (or, stated differently, the consumers eager to use the application must be those willing to pay high amounts for Internet services). Under these conditions, the provider can separate consumers into two groups willing to pay different amounts, thus allowing it to price different types of Internet service in a way that extracts as much surplus from each group as is possible.

To see how this might work, consider an example.²³⁷ Telecommuters (employees who work from their homes) usually connect to the employer’s computer system using an application called a *virtual private network*. In contrast, a typical home user, who uses the Internet in her free time to surf the Web and check e-mail, probably does not consider virtual private

networks an essential element of Internet use. Additionally, telecommuters generally are willing to pay more than home users for Internet services. If a provider disables the use of virtual private networks in the “basic” version of Internet service and allows them in a “premium” version, telecommuters will choose the premium version, which can be priced to exploit their higher willingness to pay. Users who do not value the ability to use virtual private networks enough to pay for the premium version will choose the basic version. As a result, the network provider can capture more consumer surplus than if it was forced to offer only one version of Internet service at a uniform price.

This method does not require the provider to monopolize the complementary market. Customers of the “premium” service are free to use whatever application they prefer. The monopolist does not have to participate in the market for the “blocked” application; it simply blocks basic users from accessing applications that fall into this category.

Implications for Innovation In a discriminatory network, but not in an end-to-end network, a network provider can use the strategy described above to price discriminate between consumers in the market for Internet services. The difference in available strategies affects application developers’ incentives to innovate if their profits will be lower with discriminatory pricing than without it. This question hinges on detailed knowledge about consumers’ preferences and market and cost structures, but some general observations are possible.²³⁸

We just saw that, relative to the profit-maximizing uniform price, the discriminatory pricing always increases the monopolist’s profit from Internet services.²³⁹

Discriminatory pricing policies introduce distortion into the overall market for Internet applications by disadvantaging certain classes of applications.²⁴⁰ Whereas developers of the blocked application can sell only to buyers of the premium version of Internet services, developers of other applications can sell to all Internet-service consumers. Thus, for producers of the blocked application, discriminatory pricing results in a smaller market—and potentially in lower profits. Their profits will be reduced if there are consumers of the basic version who would have bought the application if there was a uniform price for Internet services but who do not value it enough to pay the higher price for premium service in the non-uniform pricing scheme. These consumers are also worse off under the network provider’s discriminatory pricing policy,²⁴¹ because they are not able to use an application that they value. Two conditions must be

met for this situation to occur. First, there must be some consumers who choose the basic version of Internet services in the non-uniform pricing scheme and who are not priced out of the market for Internet services under the uniform price. These consumers buy Internet services under both pricing schemes, but are excluded from the market for the blocked application under the non-uniform pricing scheme. Second, under the uniform price for Internet services, it must be profitable for the blocked application's developers to sell their application to some of these consumers, using either a uniform or a non-uniform pricing scheme—for example, by offering a basic version to consumers with a low willingness to pay and a premium version targeted at consumers with a higher willingness to pay.²⁴²

In addition, the discriminatory pricing in the market for Internet services may decrease the profits of all applications' developers, because the non-uniform pricing scheme enables the network provider to capture more consumer surplus in the market for Internet services than if it was forced to charge a uniform price. As a result, the fraction of consumer surplus available for extraction in the complementary market will decrease.²⁴³

Charging Providers of Complementary Products

A network that deviates from the broad version of the end-to-end argument may also contain functionality that would let a network provider charge fees to providers of complementary products for the right to reach the provider's Internet-service customers. These fees could take many forms. For example, Comcast could charge Yahoo a one-time access fee for the right to be accessed by Comcast's Internet-service customers, Comcast could demand a share of the profits that Yahoo realizes from Comcast's Internet-service customers, or Comcast could charge a fee every time one of its Internet-service customers accesses Yahoo. The Internet does not currently offer the fine-grained accounting infrastructure that would be necessary to implement the last of these options, but a network provider may be able to implement some access charges using its ability to exclude: the network provider could threaten to exclude a specific application from its network unless the application's provider pays an access charge. Some network-neutrality proposals would ban these types of access charges.²⁴⁴

Network providers' ability to charge providers of complementary products for the right to be accessed by the network provider's Internet-service customers transforms the market for Internet access services into a two-sided market.²⁴⁵ In such a market, one or more platforms (here, the Internet-service offerings) enable interactions between two types of users, which form the two sides of the market. In our case, Internet-service customers

(one side of the market) would like to use applications or content, which are offered by providers of the complementary product (the other side of the market). Not every market that enables buyers and sellers to interact constitutes a two-sided market.²⁴⁶ For a market to be two-sided, the volume of interactions between users on both sides must depend on the structure of fees charged each side, not only on the overall level of fees charged by the platform. This condition is met if users on one side of the market cannot directly pass on the fees charged by the platform provider (for example, because transaction costs prevent this). In the Internet context, application providers and content providers may not be able to pass network providers' access charges on to end users; the transaction costs associated with setting up and administering a payment infrastructure for micropayments on the content provider side, and entering payment-related information on the end-user side, may be prohibitively high.²⁴⁷

A complete analysis of the implications of network providers' ability to charge content or application providers is beyond the scope of this book; however, two observations can be offered. First, charging content or application providers enables network providers to directly extract some of the profits that application and content providers make, so this strategy directly reduces the profits of providers of applications and content. Second, research on two-sided markets indicates that the profits of these providers may be reduced disproportionately, because the network provider has an incentive to charge monopoly prices to them. The provision of Internet transport is a two-sided market in which end users are "single-homers" (that is, they are only connected to one network, their Internet access network), whereas application and content providers are "multi-homers" (they can reach customers on all access networks). The literature on two-sided markets predicts that in a market where one side is single-homing and the other is multi-homing, the platform owner (in our case, the network provider) will charge monopoly prices to the multi-homing side even if the platforms compete for the single-homers.²⁴⁸ Thus, the literature predicts that network providers will charge monopoly prices to content and application providers even if there is competition on the end-user side of the market for Internet services.²⁴⁹

A platform owner has a terminating monopoly over access to its single-homing customers. This monopoly will exist whether or not single-homers have several platforms from which to choose. If a consumer gets his broadband access through AT&T, a content provider such as Yahoo can reach him only through the AT&T access network. This gives AT&T a monopoly over access to the consumer, which enables AT&T to charge monopoly

prices to content and application providers even if AT&T is competing with other network providers for Internet-service customers.

Charging monopoly prices to content and application providers reduces their profits and may reduce their incentives to innovate below the social optimum.²⁵⁰ If the resulting reduction in content innovation and application innovation reduces consumers' demand for Internet services, an individual network provider does not bear the full cost of this reduction; some of the reduced demand will affect competing network providers or network providers in other local markets. As a result, by increasing access charges for content providers, the network provider exerts a negative externality on other network providers that prevents it from fully internalizing the effect of higher access charges on content provision and leads it to set inefficiently high access charges.

If platforms compete for single-homers, this may affect how much of the monopoly profits that a platform makes on the multi-homing side will be passed on to the single-homing side, but it will not reduce the prices charged to multi-homers.²⁵¹ Thus, any problems that this pricing structure creates for the provision of content and applications will not be eliminated by increasing competition in the Internet-services market.

Conclusion on Pricing Strategies

A network provider in a discriminatory network can set different transport prices for different applications, and can also exclude applications from running over its network in order to price discriminate between customers. Additionally, a provider in a discriminatory network can charge content and application providers directly. These pricing strategies are not available in end-to-end networks.

Since these strategies may reduce the profits of independent developers of complementary products (relative to the profits they would earn in the absence of these strategies), the difference in available pricing strategies reduces independent developers' incentives to innovate. All strategies increase the profits of the network provider in the market for Internet services.

Conclusion

A discriminatory network offers network providers opportunities for strategic behavior against independent developers of complementary products, and increases the range of available pricing strategies in the market for Internet services.

Although a network provider does not generally have an incentive to discriminate against independent providers of content, applications, or portals, as this chapter has highlighted, in many circumstances a network provider has an incentive to use the power provided by the architecture to engage in noncooperative strategic behavior in order to increase its profits. In most cases, exclusion of all independent developers of complementary products is not necessary for a network provider to increase its profits. Instead, it will often be profitable to exclude complementary products that directly compete with one of its own complementary products. The network provider may also have an incentive to exclude complementary products that do not compete with one of its own products—for example, to manage bandwidth on its network or to exclude unwanted content. In any event, the exclusion will usually reduce or eliminate the profits of the producers of an excluded good.

Similarly, whereas the pricing strategies available in a discriminatory network are capable of increasing a network provider's profits, they usually reduce the profits of independent developers of complementary products, thus reducing their incentives to innovate.

A discriminatory network may enable a network provider to gain more profits in the market for Internet services or complementary products than an end-to-end architecture. The same strategies that increase the network provider's profits will usually reduce the profits of at least some independent developers of complementary products. Thus, independent developers of applications, portals, and content will tend to earn higher profits under an end-to-end architecture.

IV The End-to-End Arguments and Application Innovation

7 Network Architectures and the Economic Environment for Application Innovation

Chapters 4–6 highlighted three mechanisms by which an architecture may influence economic actors' incentives and ability to innovate. By affecting the transaction and coordination costs associated with organizing initial component development and subsequent innovation in arm's-length relationships, an architecture affects the ability of independent economic actors other than system architects to innovate (chapter 5). An architecture's modifiability influences the cost of developing an innovation, which not only affects incentives to innovate but also may determine *who* can innovate (chapter 4). The range of strategies an architecture supports influences the potential benefits from innovating—and these potential benefits, in turn, affect the incentives to innovate (chapter 6). Chapters 4–6 also used these insights to examine the effect of the Internet's original architecture, highlighting the many ways in which an architecture can influence the economic environment for innovation. Together, these influences determine the overall effect of architecture on innovation.

Drawing on these insights, the present chapter explains how the gradual transition from a network architecture based on the broad version of the end-to-end arguments to a core-centered network architecture that sits at the other end of the spectrum from pure end-to-end to fully deviating from end-to-end changes the economic environment for application-level innovation. By assessing how deviations from the broad version of the end-to-end arguments affect independent innovators' and network providers' incentives to innovate, and how they affect other aspects of the economic environment for innovation such as the size and diversity of the innovator pool or the control over the development and deployment of applications, this chapter will help us understand how deviations from the broad version of the end-to-end arguments affect the nature and the amount of application-level innovation.

Potential Deviations from the End-to-End Arguments

An architecture can deviate from the broad version of the end-to-end arguments along two dimensions: it can become more “opaque” by implementing more application-specific functionality in the network’s core, or it can become more “controllable” by increasing network providers’ ability to control applications and content on their networks. As we will see, changes along these two dimensions affect the economic environment for innovation in different ways.

The broad version of the end-to-end arguments requires the lower layers of the network to be very general; they should not be optimized in favor of specific applications. Moving along the first dimension increasingly optimizes the network for existing applications by placing more and more application-specific functionality in the network’s core. As we saw in chapter 2, the network engineers developing these changes and the network providers deploying them may move in this direction to optimize the performance of the network or save costs in the short term while disregarding the effects of their actions on the long-term evolvability of the network. In the Internet, the deployment of asymmetric DSL service and the introduction of firewalls and network-address translators are examples of changes along this dimension.¹

As we saw in chapters 2 and 3, increasing the amount of application-specific functionality in the network makes it more difficult to deploy new applications. The more application-specific functionality the network’s core contains, the more likely it is that new applications cannot be deployed without first making changes to the network. Whereas an architecture based on the broad version of the end-to-end arguments results in a network that is transparent for new applications, changes along this dimension make the network more opaque.

While changes along the first dimension aim to increase performance, changes along the second dimension aim at increasing control. A network based on the broad version of the end-to-end arguments is application-blind; as a result, network providers are not able to see which applications are using their networks and then control their execution. Network architects move along this dimension by adding application-specific functionality in the network’s core that increases network providers’ ability to monitor and control what is happening on the network. Small movements in this direction may involve adding devices that make the network more application-aware, thereby enabling the network provider to monitor (but not control) the network at an increasing level of granu-

larity. Further movements would add tools for control that would become increasingly powerful and granular. In the Internet, the introduction of deep packet inspection—that is, the deployment in the network of devices that can look into data packets, determine the application or content whose data the packets are carrying, and process the packets on the basis of that information—is an example of movement in this direction. Network providers move in this direction mostly to increase their profits. However, as we saw in chapter 2, changes that let a network provider monitor what is happening on the network may also help the network provider to make the network more secure, to manage traffic on the network, or to assess the evolution of the network's use for the purpose of planning upgrades.²

Generally, moving in this direction does not reduce the network's ability to support new applications. Apart from the application-specific functionality needed for monitoring and control, the network is still general and still able to support a large variety of applications. But, contrary to a network based on the broad version of the end-to-end arguments, the network provider is able to block some or all applications. Thus, in general, the network does not have to be changed to allow the deployment of new applications. However, applications blocked by a network cannot run on that network unless the network provider removes the block.

The Four Network Architectures

To better understand how deviations from the broad version of the end-to-end arguments affect the economic environment for application-level innovation, let us focus on four generic network architectures that differ with respect to the nature and the amount of application-specific functionality within the network. All architectures are assumed to provide the general functionality of an Internet; that is, they enable computers attached to different physical networks to communicate with one another. Starting from an architecture based on the broad version, we will first move along the second dimension and gradually increase the controllability of the network; we will then move along the first dimension and increase the opaqueness of the network (figure 7.1). In line with the analysis so far, let us assume that the users control which applications are installed on their end hosts.

The first architecture is based on the broad version of the end-to-end arguments and is therefore application-blind and open for new applications. I will call this architecture *end-to-end architecture*.

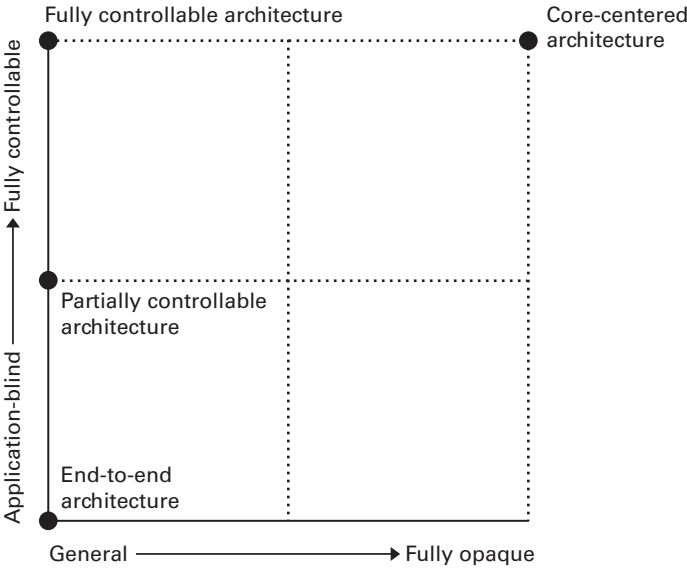


Figure 7.1
The four network architectures.

The second and third architectures deviate from the end-to-end arguments by increasing the network provider’s ability to monitor and control applications on its network. I will call these architectures *controllable architectures*. Though technically open for new applications, they include application-specific functionality that can distinguish between different applications and control applications’ execution. Architectures of these kinds, which can discriminate among specific applications or content, figured prominently in chapter 6.

The second architecture, which is “partially controllable,” is generally open to new applications, but can be used to block them selectively; the third, which is “fully controllable,” is generally closed to new applications, but can enable them selectively. To put this differently, the operator of a network having the second architecture must act to *block* new applications, while the operator of a network having the third architecture must act to *enable* them. In addition, the “fully controllable” architecture enables the network provider to charge application-specific prices to users or content providers, providing the opportunity to implement sophisticated pricing schemes.³ Relative to the end-to-end architecture, we can think of the “fully controllable” architecture as being located on the other end of the

spectrum from complete application-blindness to full awareness of and control of applications.

Finally, in addition to the functionality present in the fully controllable architecture, the fourth architecture contains a considerable amount of application-specific functionality in the core of the network. Owing to this functionality, the network is not generally open to new applications. Deploying a new application usually requires extensive changes to the core of the network. I will call this architecture *core-centered architecture*. Compared to the end-to-end architecture, it occupies the other end of the spectrum from full transparency to total opaqueness and from complete application blindness to complete awareness of and control of applications.

Analysis

Independent Application Developers' Incentives to Innovate

Independent application developers' incentives to innovate depend on the expected costs and benefits of innovation. Different network architectures may afford different costs and benefits. Innovators bear the costs of development, testing, production, and distribution. In addition, there may be costs associated with the deployment and with the ongoing operation of an application—e.g., negotiating network access and paying access fees. The expected benefits of an innovation depend on how an innovator expects to use her innovation. For example, if an innovator expects to benefit by selling the innovation to others, the benefits are the expected profits; if she expects to benefit primarily from using the innovation herself, the benefit is the expected increase in utility that the innovation offers.

In general, moving from the end-to-end architecture to the core-centered architecture increases the relative costs of innovation and decreases the relative benefits for independent innovators. As a result, independent developers' incentives to innovate decrease.

Developing and testing new applications in an end-to-end architecture requires no physical capital apart from investing in access to an end host, access to the application programming interfaces of the network protocols, and programming tools. Thus, the main costs of development under the first architecture are the costs of designing, programming, and testing the new application. For a non-commercial innovator, the only cost will be the opportunity costs of the time spent designing, programming, and testing the application. For a firm, the costs are those associated with paying employees to perform these tasks. Since applications in an end-to-end

network consist of software that needs to be implemented only on end hosts, the applications can be produced and distributed over the network at almost no cost. Because the network is open to new applications, the network provider need not incur any costs to enable the deployment of the application.

Because the network cannot distinguish among applications or control their execution, an application developer does not have to consider the possibility that the network provider may discriminate against her application. Neither does she have to consider the possibility that her negotiations for access may break down, or that her profits may be reduced by access fees. Finally, as we saw in chapter 6, since in the end-to-end architecture the network provider is not able to set application-specific transport prices for users, more consumer surplus is available for extraction in the complementary market than in an architecture in which the network provider can set application-specific transport prices. Thus, under the end-to-end architecture an independent application developer can expect to reap more of the benefits of an innovation than she can expect to reap under one of the other architectures.

Since the partially controllable network is able to discriminate against applications, a developer must take into account the possibility that the network provider will discriminate against her application. (As was discussed in chapter 6, there are many situations in which a network provider may have an incentive to do so.) Relative to an end-to-end architecture, this reduces the expected benefits, reducing the developer's incentives to innovate.

Moving from the partially controllable to the fully controllable network increases the costs of developing and testing the new application and reduces the expected benefits. As the fully controllable network blocks new applications by default, the new application cannot simply be tested on the operational network. Instead, the developer must have access to a test network. In addition, the developer must negotiate access to the network (which may require costly legal support) and may have to pay an access fee. If the network provider chooses to charge application-specific transport prices to users, less consumer surplus will be available for extraction by the application developer. The developer must also consider the possibility that bargaining will break down and the possibility that the network provider will discriminate against his application. If this occurs, the application will never be able to access the network, and the developer will not benefit from the innovation at all. These possibilities all reduce the profits that a developer can expect to reap from an innovation.

Finally, as was explained in chapter 4, the costs of developing and testing a new application will be significantly higher in a core-centered network whose core has a lot of application-specific functionality than in an end-to-end network. Often the network's core will have to be changed to enable the new application to operate. As a result, the total costs of innovation are substantially higher than under any of the other architectures; they include not only the costs of developing and testing the application itself, but also the costs of developing and testing the changes to the network's core and the subsequent changes to applications that relied on the former version of the network's core. Because the application cannot run on the operational network, the innovator must have access to a test network. If the changes to the network's core include changes to hardware, the new hardware must be manufactured and physically distributed. Thus, the costs of production and distribution also may be higher under this architecture. Finally, the developer may incur considerable costs while trying to convince the network provider of the usefulness and commercial viability of the new application and negotiate any necessary changes to the network's core and to other existing applications. She also must consider the possibility that she will fail to convince the network provider of the usefulness and commercial viability of the application; accounting for this possibility reduces the expected benefits. Ultimately, these costs may be so overwhelming that independent innovators lose the ability to innovate.

Network Providers' Incentives to Innovate at the Application Level

Moving from an end-to-end network to a fully controllable network gradually increases the network provider's incentives to innovate. The transition to a core-centered network, however, sharply reduces his incentives to innovate—probably below the level of incentives in an end-to-end network.

For the network provider, the costs of innovation are the same in each of the first three networks. The network provider's own applications are not affected by the network's ability to block new applications. A network provider does not have to negotiate access or pay access fees to get access to his own network; he can simply remove the block. In addition, a network provider can test his new applications over the operational network.

At the same time, the benefits that network providers can expect to reap from innovation increase from the end-to-end architecture to the fully controllable architecture. In the two controllable networks, the network

provider can discriminate against competing applications in order to improve the competitive positions of his own applications. As a result, his expected profits increase. The fully controllable network offers more benefits than the partially controllable network, because the fully controllable network enables a network provider to extract users' surplus by setting application-specific transport prices and to extract some of the profits that application and content providers make by setting sophisticated access charges.

Moving from the fully controllable network to the core-centered network sharply reduces the network provider's incentive to innovate, owing to the significant increase in the costs associated with developing a new application.⁴

Range of Potential Innovators

As was noted above, moving from an end-to-end network to a core-centered network increases the requirements that potential innovators must meet. Whereas developing applications for use in an end-to-end network requires only programming knowledge and access to an end host, developing applications for use in the other networks requires additional investments and capabilities. In the partially controllable network, in the fully controllable network and in the core-centered network, application developers may have to negotiate access or even sue the network provider for access, which can be difficult or impossible without legal support. In the fully controllable network and in the core-centered network, application developers must have access to a test network, as the operational network blocks new applications by default. They also must have funds (or higher ongoing profits) to pay any access fees. Finally, an application developer in a core-centered network may need detailed knowledge about the operation of the network's core in order to assess the feasibility of her application and develop the necessary changes. In addition, she must be able to convince the network provider that her application is valuable enough to merit changes to the network, and she must have sufficient manpower and processes to coordinate the development and deployment of the application and the necessary changes to the network with the network provider.

Thus, whereas in an end-to-end network the pool of potential innovators includes anyone with knowledge of a programming language and access to an end host, subsequent networks reduce the size and diversity of the innovator pool. The first developers to lose the ability to participate in innovation may be non-commercial innovators who lack the capability

to negotiate access, to pay access fees, or to gain entry to a test network. Thus, the move to a more controllable network results in an increasing professionalization of innovation. Small firms with little capital may also lose the ability to innovate as the controllability or opaqueness of the network increase. Finally, in a core-centered network, the technical and coordinational obstacles to application development may become so large that network providers are the only remaining potential innovators.

Control over Application Innovation, Deployment, and Use

As we saw in chapters 5 and 6, the owner of an end-to-end network can neither prevent nor impede development, deployment, or use of a new application. Instead, each application developer independently decides on her approach to innovation. Since users can deploy an application simply by installing it on their end host, application deployment and (consequently) the success of the resulting applications are in the hands of users.

In the two controllable architectures, users deploy an application by installing it on their end hosts; in the fully controllable architecture, the network provider must also participate in deployment by unblocking the application before it can run on the network. Although the network provider does not have to participate in deployment in the partially controllable architecture (because the network is open to all applications that are not blocked), it can influence an application's chances of being deployed by discriminating against new applications or excluding them completely. Through the same mechanism, a network provider in the two controllable architectures can influence the development of applications, as the threat of discrimination reduces independent innovators' expected profits and consequently reduces their incentives to innovate. Since network providers can monitor and control the applications running over their network, they control the use of the network.

In the core-centered architecture, network providers control the development of new applications, since their power over the network makes them the only remaining potential innovators. They also directly control the deployment of new applications, since deployment requires changes to end hosts and to the network's core. Owing to the controlling functionality in the network, network providers are able to control the use of the network. In a purely opaque architecture, network providers have some control over the use of the network through their control over the deployment of new applications. Once changes to the network that enable a new application to run have been made, an opaque network does not give the

network provider additional control over the extent to which users actually use the applications.

Thus, the transition from an end-to-end network to a more controllable or more opaque network shifts control over application development from developers to network providers, and shifts control over the deployment and the use of applications from users to network providers. As the controllability or the opaqueness of the network increases, control over application-level innovation and network use becomes more centralized.

Relative Costs of Application Deployment

In an end-to-end network, deploying a new application (that is, making all the changes to the operational network and to the end hosts that have to be made before the application can be used for the first time) requires no changes to the network's core.⁴ New applications can be deployed incrementally by installing them at an end host. Starting in the third network, deploying a new application requires action by the network provider to unblock the application. The associated costs will be rather low. In addition, users must install the application at an end host. Finally, deploying a new application in a network that implements a large amount of functionality in the network's core probably will require costly changes to the network's core. Users cannot deploy an innovation incrementally; the network provider must change the network first. If the deployment context consists of numerous network providers (as in the commercial Internet), all networks must be changed before the application can run across the Internet.

Thus, the transition from an end-to-end network to a core-centered network also increases the relative costs that economic actors other than the innovator must incur before they can begin to use the new application. This has two implications. First, it increases the threshold for the overall benefit an innovation must provide to make adoption worthwhile. Second, it may affect what kinds of applications will be deployed under the various architectures. As we saw in chapter 4, the architecture and the economic system in which the network is used jointly influence what innovations will be deployed. A network's architecture affects what components will have to be changed to deploy an innovation (that is, what components have to be changed before the innovation can be used for the first time) and what the costs and benefits of deployment will be. The economic system in which the network is used influences who controls the components that must be altered and what incentives these actors may have to make these changes. For example, in the commercial Internet, end hosts

are generally controlled by users, whereas the network's core is controlled by many different providers. Thus, in an end-to-end network with a deployment context similar to the commercial Internet, the small costs of deploying an application are borne by users who gain the immediate benefit of being able to use the application. Thus, only users must be convinced of the potential usefulness of the application; at the same time, the low costs of downloading and installing new applications make it easy to try new things. Because a new application can be deployed incrementally, users can decide whether to deploy it one by one. This is particularly important for innovations that take time to spread (for example, because they require users to develop new practices before the application is useful). In contrast, in an opaque network facing the same deployment context, network providers may have to make substantial investments to enable an application to run on their network—and must do so before it is clear whether users will even value the application enough for the provider to recoup her investments. These differences make it much more likely that new and unproven applications will be deployed in an end-to-end network than in a non-end-to-end network, owing to the uncertainty associated with their adoption. Similarly, applications catering to small, widely dispersed audiences may be deployed in an end-to-end network, but not in a non-end-to-end network. In the end-to-end network, only the users interested in these applications have to make changes. In contrast, the small number of potential users of the application among a specific network provider's Internet-service customers may not promise enough benefits to the network provider to justify the costs of changing the network, even if the overall number of potential users across all network providers would be large enough to justify the development of the application.

In sum, other things being equal, the different network architectures result in very different economic environments for innovation in applications. Before we can assess how these differences affect the overall amount, quality, and type of innovation, we need to understand how differences in the size and the diversity of the innovator pool and differences in control over application innovation and deployment influence how much innovation will occur. This is the topic of the next chapter.

8 Decentralized versus Centralized Environments for Application Innovation

As we saw in chapter 7, the transition from a network architecture based on the broad version of the end-to-end arguments to a core-centered network architecture gradually changes the economic environment for innovation at the application level. As the amount of application-specific functionality in the network increases, independent application developers' incentives to innovate can be expected to decrease. Network owners' incentives to innovate at the application level increase with the ability to control the network, but then decrease sharply as the rising amount of application-specific functionality in the core of the network increasingly prevents the deployment of new applications without extensive changes to the core of the network. The range of possible innovators decreases too—from anyone with access to an end host and the ability to learn a programming language (in an end-to-end network) to mostly commercial innovators able to negotiate and finance access to test networks and operational networks (in controllable networks) to the network owners (in the core-centered network). At the same time, control over the development and deployment of innovations in applications and the ability to innovate at this level gradually shifts to the owner of the network. Finally, the costs of adopting new or improved applications increase, both for users and for network owners.

Thus, moving from an end-to-end architecture to a core-centered architecture gradually reduces the number of potential innovators and changes the environment for application-level innovation from decentralized to centralized. Under the network architecture based on the broad version of the end-to-end arguments, independent application developers as well as network operators independently identify and realize opportunities for innovation. Users independently choose the applications they want to use. Thus, network operators are not able to control independent application developers' approaches to innovation or their ultimate success in the

Table 8.1

Decentralized versus centralized environments for application innovation.

	End-to-end network	Core-centered network
Control over application innovation	Application developers	Network providers
Control over application deployment	Users	Network providers
Innovator pool	Large and diverse	Small and homogeneous

marketplace. At the other extreme, under the core-centered architecture, only the network operators can engage in innovation at the application level. Because they have to change the network before an application can be deployed, they also determine which applications are made available to users. As a result, control over application-level innovation and deployment is centralized in the hands of the network operators.

This chapter focuses on how the differences in the number and type of innovators and in the different allocations of control over development and deployment of innovation affect the nature and the amount of innovation (table 8.1). Does decentralized innovation by many innovators offer specific advantages that cannot be achieved by a potential increase in centralized application-level innovation by a few network providers?

Effects of Differences in Innovator Diversity and in Control over Innovation

The Value of Many Innovators

To isolate the consequences of centralized innovation (with only network providers innovating) vs. decentralized innovation (with many more potential innovators), suppose the costs of innovation under the end-to-end architecture and the core-centered architecture were the same. How would the differences between centralized and decentralized innovation affect the amount and type of innovation?¹

If we look at the question through the lens of neoclassical economic theory, the differences in the number and the type of potential innovators may not appear particularly relevant.² In line with the usual behavioral and cognitive assumptions used in neoclassical economic theory, neoclassical models usually assume that all actors know all potential opportunities for innovation, and only need to choose among them. To account for the inherent uncertainty of innovation, the models assume that actors know and agree on the probabilities of the various outcomes a particular project

and a particular approach may produce.³ Given this information, potential innovators are assumed to choose between different projects and approaches by conducting a cost-benefit analysis. The theory assumes that a potential innovator will choose the project (or the set of projects and approaches), that maximizes the expected net benefit.⁴ These assumptions imply that it is possible to determine *ex ante* how many innovation projects, and which ones, should be pursued.

In the world of these models, there is not much value in having different economic actors perform this choice. After all, other things being equal, their decision will be the same.⁵ Given these basic assumptions, it is not surprising that in many neoclassical models parallel innovation by different economic actors constitutes a wasteful duplication of efforts that would better be avoided.⁶ As a result, central control over innovation appears beneficial, because it enables a coordinated approach to innovation that avoids duplication of efforts.⁷

Evolutionary and neo-institutional theories of economic change reach a different conclusion. According to them, decentralized innovation by diverse economic actors leads to a more diverse set of approaches to innovation, and this diversity is often socially beneficial.⁸

Drawing on numerous studies of specific innovations, these theories contend that the neoclassical models just described fail to capture crucial aspects of innovation, such as the effect of firm heterogeneity under uncertainty.⁹ As countless studies of specific technologies and innovation show, often no one knows for sure what the best uses for a new technology may be, what consumers really want, or what the best technical approach to a certain problem may be. Under these circumstances, economic actors do not know the complete set of options available to them, nor can they foresee all potential outcomes of a specific project. While each actor may have a view on the likely probabilities of the outcomes of the options he sees, these views will differ across actors and will be shaped by actors' past experiences and theories about the world.¹⁰ Theories of innovation that account for this fundamental uncertainty predict that different economic actors, when confronted with the same situation, will see different opportunities for innovation, will differ in their assessment of the value of the same opportunities, and consequently will select different projects, or different approaches to similar projects.¹¹ *Ex post*, some approaches will turn out to be successful and others will not, but it is not possible to determine the best or most efficient approaches to innovation in advance.¹²

As a result, if different economic actors can independently choose their innovation projects, the range of projects will reflect the various actors'

idiosyncratic perceptions and judgments. In contrast, if only one economic actor chooses innovation projects, the project portfolio will reflect only that actor's view of the opportunities. In view of these differences, the projects pursued by multiple independent innovators probably will be more diverse than the projects pursued under the central control of a single economic actor.¹³

Thus, under fundamental uncertainty, it matters who makes the choice. At the same time, it is not possible to know in advance which choice will be the right one. In view of these two facts, proponents of evolutionary and neo-institutional theories of innovation suggest that trying a variety of approaches and waiting to see which turns out to be the best is better than trying only one. These theories do not deny that it would be more efficient if the best allocation of resources to innovation could simply be imposed. But since the best allocation cannot be known in advance, it seems desirable to explore a wide range of alternatives so as to increase the chance that a good solution will be found.¹⁴

In these theories, the differences between decentralized and centralized innovation become more profound as uncertainties about technology or market demand increase. The higher the uncertainty, the greater the effect of differences among potential innovators will be—and, consequently, the more likely it will be that different economic actors will make different choices, and the less possible it will be to objectively identify the best projects and approaches in advance. In contrast, in relatively mature industries it may be quite obvious to market participants which innovative efforts may be feasible and profitable, and different economic actors may be less likely to select different projects than they would be in a highly uncertain environment.¹⁵ Similarly, as the complexity of the problem to be solved increases, it becomes less likely that economic actors will be able to foresee the various outcomes perfectly, and less likely that they will assess the underlying probability distributions correctly. Differences in assessment will then lead them to choose different approaches to the same problem.¹⁶ Finally, the effect of differences between actors may increase with the heterogeneity of customers.¹⁷

As this chapter will show, the history of application-level innovation on the Internet strongly supports the second set of theories. Throughout the history of the Internet, innovation in applications has often been afflicted by the type of fundamental uncertainty described above. Even when they have had access to the same information, different innovators, investors, or observers have often disagreed on the prospects of specific projects. That even the innovators themselves were unable to predict the

success of particular innovations in advance is a recurring theme in the history of many applications. In light of this fundamental uncertainty, differences between innovators were highly relevant. In line with the predictions of evolutionary and neoclassical theories of innovation, different actors often chose different applications or approaches, and these choices were often highly idiosyncratic.

All this suggests that the differences in the number and type of innovators between the two architectures will have important implications for the amount and type of application-level innovation. If, as in the end-to-end architecture, different economic actors can independently choose their innovation projects, the range of projects will reflect the different actors' idiosyncratic perception and judgment. In contrast, if, as in the core-centered architecture, only a few network providers choose innovation projects, the project portfolio will reflect only their view of the opportunities. In view of these differences, the projects pursued by multiple independent innovators probably will be more diverse than the projects pursued under the central control of a few network providers. Moreover, as we will see, even if we ignore the differences in innovation costs between the two architectures, letting everyone with access to an end host choose independently whether and how to innovate will result not only in a more diverse set of projects but also in a larger set of projects than if a few network providers and their employees are the only potential innovators. Not only will a large, diverse group of potential innovators discover a larger number of opportunities for innovation than a small group of homogeneous network providers; they will also realize a larger number of the opportunities that were discovered.

First, a large and diverse group of potential innovators will discover a larger number of opportunities for innovation than a small group of homogeneous network providers. Network providers may not see all innovative opportunities. This problem is neglected by neoclassical models of innovation which often assume that all actors know the potential opportunities for innovation.¹⁸ In reality, discovering opportunities for innovation is the first step in the innovative process.¹⁹ Often it is not immediately apparent how a new technology could be used, or that customers have unmet needs that a new or improved product could fill. This problem is particularly pronounced for relatively general technologies, such as the Internet, that can be put to a wide variety of uses. For these technologies, identifying ways in which the technology could be used is an important kind of innovation. As history has shown again and again, it is usually not possible to decide in advance which new uses will become successful.²⁰ Most Internet

applications that later became highly successful either weren't envisaged by the designers of the network or were met by widespread skepticism when they first became available. For example, that e-mail would become one of the most popular applications for the ARPANET (the Internet's predecessor) was not envisaged by its planners or by its funders (box 8.1).²¹ For years, Tim Berners-Lee and Robert Cailliau failed to convince the relevant committees at CERN that Berners-Lee's vision of the World Wide Web was something CERN should pursue. Though Berners-Lee managed to pursue the project informally with the consent of his immediate boss, Berners-Lee and Cailliau were not able to get the project accepted as a formal project, or to obtain the manpower they would have liked (box 8.2). In 1995, when Pierre Omidyar first developed eBay, friends and

Box 8.1

The Unexpected Success of E-Mail

The first working networked e-mail application was developed by Ray Tomlinson, a programmer at Bolt, Beranek and Newman, in 1971 (box 8.5). According to the ARPANET's Draft Completion Report, the success of e-mail was "unplanned, unanticipated, and mostly unsupported."^a In 1967, Lawrence Roberts, who oversaw the development of the ARPANET as assistant director and later as director of the ARPA Information Processing Techniques Office,^b had written, in an article describing the ARPANET, that the ability to "handle interpersonal message transmissions" was "not an important motivation for a network of scientific computers."^c The ARPANET was intended to support resource sharing, i.e., to allow users to remotely access and share computing resources, software programs, or databases available on different computers attached to the network. In a 1990 interview, Frank Heart, who led Bolt, Beranek and Newman's ARPANET team, recalled: "When the mail was being developed, nobody thought at the beginning it was going to be the smash hit that it was. People liked it, they thought it was nice, but nobody imagined that it was going to be the explosion of excitement and interest that it became."^d But by 1973, two years after the first e-mail had been sent over the ARPANET, 75 percent of ARPANET's traffic was email.^e

a. Heart, McKenzie, McQuillan, and Walden 1977, Draft ARPANET Completion Report, p. III-67, as cited by Abbate (1999, p. 109).

b. Abbate 1999, p. 46.

c. Roberts 1967, p. 3/1.

d. Interview by Judy O'Neill, as cited by Abbate (1999, pp. 108–109).

e. Tuomi 2002, p. 139.

Box 8.2

The World Wide Web

Tim Berners-Lee, a physicist by training, developed his vision of the World Wide Web while working as a hardware and software engineer in the data control and acquisitions group at CERN, the European laboratory for particle physics.^a He envisioned a decentralized global hypertext system that would enable researchers who came to CERN to run their experiments and then returned to their home institutions to add all sorts of documents to the system and link to related information. Berners-Lee repeatedly submitted his proposal to his boss and to other people at CERN in 1989 and 1990, but he did not get a reaction. Working with Robert Cailliau (a colleague in the Electronics and Computing for Physics division) and a student intern, he built the first versions of a browser and a server in an informal project that was justified internally as a way to test the operating system and software-development environment on a new NeXT computer. But when he and his co-workers were trying to encourage people in the Computing and Networking division to use the system to document their projects, Berners-Lee recalls, “they didn’t seem to see how it would be useful.”^b In the spring of 1991, enthusiastic researchers and librarians at the Stanford Linear Accelerator set up the first Web server outside CERN after seeing the system during a visit at CERN. Berners-Lee’s boss arranged for Berners-Lee to present the World Wide Web to the CERN Computer Centre Coordination Committee, which consisted of the managers of the main services of CERN’s IT department, in order to explain the usefulness of the Web and get more funding. In his book, Berners-Lee recalls: “What we hoped for was that someone would say, ‘Wow! This is going to be the cornerstone of high-energy physics communications! It will bind the entire community together in the next ten years. Here are four programmers to work on the project and here’s your liaison with Management Information Systems. Anything else you need, you just tell us.’ But it didn’t happen.”^c Hoping to find programmers interested in improving the program, Berners-Lee made the documentation and code for the World Wide Web available on the Internet in August 1991, in particular on a newsgroup for hypertext enthusiasts. Berners-Lee later characterized this as a “water-shed event.”^d Those enthusiasts downloaded the code, found bugs, suggested features, and put up their own servers. As of the summer of 1991, content was being created, and CERN’s server was getting 10 to 100 hits a day.

a. The account in the text is based on Berners-Lee and Fischetti 2000.

b. *Ibid.*, p. 34.

c. *Ibid.*, pp. 50–51.

d. *Ibid.*, p. 51.

potential investors could not imagine that ordinary people would be willing to buy from or sell to complete strangers in an online auction.²² At the time, even established companies had difficulties attracting customers willing to use their credit cards online, and it was not clear whether e-commerce would ever take off. Similarly, in the mid 1990s people in the Internet industry had severe doubts whether operating a search engine was a sustainable business at all. “Back in those days,” Joe Kraus (a co-founder of Excite, one of the first search engines for the Web) recalled, “it was legitimate to [think] ‘Why would I use a search engine more than a couple of times to find the sites that I like? Then I’ll bookmark those sites and never go back to a search engine again.’”²³

Even successful innovators are not necessarily able to predict in advance which uses will take off. For example, Max Levin and Peter Thiel started Confinity, the company that later became PayPal, to write cryptographic software for handheld computers. They went through several unsuccessful products until arriving—almost accidentally—at the product PayPal offers today, which allows people to make and receive online payments. At the time, their company was offering an application to transfer money from one handheld computer to another. To get people interested in buying the application, they built a “demo” for the company’s website that let people do everything on the website that they could do on a handheld computer. After a while, they became aware that people were using the demo version to actually transfer money, and that the website was much more successful than their “real” product.²⁴

When Ludicorp launched Flickr, it was an online photo-sharing application that enabled users to share photos and chat about them in real time. Users, however, wanted to share their pictures with others who were not online at the same time. The Flickr team had assumed that users would want to limit access to their pictures to family members and friends. Instead, most users left their photos public (box 8.3).

Uncertainty does not end once new uses are identified. Often it is not clear which technical approach will be the best for a particular use, which features users may want, or which business model might be the most appropriate. Under these circumstances, actors’ past experiences, the context in which they are operating, their ties to other actors, or characteristics such as alertness to innovative opportunities influence what opportunities people see and what approaches they find valuable.²⁵

For example, innovators often discover a potential opportunity for innovation when they realize that some difficulty they are experiencing might be problematic for others, too. For example, Sabeer Bhatia and Jack Smith

Box 8.3

Flickr

Flickr became a social network almost by accident.^a It was launched in February 2004 as a place for people to chat about photos, but user feedback encouraged its evolution into a platform that lets people share their pictures with the world. From a business perspective, the idea that a photo-sharing website could be profitable standing alone stood in stark contrast to the business models of existing photo-sharing applications that were loss leaders for photo-printing websites.

In 2002, Caterina Fake and her husband, Stewart Butterfield, founded Ludicorp to build a Web-based multi-player online game called Game Neverending. Inspired by Neopets (a game in which children can interact with other children while tending to virtual pets), they wanted to build a game in which people could build a community and interact in whimsical ways. In late 2003, they found themselves at the point where “we’re six months behind on back-end development[, we’re] trying to raise venture capital[, and we’re] trying to figure out what furniture we should sell to make payroll.”^b

While waiting for the “back-end” developers to catch up, the “front-end” developers had developed a feature that enabled players to drag and drop objects so as to “give” them—that is, show them—to other players. That feature inspired the original version of Flickr. The idea was to expand that feature so that photos could be dragged and dropped into a chat and onto chat participants’ desktops in the same way as game objects to create real-time interaction around photo sharing. Since “we only had money for one last shot, and we knew we could deploy Flickr faster than the game,” the team took a vote and decided to work on Flickr.^c When Ludicorp launched Flickr (in February 2004), it enabled users to chat and share photos in real time; technically, the flash-based application was based on a stripped-down version of the Game Neverending user interface. Owing to the real-time nature of the application, people who wanted to share photos with each other had to be online at the same time, which limited Flickr’s usefulness. Users preferred a website that would let them save photos that others could see at a later stage, so the Flickr team (which was closely following the discussions in user forums) added a website where people could upload and store pictures that others could view at a unique URL.^d The Flickr team had assumed that users would use the site to store and organize photo collections and share them with their families and friends. When the team built a “social network” into Flickr, the idea was that it would allow people to limit access to their photos to family and friends. But they were surprised to find that about 80 percent of Flickr photos were left public. High-quality photographers began to see Flickr as a place to show their work. When Fake and Butterfield became aware that people

Box 8.3

(continued)

were using Flickr to share their photos with the public, their background in communal gaming and social software became their big advantage. It allowed them to capitalize on this trend as soon as they saw it by using open APIs, making it easy to integrate pictures into blogs, enabling the tagging program to allow people to pull together groups around certain types of photos, and unobtrusively finding ways to feature some of the best photos.

- a. The account in this box is based on the following sources: Garrett 2005; Fitzgerald 2006; Gabbay 2006; Hawk 2005; Herron 2006; Naughton 2008; Fake 2007; Livingston 2007.
- b. Fitzgerald 2006.
- c. Ibid.
- d. The website was far more popular than the real-time application, which was phased out.

got the idea for Hotmail, one of the first Web-based e-mail systems, when they were working on another project. While working at a start-up firm named FirePower, they were trying to come up with a “killer app” that would make their fortunes. Their original idea was to create a Web-accessible relational database tool, which they named JavaSoft. They worked on the coding for JavaSoft while they were employed full-time at FirePower.²⁶ They ran into trouble developing the database, though, because they had difficulty communicating when they weren’t in the same locations. They didn’t want their bosses to discover this project, so they were using their personal e-mail accounts, but FirePower’s corporate firewall prevented them from “dialing out” to reach those accounts. The solution came to Smith one day in December 1995: Web-based email would let them access their e-mail from any computer, not just from home or at work. Since HTTP can easily traverse firewalls, firewalls would not be a problem, either. Bhatia thought the idea was perfect, and they began working on it immediately.²⁷ Others discover an opportunity for innovation when they watch others experience a need, and start thinking about potential solutions. For example, Shawn Fanning started thinking about Napster, the first digital file-sharing application, after his roommate repeatedly complained about the difficulty of finding MP3 files for download on the Internet.²⁸

Often, innovators’ ideas are inspired by the innovators’ backgrounds. For example, the decision by Niklas Zennström and Janus Friis to develop

Skype, an Internet-telephony application based on a peer-to-peer architecture, was influenced by their desire to leverage their intimate knowledge of peer-to-peer applications. The pair had already created another highly successful peer-to-peer application: Kazaa, a peer-to-peer file-sharing application that enabled users to share files and was used mostly to share MP3 music files.²⁹ Friis recalls: “After Niklas Zennström and I did Kazaa, we looked at other areas where we could use our experience and where P2P technology could have a major disruptive impact. The telephony market is characterized both by what we think is rip-off pricing and a reliance on heavily centralized infrastructure. We just couldn’t resist the opportunity to help shake this up a bit.”³⁰

Jim Clark’s and Marc Andreessen’s decision to develop the first commercial browser for the World Wide Web was deeply influenced by Andreessen’s background, which gave him intimate knowledge of the inner workings (and the limitations) of the most successful browser to date, a recognition of the growing importance of the World Wide Web, and, potentially, a desire to get back at his former employer. Together with a colleague, Andreessen had developed the first user-friendly and widely successful browser, Mosaic, while working as a student at the University of Illinois’ National Center for Supercomputing Applications (NCSA). After Mosaic had been released, the relationship between the NCSA and Andreessen had deteriorated, and Andreessen had moved to Palo Alto, where he met Clark (box 8.4).

Others discover opportunities for innovation when they perceive that a technology they are or were working with may be useful in another context. For example, the idea for the first working e-mail program for the ARPANET was born when Ray Tomlinson, a programmer at Bolt, Beranek and Newman, realized that the application he was working on (a program for sending e-mail messages between users of a single mainframe computer) could be improved using another application he had been working on earlier—a program for transferring files on the ARPANET (box 8.5). Similarly, the Google search engine was an unexpected application of an algorithm that Larry Page and Sergey Brin had developed for a different goal in their PhD work (box 8.6).

All this suggests that the discovery of opportunities for innovation is often highly idiosyncratic. As we just saw, some of today’s most important applications (e.g., e-mail, search, the World Wide Web) have idiosyncratic roots. Under the end-to-end architecture, anyone with access to an end host can become an innovator. Though network providers—the only potential innovators under a core-centered architecture—can try to hire employees with diverse backgrounds and experiences, the wide range of backgrounds,

Box 8.4

Netscape

In April 1994, Jim Clark and Marc Andreessen founded Mosaic Communications Corporation (which later became Netscape) to develop the first commercial browser for the World Wide Web.^a

Andreessen, 22 years old, had just graduated from the University of Illinois Urbana-Champaign. With a colleague, Eric Bina, he had created the first version of the Mosaic browser while working as a student at the University of Illinois' National Center for Supercomputing Applications (NCSA). Mosaic was not the first browser (the first one had been developed by Tim Berners-Lee, and other researchers had developed browsers too), but it became the first widely popular browser. Unlike most other browsers at the time, Mosaic was easy to install and use, and it had a clean interface. Whereas existing browsers displayed graphics in a separate window, Mosaic could display graphics as part of a webpage, making websites look more like magazine pages. Its point-and-click interface made it easy to surf the Web. Mosaic was first released in January 1993. Distributed freely, it quickly became popular. The NCSA, noting the success of the Unix version developed by Bina and Andreessen, decided to also develop versions for the PC and the Macintosh, making Mosaic the first cross-platform browser. Suddenly every PC or Mac user with an Internet connection could surf the Web, even if he did not understand HTML or TCP/IP. In the months following Mosaic's release, the Web quickly took off. In January 1993 there were 50 known Web servers; by October there were more than 500; by June 1994 there were 1,500.^b

After Mosaic was released, relations between the NCSA and Andreessen quickly unraveled. The NCSA, wanting to make Mosaic more corporate, began to "marginalize" the engineers from the project. Eventually they asked Andreessen to stop working on Mosaic. Rather than stay at the NCSA in some other capacity, Andreessen moved to Palo Alto, took a job at Enterprise Integration Technologies, and eventually met Jim Clark.

Clark was a Silicon Valley entrepreneur who had founded Silicon Graphics. His new interest was in how to make money off the Internet—a relatively new idea in 1994. He was looking for a technical whiz with big ideas that he could help turn into reality. Someone mentioned Andreessen's name to him. He contacted Andreessen, and they agreed to start collaborating.

Initially, the two were not sure what they wanted to develop. Clark was interested in interactive TV and wanted to create something like an online Nintendo. Andreessen wasn't convinced. After a while, Andreessen's lingering bitterness toward the NCSA resurfaced. "We could always create a Mosaic killer—just build a better product and build a business around it," Clark

Box 8.4

(continued)

later remembered Andreessen saying.^c Clark agreed to the idea. Both were convinced that cross-platform browsers would be important for the Internet, and that they might even replace operating systems.^d They quickly hired most of the Mosaic team from the University of Illinois and formed Mosaic Communications. (They later changed the name to Netscape to avoid battles with the NCSA.)

a. The account in this box is based on Wolfe 1994, Berners-Lee and Fischetti 2000, Kaplan 1999, and Hamm 1998. Who had the idea to develop a browser that would be user-friendly and easy to use is contested. Kaplan (1999) and Hamm (1998) attribute the idea to Andreessen; Deutschman (1997) attributes it to Dave Thompson, a full time-employee of the NCSA. All accounts agree that Andreessen and Bina developed the Mosaic browser.

b. Wolfe 1994.

c. Hamm 1998.

d. Afuah and Tucci 2001, p. 203.

Box 8.5

The First E-Mail Program

Ray Tomlinson, a programmer at Bolt, Beranek and Newman, got the idea for the first working networked e-mail application in late 1971 while he was working on the TENEX operating system, an operating system for Digital Equipment Corporation's PDP-10 mainframe computer.^a (A mainframe is a large computer that is shared by a number of users who access the system from different terminals.) At the time, most mainframes had mail programs that enabled users of the same mainframe to exchange messages. Tomlinson was improving TENEX's mail program when he realized that he could add functionality that would let users connected to different mainframes on the ARPANET send e-mail to one another by modifying code from an experimental file-transfer protocol for the ARPANET that he had worked on earlier. Instead of transferring files from one ARPANET computer to the other, the new mail program would transfer e-mail messages from one ARPANET computer to the other and add them to a user's mailbox.

a. The account in this box is based on the following sources: Tomlinson, undated b; Crocker, undated; Abbate 1999, p. 106; Partridge 2008b.

Box 8.6

Google

In 1995, Larry Page and Sergey Brin were studying the link structure of the World Wide Web in the course of their PhD work in computer science at Stanford University.^a Because it was easy to see the links leaving a website but difficult to see the links pointing to the same site, they thought it might be interesting to find and analyze the links linking to a site. Their goal was to develop a system (to be named BackRub) that, if given the URL for a website, would list all the websites that linked to this site, —the “backlinks”—ranked by importance. To rank the backlinks, Page and Brin developed PageRank, an algorithm that would assign a rank to a website on the basis of the number of websites linking to this site, taking into account the relative importance of these sites. The idea was that if there were a large number of backlinks, BackRub would show the most important backlinks first. While Page and Brin were studying the results of their system, it occurred to them that the rank of a page might be useful for purposes other than ranking backlinks—in particular, for searching. To test this intuition, they wrote a rudimentary search tool based on PageRank. When it yielded results far superior to those yielded by existing search engines, they decided to focus their energy on building a real, full-text search engine that would use the PageRank algorithm to help determine how to order search results. They named it Google.

a. The account in this box is based on Battelle 2005 and on Page et al. 1999.

experiences, and contexts of potential innovators under an end-to-end architecture cannot be replicated within a few network providers' organizations. As we saw, experiencing or watching a need that may be met by using the Internet in a new or different way is an important way of identifying new uses for the Internet. A few network providers likely cannot re-create the use contexts of all the various areas of business and society within their organizations. Thus, by increasing the range of actors who can discover opportunities for innovation and act on them, the decentralized environment for innovation provided by an end-to-end architecture greatly increases the number of opportunities that will be discovered.

Second, a larger, diverse group of potential innovators will realize a greater number of the opportunities that were discovered. Even if a network provider sees an opportunity, it may not find that opportunity attractive. If there is fundamental uncertainty of the type described above, where different innovators cannot agree on the prospects of the various options, potential innovators' perceptions of the likely costs and benefits may

differ; owing to these differences, the innovation may appear profitable to some of the innovators, but not to others (box 8.7).

Beyond differences in perception, different cost structures may make a project profitable for some innovators but not for others. Consider an individual programming at home in her free time, a start-up firm, and a larger firm (perhaps a network provider). To make their funds last longer, start-ups often cut costs in a number of ways—for example, the founders may work from their homes, or may rent only cheap office space, keep administrative overhead low, or forgo a part or all of their salary.³¹ All this may enable them to produce their application at lower costs than an established larger firm. For example, for the first six months of its existence Amazon was run from a converted garage in the house Jeff Bezos lived in. Meetings with third parties were held in a nearby café.³² Even after eBay had moved into rented offices, desks for new employees were picked up

Box 8.7

How Differences in Perception Affect the Number of Projects

To think more formally about how differences in perception affect the number of innovative ideas that will be realized, assume that there are a number of potential projects that have the same probability of success.^a Suppose the various innovators differ in their beliefs about the likely success of projects, and that the probability that a particular innovator thinks a project will be successful (and therefore decides to pursue this project) is δ for all innovators, with $0 < \delta < 1$. Also assume that the probability that a particular approach is successful is ϵ , with $0 < \epsilon < 1$. In this case, the probability that a particular project is pursued by at least one innovator is δ under centralized innovation, and it is $1 - (1 - \delta)^n$ under decentralized innovation with n innovators.

Because $\delta < 1 - (1 - \delta)^n < 1 - (1 - \delta)^{(n+1)}$ for all $n \geq 2$, the probability that a particular project is pursued by at least one innovator rises with the number of innovators. Intuitively, the larger the number of potential innovators, the more likely it is that at least one innovator will eventually choose a particular project. In this example, the expected ratio of successful approaches under decentralized innovation vs. centralized innovation is $[1 - (1 - \delta)^n]/\delta$, which is potentially very large. Thus, centralized innovation is likely to result in substantially less successful innovation than decentralized innovation.^b

a. Similar models are used by Cohen and Klepper (1992, pp. 7–8) and by MacKie-Mason, Shenker, and Varian (1995, pp. 20–21).

b. It will also result in substantially more unsuccessful innovation. See below.

by an eBay employee using her uncle's truck and then assembled by the employees.³³ It is easy to imagine projects whose expected benefits are larger than the costs incurred by the programmer or the start-up firm, but lower than the costs incurred by a larger firm.

Even if the benefits are larger than the costs for all different potential innovators that see the opportunity, the size of the potential benefits may make the project unattractive for the network provider and other large firms.³⁴ Successful companies need to continue to grow in order to maintain their share prices and to provide the opportunity for employees to expand the scope of their responsibilities. As growth rates are expressed as a percentage of the current market valuation, the additional revenue required to maintain a particular growth rate increases with the size of the company. The larger a firm, the higher the additional revenue required to maintain a particular growth rate. For example, while a project with potential revenue of \$10 million will let a \$50 million company grow at a rate of 20 percent, it will let Comcast, whose market capitalization at the time of this writing (June 2008) is \$63.66 billion,³⁵ grow at a rate of 0.00015 percent; for AT&T, whose market capitalization at this writing is \$214.71 billion, the relative contribution to growth would be even smaller.³⁶ Because the overall number of projects a firm can pursue is usually limited (box 8.8), it will prefer projects that contribute a significant portion of the additional revenue needed to maintain its growth rate to projects that will lead to revenues that are too small to help the company achieve its growth rate. Thus, the project with potential revenue of \$10 million probably will not be attractive to a larger company (such as Comcast or AT&T) but may be quite attractive to a start-up or a smaller firm. Similarly, the prospect of getting bought for \$5 million, \$10 million, or \$20 million may be attractive for an entrepreneur who financed a project herself or through angel investors, but may not motivate a venture-capital firm to invest in the project.³⁷ Users who innovate to use the resulting application themselves may not care about the size of the market at all; they may care only about the potential benefit from using the program. Moreover, as we will see below, network providers and other established firms may have a bias against specific kinds of projects that may lead them to not select such a project even if they see the opportunity. Finally, strategic interests may make network providers and other established firms less interested in certain projects than other actors. For example, phone companies may be deterred from aggressively pursuing innovation in Internet telephony by the fear of cannibalizing their existing revenues from traditional telephony, and cable companies may be deterred from innovating in Internet video by the fear of cannibalizing their revenues from their traditional cable offerings.

Box 8.8

Limits of Firm Size

The number of projects that a firm can pursue internally is usually limited. Though it is difficult to pin down where exactly these limits are (they may also be changing with organizational form, or with technical advances that facilitate information sharing and coordination), most researchers agree that there are limits to how large a firm can become.^a For example, it may be more difficult to monitor performance and maintain incentives in larger organizations. At some stage, increasing the scope of the firm will require adding hierarchies, which is costly and may make it more difficult for relevant information to travel to the top. Larger organizations may also incur higher influence costs, since the larger amount of resources available may motivate the larger number of stakeholders within the firm to compete more vigorously for their share of existing resources.^b

In addition, the architecture may limit the number of projects that a firm can pursue. For example, developing a new application in the core-centered architecture will often require changes to the network. The complexity of making changes to the network and keeping track of the interactions between the changes required for different applications will limit the concurrent number of new application projects that a network provider can pursue.

a. See, e.g., Williamson 1975, chapters 7, 8; Williamson 1985, chapters 6, 11; Milgrom and Roberts 1992, pp. 571–575; Besanko, Dranove, and Shanley 2000, pp. 87–91.

b. See, e.g., Milgrom and Roberts 1992, pp. 571–575.

As we saw in chapter 5, the low costs of application innovation under an end-to-end architecture let innovators with all sorts of motivations innovate: the user who wants to get the application she needs, the volunteers who cooperate (often remotely) to develop an application, the potential entrepreneur who tests his idea as a side project while making money in his day job, the small company without outside funding whose owners want to be profitable, but value control over their company's destiny more than the opportunity of getting insanely rich, firms funded by venture capitalists, larger existing companies. Though the diversity of funding sources available to potential innovators may differ from country to country or even from region to region (for example, the availability of venture capital varies considerably across countries or regions), the broad range of cost structures, perceptions, and motivations able to support innovation under an end-to-end architecture makes it more likely that

someone will find it attractive to realize an innovation even if the network provider does not find it attractive.

In sum, a large, diverse group of potential innovators, with diverse motivations and with access to diverse types of funding, not only will discover more opportunities for innovation than a small group of homogeneous network providers; it also is likely to realize a larger number of the opportunities that are discovered.

Having a larger and more diverse set of innovation projects, in turn, guarantees a more complete search of the problem space, reducing the chance that beneficial uses (or approaches to specific problems) will not be detected. If the various innovators pursue different technical approaches to the same problems or explore different business models, having a larger number of innovators increases the likelihood that at least one of them will be successful and increases the expected quality of the winner.³⁸ The benefits of diversity increase with uncertainty. Intuitively, if uncertainty is high, it is not likely that a single approach will yield a really good result. The more independent approaches are tried, the higher the probability that one of them will turn out to be really good. Certain innovators' idiosyncratic experiences or capabilities may further improve the expected quality of the outcome.

For example, Skype was not the first Internet-telephony application; it was not even the first one based on a peer-to-peer architecture. But the experience that the Skype team had gained developing Kazaa and a clear focus on voice quality and ease of use resulted in an application that surpassed the quality of existing Internet-telephony applications and was offered for free (box 8.9). Similarly, Flickr was not the first photo-sharing application; however, for reasons that were tightly linked to the circumstances surrounding its inception and the founders' backgrounds, it evolved into an offering with distinctive features—a platform for photo sharing with elements of social networking—that were central to its success (box 8.3).

Having several innovators approach a particular problem may also increase the overall quality of innovations through spillovers.³⁹ By watching others experiment, innovators can gain valuable insights about what works and what doesn't work without having to try the same approaches themselves, and can use these insights to improve their own approaches.⁴⁰ For example, providers of search engines initially struggled to find a good business model. The lack of a good business model was one of the main things that deterred potential investors from investing in search engines. The business model that Google pioneered, which dominates the industry today, evolved in a series of steps that involved substantial trial and error.

Box 8.9

Skype

When Niklas Zennström and Janus Friis began developing Skype, most of the existing Internet-telephony applications were based on a client-server architecture and used the Session Initiation Protocol, which didn't handle broadband users behind firewalls and network-address translators well. One company, PGPfone, was peer-to-peer, but couldn't work around firewalls.^a Most of the services were difficult to use, and the voice quality of the services was relatively low.^b Zennström and Friis decided to base Skype on a proprietary protocol, to use a peer-to-peer architecture, and to focus on call quality and ease of use. They hired the team of Estonian programmers that had developed Kazaa for them to adapt the technology underlying Kazaa for Internet telephony. Thanks to their experience with Kazaa, the programmers were able to overcome the problem of network-address translator and firewall traversal, something that Zennström perceives as critical to Skype's success.^c The low ongoing costs associated with a peer-to-peer application (in 2004, it cost Skype only one-tenth of a cent to add a new user, as Skype needed only servers from which the application could be downloaded and an authentication server) and the viral nature of the software (since Skype uses a proprietary protocol, Skype users can talk only to other Skype users, which gives users an interest to convince their friends to use Skype, too) enabled Zennström and Friis to make the software available for free and to charge only for premium features. In contrast, Vonage and other companies that used the Session Initiation Protocol and a client-server architecture had to use a subscription model, because of the costs of operating the server infrastructure and the higher costs of marketing (in 2004, it cost Vonage \$400 to add a new customer).^d Because it enabled people to make high-quality phone calls for free with a program that was easy to install and easy to use, Skype was an instant success.

a. McCullagh 2003.

b. Charny 2003.

c. Mohny, undated; Maney 2006.

d. The costs of adding a customer to Vonage or Skype are taken from Roth 2004.

Over time, Google and other actors within the industry tried a variety of different approaches. At each step along the way, Google not only reacted to its own experience but also built on the experiences gained by others in the industry (box 8.10). In general, innovators can learn from other innovators' successes or failures, and such learning reduces the social costs of failure.

Box 8.10

The Evolution of Google's Business Model for Search

In 1999, when Larry Page and Sergey Brin secured their first venture-capital funding for Google, receiving \$25 million from Sequoia Capital and Kleiner Perkins Caulfield and Byers, they had not yet found a business model.^a At the time, existing search engines were either using banner advertising or diversifying into portals. Brin and Page argued that banner ads would cause the Google page to load slower, and that irrelevant ads would distract from the search results. Instead, Google first tried to make money by syndicating its search technology to large websites for a fee, in addition to offering it to consumers directly. But that approach was not very successful, so at the end of 1999 Page and Brin decided to try a variant of the dominant advertising model. As the providers of other search engines did, they would sell advertising space on their site, basing the price on the number of users seeing the ad (this method of pricing is called *cost per thousand impressions*). Unlike the others, however, they would show only text-based ads that were relevant to the topic of the search.^b To prevent any confusion, paid ads were clearly separated from the normal search results. The system worked, but it did not provide enough revenue. In particular, it did not scale well, since the advertising deals were negotiated in person. In 2000, to remove the bottleneck and increase revenue, Google introduced AdWords, an automated system that let advertisers buy text ads online using a credit card. With AdWords, Google's revenue grew considerably, but GoTo, a competitor in the search market, was growing much faster.

GoTo, which later changed its name to Overture, had had been founded by Bill Gross as a commercial search engine. Whereas other search engines used algorithms or humans to rank search results, the order in which GoTo's search results were shown was determined by how much the owners of the corresponding websites were willing to pay. GoTo's business model contained two major innovations. One was that advertisers would pay only if a user actually clicked on their ad (a model now known as "pay per click"); the other was that advertising prices would be determined through an online auction in which advertisers could bid to be shown when searchers searched for particular keywords.

Although observers and industry participants had been highly skeptical when it was launched,^c GoTo became very successful. Because users who search for something are really interested in it and, if their query is related to commercial activity, willing to act, advertisers found that users searching for relevant keywords were often clicking on their ads and then buying something. As a result, advertisers on GoTo were willing to pay much more than they paid for undifferentiated traffic from other websites. GoTo also syndicated the top paid results (as determined by the amount that advertisers were willing

Box 8.10

(continued)

to pay) to other search engines or portals, sharing the advertising revenue with them. Even though the market for display advertising crumbled after the NASDAQ crash in the spring of 2000, advertisers continued to advertise on GoTo, and GoTo became profitable while other search engines and portals were struggling from the slowdown in display advertising. The fact that GoTo was able to offer its syndication partners substantial revenue per search when the traditional model of display advertising had broken down fueled its syndication business and motivated others to imitate its approach.

Google executives, who had been following Overture's success, determined that Google's lack of auction-based pay-per-click pricing for its paid ads was an important reason for the difference in the growth of the two businesses.^d In February 2002, Google introduced AdWords Select, which was based on Overture's model but which differed from it in important ways. As in Overture's system, the advertising price was determined by an auction, and advertisers paid only for actual clicks. But whereas Overture based the order in which ads were shown on the advertisers' bid, Google's system also considered the likelihood of users' clicking on the ad. Thus, an advertiser who bid \$1 per click but had a large number of users clicking on its ad would be ranked higher than an advertiser who bid \$2 per click but had few users clicking on its ad. Relative to Overture's system, these differences not only made ads more relevant for search users but also led to higher revenue for Google. (In April 2002, Overture filed a lawsuit claiming that Google had infringed Overture's patent covering its ad-placement method. The suit was settled out of court in 2004 after Yahoo acquired Overture. As part of the settlement, which also settled another claim by Yahoo, Google gave Yahoo 2.7 million shares of its stock in exchange for a fully paid, perpetual license to Overture's patents.^e) Finally, whereas in Overture's system content providers' or advertisers' payments determined the placement in the search engine's main search results, Google only used the pricing mechanism to determine the order in which its paid links were shown on the site, and continued to clearly separate between the organic search results produced by Google's search algorithm and the paid results produced by AdWords Select. The new system not only made Google's own search website highly profitable, it also enabled Google to change its syndication model. Instead of syndicating its algorithmic search results to third parties for a fee, Google now syndicated both algorithmic and paid search results to other websites, offering them a share of the revenue. Today, this general approach is still at the core of Google's highly successful business model. Yahoo and Microsoft, Google's two biggest competitors in the search-engine market, adopted similar approaches.^f

Box 8.10

(continued)

- a. The account in this box is based on the following sources: Business Wire 1999; Google 2000; Bowman 2002; Festa 2002; Kane 2002a,b; Olsen 2002a,b,d; Sullivan 2002a,b; Wolverton 2002; Battelle 2005, pp. 88–146; Vise and Malseed 2005, pp. 62–69, 82–129.
- b. Google's sales chief had used a similar approach in his previous job at Netscape, combining cost-per-million pricing with showing advertising results based on keywords (Battelle 2005, p. 124).
- c. See, e.g., Peline 1998; Sullivan 1998.
- d. In 2001, Gross approached Brin and Page to discuss a potential merger, but Brin and Page did not approve the mixing of paid advertising with normal search results (Battelle 2005, pp. 115–116).
- e. Olsen 2002c, 2004.
- f. Yahoo did not use a relevancy factor such as click-through rates until it introduced its new search marketing system Panama in early 2007. The lack of such a factor is usually viewed as one important reason for the gap in search-engine monetization between Google and Yahoo. Another reason is the different number of users. See chapter 6 above, subsection "Complementary Product Source of Outside Revenue."

Finally, if consumers are heterogeneous, having a larger and more diverse group of potential innovators is likely to result in applications that better meet the needs of different consumer groups.⁴¹ Because innovators often discover opportunities for innovation that are related to their backgrounds or to the context they are operating in, letting anyone with programming knowledge and access to an end host become an innovator makes it more likely that a specific user need is recognized. More innovators will not only discover more opportunities for innovation, they probably will also realize more of the opportunities that are discovered. For reasons discussed above, there will be many potential applications or services that users may want but which a network provider may not pursue. For example, catering to a niche market with relatively low profit margins may be interesting for a small software company with little overhead whose owners are happy to be just profitable, but not for larger companies with different cost structures that look for larger profit opportunities.⁴² By enabling a wide range of actors with diverse backgrounds and diverse motivations to innovate, the end-to-end architecture makes it more likely that someone will find it attractive to cater to a particular group of consumers, even if this would not be attractive for the network provider. If no

one else is interested in meeting their needs, users can always realize the application themselves.

To better understand when and why a larger and more diverse pool of innovators increases the amount of innovation, let us look at three specific types of innovators: new entrants, companies backed by venture capital, and users. These innovators are able to innovate under an end-to-end architecture, but may lose their ability to participate in innovation as an architecture moves away from the broad version of the end-to-end arguments. As we will see, these innovators often have an incentive to innovate in situations where the network provider does not have such an incentive. When these types of innovators lose the ability or incentive to innovate as a result of deviation from the broad version, this innovation will be lost. In addition, innovation by these types of innovators offers specific benefits that a network provider cannot replicate. Thus, the following subsections focus on these types of innovators, because they are able to innovate under an architecture based on the broad version, but they are not necessarily able to innovate under an architecture deviating from it. The resulting differences in the size and diversity of the innovator pool, in turn, may result in differences in the amount and the type of innovation under these alternative architectures. The focus on these types of innovators is not meant to imply, however, that they are the only potential innovators, or the only important types of innovators. For example, large and established firms may have specific advantages that a start-up firm cannot match. Large firms often benefit from economies of scale and scope, from economies of specialization, or from established business relationships with other firms. As we saw above, they are part of the pool of potential innovators under an end-to-end architecture. Their perspectives and specific advantages enable them to focus on projects that other types of innovators may neglect or may not be as good at, which contributes to the diverse set of innovations that will occur under such an architecture. Large, established firms will, however, often be less affected by deviations from the broad version of the end-to-end arguments, so their contribution to application innovation is not necessarily lost as a result of such deviations. Therefore, they are less important to the overall argument of this chapter. This is not meant to discount their importance to application innovation; it only reflects the specific focus of the analysis in this chapter and in this book.

The Value of Innovation by New Entrants

Different network architectures let new firms influence the rate and the direction of innovation in different ways. Under network architectures that centralize control over innovation, the rate and the direction of innovative

activity are determined by a few network providers—established firms already operating in the industry. Either new entrants are blocked from innovating or their freedom to innovate is restricted by the need to get permission from the incumbent. In contrast, under the decentralized environment for innovation provided by an end-to-end architecture, incumbents and new entrants independently determine the rate and direction of innovation.

Throughout the history of the Internet, important new applications have often been developed by new entrants, not by established firms. For example, the online auction site eBay was started by Pierre Omidyar, not by a newspaper or a major Internet service provider. In the early years, the members of the eBay team constantly feared that an Internet service provider (e.g., AOL, CompuServe, or MSN) or one of the big portals would start competing with them. They felt that online auctions were a logical next step from the online classified sections that the big ISPs had developed. But none of the major ISPs entered the field.⁴³ Web-based e-mail was pioneered by startups (including Hotmail and Four11, which developed Rocketmail), not by the big portals. Instant messaging was invented by the Israeli startup ICQ, not by a portal or an ISP. Online music distribution was revolutionized by Shawn Fanning's Napster, not by the record companies.

In general, it took established firms who had been in the industry before the Internet became popular longer to realize the commercial potential of the Internet. For example, the most successful online bookshop, Amazon.com, was started by Jeff Bezos, a former senior vice president at a quantitative hedge fund who had a degree in computer science and electrical engineering, in the garage of his newly rented house in Seattle in June 1994. Amazon.com started selling books on the Internet in July 1995.⁴⁴ Barnes & Noble, the first major brick-and-mortar bookseller to sell books on the Internet, opened its first online store in the spring of 1997, after Amazon.com had become so successful that its success could not be ignored.⁴⁵ (In May 1996, a front-page article in the *Wall Street Journal* reported that Amazon was on its way to selling more than \$5 million in books in 1996. Sales were growing by 20–30 percent every month, and Amazon's sales revenue that year surpassed \$15 million.⁴⁶)

Another incumbent, Microsoft, initially failed to understand the growing importance of the Internet and the World Wide Web.⁴⁷ The commercial potential of providing software with which to access the World Wide Web was recognized by Netscape, a start-up firm, not by Microsoft. After the 1993 release of the Mosaic browser, the Web grew rapidly. Jim Clark and

Marc Andreessen founded Netscape in April 1994 to produce the first commercial Web browser. After October 1994, when Netscape released the first version of its browser, Netscape quickly garnered a dominant market share. In May 1995, Sun launched Java, a programming language that enabled programs to run within a browser, making the programs independent of the computer's operating system. Websites were using server programs such as the NCSA's HTTPD (released in mid 1993), Netscape's commercial server (launched in December 1993), and the open-source Apache server (released in April 1995), creating the prospect of an Internet free of Microsoft products. Meanwhile, Microsoft was focusing on finishing Windows95, WindowsNT, and Microsoft Network (an online service designed to compete with AOL). Although Bill Gates assigned the Internet "the highest level" of importance in an internal memo in 1995, changing Microsoft's strategy in response to the Internet and the competitive threat posed by Netscape and Java was delayed until after the launch of Windows95 and Microsoft Network. When Windows95 was shipped in August 1995, it included TCP/IP support, so that computers running the Windows95 operating system could connect to the Internet, but it did not include a browser. (Customers could buy Internet Explorer 1.0 as part of the Microsoft Plus add-on, though.) Microsoft Network went live using proprietary standards, not Web standards. In response to the threat posed by Netscape, Microsoft included Internet Explorer 2.0 with the operating system in November 1995 and released its first Web server in February 1996.

Other incumbents were offered the chance to commercialize important innovations, but declined. In line with CERN's "buy, don't build" policy, Tim Berners-Lee initially tried to get hypertext firms to work with him to build the World Wide Web. Existing commercial hypertext editors already let users open, display, and modify hypertext documents. Berners-Lee felt that these editors could easily be turned into the browsers he envisioned for the World Wide Web by adding functionality that could send hypertext documents across the Internet. However, none of the firms he approached wanted to pursue the idea, so Berners-Lee decided to write a browser and a server himself.⁴⁸

Sergey Brin and Larry Page, the founders of Google, originally did not want to form their own company, but tried to sell or license their search technology to existing portals. Between 1996 and 1998 they offered their search-engine technology to the major portal and search-engine companies, but all declined. At the time, all the existing search-engine companies (including AltaVista and Excite) had become portals that made money from advertising. Search was just one of many services they wanted to

provide, and not a particularly important one; after all, the portals' goal was to keep users on the portal, where they would create more advertising revenue or commissions for online sales. A search engine that by design led people away from the portal did not fit with this strategy. Thus, the existing search-engine companies had no interest in acquiring a better search-engine technology; search had to be only "good enough." For the same reasons, Yahoo, MSN, AOL, and other portals that did not have their own search engines were uninterested. For them, search was something to be outsourced to a third party, not a cornerstone of their competitive advantage.⁴⁹

Research in management strategy suggests that this pattern was no accident. There seem to be systematic differences between firms that have been successful in an industry for some time and firms that are new to an industry, and these factors influence the relative abilities of the incumbent and the entrants to perceive the value of specific new technologies and to exploit them.⁵⁰ Moving from an end-to-end architecture that lets new entrants innovate to a network architecture that lets only established firms innovate would result in the loss of innovation by newcomers. (I do not mean to imply that established firms generally fail to take advantage of new technical opportunities. In fact, established firms may have an advantage with respect to innovations that require a large amount of resources. They may benefit from economies of scale or scope, or they may be able to build on existing business relationships. For example, when Google develops new applications, it can take advantage of its existing data centers or of its existing network infrastructure in a way that a new entrant who lacks these resources cannot. The argument is, rather, that there are certain opportunities for innovation that established firms may find unattractive, and that these innovations would not be realized under a non-end-to-end architecture under which only network providers or other established firms can innovate.)

The research reveals that a firm's competitive position is crucially influenced by its resources and capabilities, and that resources and capabilities may be difficult to both acquire and change.⁵¹ Firms that have been successful in an industry for some time have optimized their resources and capabilities for the technological and competitive circumstances in the industry; in fact, this is the basis of their success. For example, the engineers of such a firm may be experts in the leading technology and its particular application in the company's products, and the firm's organizational structures and processes may be optimized to enable efficient product development and improvement based on the leading technology and on

the needs of the customers in the firm's current market. But the assets that originally created the firm's competitive advantage may become disadvantageous in the face of particular types of technical change. If such a change occurs, incumbent firms may find it difficult to recognize the implications of the new technology or to change their resources and capabilities to take advantage of the new situation. New entrants whose perceptions and capabilities have not been shaped by the existing situation have an advantage in recognizing the value of the new technology and in mustering the resources and capabilities to exploit it.⁵² Thus, this research predicts that incumbent firms will be good at pursuing innovations that fit their resources, their capabilities, and their economic position, but that they will be less able than new entrants to see the value of a new technology and adopt it if it is incompatible with incumbent firms' capabilities at the technological level, at the organizational level, or with respect to its economic position in the industry.

At the technological level, a new technology may require radically new technological competences that the incumbent is unable to acquire. Because that is more likely to occur with technological change involving radically new technology than with incremental changes, scholars of management strategy predict that new entrants may have an advantage with respect to radical technological change.⁵³

At the organizational level, a firm's organizational structure may prevent it from seeing the significance of certain types of technological change and adapting to it. For example, the organizational structure of a firm often mirrors the architecture of its product, with different organizational units responsible for different components. Knowledge about the relationships and interdependencies between components shapes the organizational structures and processes that guide interactions among different organizational units. Even though a new technology might improve the overall quality of the product by improving the product architecture, the significance of this fact may get lost in the organizational structures and processes shaped by the existing architecture. Even if an established firm recognizes the value of the new technology, changing its organizational structures and processes may be difficult; owing to organizational inertia, it is usually more difficult to change established structures and processes than to build new ones from scratch. Thus, scholars of management strategy predict that technologies that require or enable changes at the architectural level of a product architecture will pose more of a challenge to leading firms than changes at the component level that leave the overall architecture of the product unchanged. New entrants whose organizational structures are not

yet shaped by the existing architecture may have an advantage in perceiving the value of architecture-level innovations and in realizing them.⁵⁴

Finally, and most importantly in the current context, a firm's economic position in the market may influence its perception of the value of different opportunities for technological innovation.⁵⁵ Firms often operate in differentiated product markets where the definition of quality differs considerably among groups of consumers; stated differently, consumer groups base their purchasing decision on very different criteria. For example, large corporations with dedicated project managers value project-management software with a rich set of features that let them produce Gantt charts or identify the critical path of the project; they do not mind the one-time effort to learn to navigate the complex software. In contrast, a small business that cannot afford to assign one person to project management full-time needs simple software that lets project members cooperate and track progress. Since no one can focus on this task full-time, the software has to be intuitive and easy to use.⁵⁶ As the example of project-management software shows, different quality attributes are often mutually exclusive, forcing firms to focus on a particular group of customers with common valuation of quality attributes. In this case, such characteristics of the competitive environment as typical cost structures and available profit margins are determined by the competition between the firms that compete for the same group of customers. In such a setting, a leading firm that has been successful for some time is successful because its resources, technological capabilities, organizational structures, and processes are well adapted to meet the distinctive needs of the group of customers it has decided to serve; it has found a way to outperform its competitors given the cost structures and available profit margins that are typical of its sub-industry. In other words, a successful firm's resources, technological capabilities, organizational structures, and processes are optimized to the specific characteristics of the competitive environment in which it operates.⁵⁷

Although integral to the firm's success, this optimization constrains an established firm's perception of the value of different opportunities for technological innovation.⁵⁸ Contrary to an entrant, it is constrained by five factors: its existing group of customers, its existing cost structure and profit margins, its size, the incentive structure of its managers, and the firm's processes for managing innovation. As we will see, these constraints will bias an established firm in favor of products that let it better meet the needs of mainstream customers in its current market or that enable it to move into established markets with higher profit margins; they will bias it against products that do not meet the needs of customers in the firm's

current market, that must be commercialized in new or emerging markets, or that are afflicted with significant business uncertainty.⁵⁹ New entrants that do not share these constraints will view such projects more favorably. Thus, understanding how these constraints affect the way in which successful firms assess potential opportunities for innovation will help us understand better which types of projects they may reject. Although potentially attractive for a new entrant, these projects would not be realized under an architecture in which only network providers or other established firms can innovate.

First, a successful established firm is constrained by its existing group of customers. A firm is successful in an industry if it can give its customers what they want.⁶⁰ Even though this orientation toward existing customers is the basis of the firm's success (the firm is successful precisely because it puts a lot of effort into identifying the needs of its customers and evaluates potential projects based on how well they meet these needs), it biases the firm in favor of projects that let it better meet the needs of these customers.

Second, a firm's existing cost structure and profit margins influence the type of projects it finds attractive, biasing it to favor projects that increase its profit margins,⁶¹ a natural way to increase profits. Projects with lower profit margins than existing products would require the firm to cut costs in order to maintain the current level of profitability. Successful companies probably already have cost structures that let them outperform their competitors in the current market. As a result, it may not be possible to cut costs and remain competitive in the current market. In any event, cost cutting is usually difficult and painful, making projects with lower profit margins unattractive. These considerations also influence the markets an established firm finds attractive: moving into markets with higher profit margins enables the firm to earn higher profit margins based on its current cost structure and potentially gives it a cost advantage over firms already operating in that market. In contrast, moving into markets with lower profit margins would require the firm to cut costs to maintain its current level of profitability. As a result, established firms have a tendency toward upmarket mobility and downmarket immobility.

Third, a firm's size influences the type of projects it considers valuable.⁶² As we saw above, the size of a firm determines the additional amount of revenue a firm must earn in order to maintain its desired growth rate. As a result, a firm will prefer projects that contribute a significant portion of the additional revenue needed to maintain its growth rate to projects that will lead to revenues that are too small to help the company achieve its

growth rate. On the basis of these considerations, an established firm will prefer projects targeting large markets to projects aimed at small markets.

Fourth, in established companies, proposals for innovation are evaluated at various levels of the organizational hierarchy. When evaluating proposals, managers also consider the potential effect of these decisions on their own careers. This biases managers against projects afflicted with considerable market uncertainty. A failure in the product market due to lack of consumer demand happens after investment in product development, manufacturing, marketing, and distribution has been made, and is therefore considerably more costly and publicly observable than failures in the early stages of an innovative project due to technological problems. As a result, such a failure is far more detrimental to a manager's career than a failure due to technological problems. At the same time, the reward from successful innovation (e.g., a bonus or a raise) is usually limited. As a result, a manager will tend to support projects directed toward well-known customer needs in established markets for which market uncertainty is low.⁶³

Finally, the processes used to plan and manage innovation for well-known customer needs in established markets are not appropriate for managing innovation in emerging markets.⁶⁴ In established markets, consumers' needs and market sizes are usually well known; as a result, market uncertainty is low. In this situation, sophisticated marketing, investment, and management processes that require a wealth of data can be used to plan and control the innovation and product development process. Emerging markets, with their inherent uncertainty about consumer demand and market evolution, are not amenable to methods that require detailed data on consumers' needs and reliable forecasts on market development. Thus, the same processes that are crucial to managing innovation in established markets leave companies ill-equipped for dealing with emerging markets.

Given these constraints, an established firm will value technologies or innovations that let it better meet the needs of mainstream customers in its current market or that enable it to move upmarket into established markets with higher profit margins.⁶⁵ These technologies have the potential to increase profit margins on the basis of the firm's current cost structure. In established markets in which customers' needs are well known, market uncertainty is low. In addition, established markets are large enough to contribute the amount of revenue that a large company needs to sustain its growth, making them attractive targets for innovation.

In contrast, an established firm will be biased against innovations that do not meet the needs of customers in the firm's current market, that must

be commercialized in new or emerging markets, or that are characterized by significant business uncertainty. Disruptive technologies are an important example of such innovations. Disruptive technologies typically do not meet the needs of customers in the firm's current market today, but they have the potential to evolve to a point where they better meet the needs of customers in this market in the future.⁶⁶ Products based on disruptive technologies are usually cheaper and simpler than existing products, resulting in lower profit margins. As they initially underperform established products in mainstream markets, they cannot be used in these markets, but must be commercialized in new or insignificant markets. Such markets do not solve the growth needs of a large successful company.⁶⁷ Finally, the uncertainty associated with creating new markets requires very different approaches to managing product development than facilitating innovation in established markets.⁶⁸ Consider Internet telephony, or video over the Internet. When the first Internet-telephony applications were developed, voice quality was lower than on the wireline network. Calls were less reliable and often required a running computer. Similarly, video-over-the-Internet transmissions often do not have the same high resolution that customers have come to expect from conventional television; it may take a moment until the video starts playing, and the video may stop unexpectedly to wait for more data. Sitting in front of a computer screen is not as relaxing as sitting on a couch in front of a large-screen television. As a result, these applications originally did not or still do not meet the needs of mainstream customers in the markets for telephony and video broadcast. As network and application technologies evolve, they may offer competitive performance. Owing to their initial quality, Internet telephony and video over Internet had to be commercialized in new markets. Given the uncertainty associated with emerging markets and the profit and growth needs of established companies, it is not surprising that Internet telephony and video-over-Internet applications were originally developed by new entrants, not by incumbent phone companies or cable providers.⁶⁹

Contrary to an established firm, an entrant was not active in the industry before the advent of a new technology. As a result, an entrant's approach to innovation is not constrained by an existing group of customers, an existing cost structure, existing profit margins, and large size. Consequently, the entrant may be better able to identify the value of disruptive technologies and to exploit them.⁷⁰ More generally, a new entrant may find it easier to find new applications and markets for a new technology. A new entrant is not already dedicated to a particular group of customers.

As a result, it can evaluate the potential of the technology without reference to the needs of a particular group of customers. Instead, it can look for a group of customers that actually values the characteristics of the new technology.⁷¹ Contrary to an established firm, a new entrant is not constrained by an existing cost structure, but can adapt the cost structure of the organization to the profit margins that the technology enables.⁷² Finally, a new entrant is small and therefore needs less revenue than a large established firm to grow at a particular rate. Thus, for a new entrant, an emerging or insignificant market may constitute a viable growth engine.⁷³ As a result, an entrant may see attractive opportunities for innovation where an established firm sees none.

The Value of Venture-Capital-Backed Innovation

Under an end-to-end architecture, anyone, including a company financed by venture capitalists, can be an innovator. But at some stage along the path from an end-to-end architecture to a more centralized environment for innovation, firms backed by venture capital lose their ability to innovate. Under the core-centered architecture, network providers are the only remaining innovators.

Venture capital can be defined as “equity or equity-linked investments in young, privately held companies, where the investor is a financial intermediary who is typically active as a director, an adviser, or even a manager of the firm.”⁷⁴ Venture capitalists raise funds from investors and use these funds to make equity or equity-linked investments in young, privately held companies. Thus, venture capitalists act as financial intermediaries between outside investors and young, privately held companies that need financing. Their function goes beyond providing funds. Once the decision to invest in a company has been made, venture capitalists closely monitor and control the company’s progress, providing strategic advice, reputation, and contacts.⁷⁵

Throughout the history of the Internet, companies backed by venture capitalists have commercialized important new applications. Netscape, Yahoo, eBay, Amazon.com, Excite, ICQ, Hotmail, Rocketmail, Google, Napster, PayPal, Skype, Vonage, Friendster, MySpace, Facebook, and del.icio.us were all funded by venture capitalists at some stage in their history. Many of these firms pioneered completely new types of applications. This is not an accident. Venture capitalists specialize in high-risk, high-return projects that established companies often reject. In addition, innovation by venture-capital-backed firms offers specific benefits that other companies may not be able to replicate. When deviating from the broad version

of the end-to-end arguments increases the costs of innovation so much that firms backed by venture capitalists lose the ability to innovate, these innovations and the associated benefits will be lost.

Compared to innovative employees and middle managers in established firms, entrepreneurs and venture capitalists face very different incentives, which lead them to pursue very different types of projects.

Though innovative employees and middle managers in established firms usually benefit from a successful innovation, the rewards are relatively limited. They may get a raise or a bonus, and their prospects for promotion may increase. At the same time, if a project fails for lack of market demand, the results are usually very expensive and public and thus probably will have a negative effect on the reputation and career of the middle manager who backed it. Thus, whereas the personal gain from backing a successful project is limited, the personal risk associated with backing a project that is afflicted with significant market uncertainty is high. Owing to this incentive structure, middle managers are believed to be averse to risk when choosing inventions for commercial development.⁷⁶ In particular, they will prefer a project with low market uncertainty and a moderate return over a project with high market uncertainty and high potential gains.

In contrast, entrepreneurs and venture capitalists retain a significant proportion of the gains associated with successful projects. Their compensation is directly linked to the firm's success. A considerable fraction of the compensation of managers and critical employees in venture-capital-backed firms consists of equity or stock options.⁷⁷ Similarly, venture capitalists usually receive a proportional share (traditionally around 20 percent) of the profits when the investment in a firm is liquidated.⁷⁸ Successful projects can create enormous profits. Thus, entrepreneurs and venture capitalists can make a lot of money if a firm is successful. But the personal loss from unsuccessful projects is limited. For an entrepreneur, failure of a firm is not necessarily regarded as a personal failure. If the firm fails as a result of circumstances the entrepreneur cannot influence, he may well be able to get funding for a new promising venture.⁷⁹ Similarly, owing to the high risk associated with investment in young, innovative companies, unsuccessful investments are regarded as inevitable in the venture-capital business. As a result, a venture capitalist's reputation is based on her overall track record, not on the failure of isolated investments.⁸⁰ Owing to this incentive structure, entrepreneurs and venture capitalists are usually believed to be "skewness lovers"—that is, to be motivated by the relatively low probability of an extremely high reward.⁸¹

Thus, venture capitalists have an incentive to invest in companies (and indirectly in projects) that they believe to have the potential to yield attractive returns. Contrary to middle managers in established companies, they will not be deterred by the uncertainty surrounding such projects or be biased by existing customers, cost structures, and business lines, because the management structures and processes of venture capitalists are designed to deal with high-risk, high-return projects (box 8.11).⁸² Historically, venture capitalists concentrated most of their investments in early-stage companies in high-technology industries, where the perceived opportunities for innovation-based growth are huge and where the problems caused by uncertainty and information asymmetry are particularly profound.⁸³ For example, in the early 1980s venture capitalists made a large fraction of their investments in computer hardware and semiconductor electronics. Venture capitalists also invested heavily in the emerging computer software, biotechnology, Internet, and telecommunications industries.⁸⁴ For example, in 1999, 39 percent of all venture disbursements were directed at Internet investments, and 17 percent of all venture disbursements at telecommunications investments.⁸⁵ Between 1994 and 2007, venture capitalists made more than \$218 billion in Internet-related investments in 19,664 US companies.⁸⁶ In 2007, 38 percent of all venture investments in US companies were directed at Internet-related investments.⁸⁷

Thus, venture capitalists enable independent innovators to pursue high-risk, high-return projects that established firms, for whatever reason, reject. As a result, the ability of venture-capital-backed firms to participate in innovation widens the range of innovative projects and increases the probability that projects directed toward new or emerging markets will be adopted.

Apart from influencing the type of innovation, firms backed by venture capital may also create innovations of higher quality. Like all start-up founders, entrepreneurs in venture-capital-backed companies face strong incentives to innovate. Entrepreneurs, critical employees of the firm, and venture capitalists have high-powered compensation schemes that closely link their compensation to the performance of the firm and potentially enable them to reap enormous profits. Owing to the high incentive intensity, all parties are likely to expend greater effort than employees in an established firm, who do not similarly profit from successful projects.

Relative to start-up innovators using other forms of financing,⁸⁸ however, entrepreneurs whose companies are backed by venture capitalists may be better able to realize and commercialize their innovation thanks to the financial and other support provided by their venture capitalists. Venture

Box 8.11

Venture-Capital Mechanisms for Managing Uncertainty

Venture capitalists have developed management structures and processes that enable them to deal with the specific problems associated with high-risk, high-return ventures. To manage the risk associated with such investments, they carefully screen investment proposals to identify promising opportunities for investment, using their knowledge of the industry and a wide network of contacts.^a Usually one venture-capital firm invests together with a similarly experienced venture-capital firm (syndication), so as to get a second opinion and so as to reduce the chance of funding unsuccessful projects.^b

Venture capitalists also use mechanisms to minimize potential losses. Apart from providing a second opinion, syndication lets a venture-capital firm invest in more companies, reducing the exposure to any single investment.^c Once the decision to invest in a company has been made, venture-capital firms disperse funds in stages. They evaluate the progress of the funded firm at each stage and decide whether to continue funding. They can thus monitor and control the progress of the funded firm and terminate funding if the firm's prospects turn negative, thereby capping potential losses.^d Finally, venture capitalists actively support the operation of the firm to increase the probability of success.^e

Finally, venture capitalists have developed mechanisms to address potential agency problems between the entrepreneur and the investors.^f Through day-to-day involvement with his firm, the entrepreneur has more information than venture capitalists or investors about the progress of the company and its future viability. To reduce the possibility that the entrepreneur exploits this informational asymmetry to the detriment of investors, the venture capitalists closely monitor and control the progress of the firm^g and structure the compensation of the entrepreneur and important employees to align the incentives of managers and investors.^h

a. See, e.g., Kaplan and Strömberg 2001, pp. 427–428; Gompers and Lerner 2001a, pp. 44–46.

b. See, e.g., Gompers and Lerner 2004, pp. 166–167 and chapter 11.

c. Gompers and Lerner 2004, p. 167.

d. For a theoretical and empirical analysis of this practice, see Gompers and Lerner 2004, chapter 8.

e. See, e.g., Hellmann and Puri 2000; Kaplan and Strömberg 2001; Hellmann and Puri 2002.

f. For an overview of the theoretical literature on agency problems between the entrepreneur and the investors, see Gompers and Lerner 2004, pp. 157–163.

g. See, e.g., Kaplan and Strömberg 2001, pp. 428–429.

h. See, e.g., Gompers and Lerner 2001a, pp. 50–51, 53–58.

capitalists can give reliable and predictable financial support. This lets the firm focus on the efficient realization of their innovation without being impeded by a lack of funding or being distracted by the need to organize more financing.⁸⁹ Apart from providing the funds necessary to pursue the project, venture capitalists support the firms in their portfolio in a variety of ways; for example, they provide contacts and reputation that gives the firm access to other sources of financing or to important strategic business partners.⁹⁰ During the 1990s, Kleiner Perkins Caulfield and Byers, one of the leading venture-capital firms in Silicon Valley, became well known for fostering connections between the companies in Kleiner's portfolio and supporting strategic alliances among them.⁹¹ Intuit, Sun Microsystems, Macromedia, and Netscape had all been funded by Kleiner Perkins at some point in their history. Shortly after the Netscape browser was launched, Intuit released a new version of its personal finance software Quicken that was tightly integrated with a special version of the Netscape browser. Sun Microsystems integrated Netscape's server software into its server offerings, while Netscape licensed Sun's Java programming language and integrated Java support into its browser. Netscape integrated Macromedia's director player, which enabled users to view multimedia files created using Macromedia's software, into the browser as well. Under a deal with Sun, Macromedia would license Java and integrate Java support into Macromedia's multimedia products.

Venture capitalists are used to the problems of young firms operating in uncertain environments and can help them adopt viable product-development, marketing, and finance strategies. They support the professionalization of the firm's internal organization in a number of ways: they help select the management team, help professionalize the recruitment and human resource processes, and foster the adoption of stock option plans.⁹² For example, Kleiner Perkins Caulfield and Byers and Sequoia Capital had only agreed to fund Google on the condition that Brin and Page would hire an experienced chief executive officer. When Eric Schmidt, at the time CEO of Novell, was not interested in Brin's invitation to interview for the CEO position, John Doerr, one of the partners at Kleiner, convinced him to accept. Schmidt became chairman of Google's board in March 2001 and CEO in July 2001, forming a triumvirate with Brin and Page.⁹³ When their initial collaboration turned out to be a bit rocky, Doerr suggested the three get some informal mentoring from Jim Campbell, former CEO and current chairman of the board at Intuit and a board member at Apple. The three agreed, and observers credit Campbell with helping them find ways to work together and make decisions.⁹⁴

Finally, venture capitalists help a firm decide when to go public, and prepare the initial public offering.⁹⁵ The importance of this non-financial support is highlighted by the stories of Netscape and eBay. Although the two firms did not depend on venture capitalists to fund their firms (Jim Clark, co-founder of Netscape, had the financial means to finance the firm himself; eBay was profitable from the beginning), both firms chose venture-capital financing for the non-financial support associated with it.⁹⁶

The evidence shows that the support of venture capitalists pays off. Empirical studies demonstrate that companies backed by venture capitalists bring their products to market faster than similar entrepreneurial companies with other types of financing.⁹⁷ Firms that receive venture-capital financing grow more quickly and tend to be considerably younger at their IPO dates than non-venture companies.⁹⁸ Finally, venture-capital-backed firms continue to outperform non-venture-capital-backed companies even after the IPO.⁹⁹

In a large number of market categories related to hardware or software for personal computers or the Internet, venture-capital-backed firms have become market leaders or acquired dominant positions. Intel, Microsoft, Cisco Systems, America Online, Yahoo, Adobe,¹⁰⁰ eBay,¹⁰¹ Amazon, Google, and PayPal all received funding from venture capitalists in their early stages.¹⁰² In 2000, in the computer software industry, firms that had been funded by venture capitalists in their early stages represented more than 75 percent of the software industry's market value; in the computer hardware industry, such firms represented 78 percent of the industry's value.¹⁰³

Finally, there is some evidence that venture capital fosters innovation: for example, in the biotechnology industry, venture capitalists provided financing to 450 out of 1,500 firms between 1978 and 1995. These 450 firms, however, "accounted for over 85 percent of the patents awarded and drugs approved for marketing."¹⁰⁴ In another study, researchers used data covering twenty US manufacturing industries over three decades to assess the effect of venture capital on innovation.¹⁰⁵ According to their statistical calculations, "one dollar of venture capital appears to stimulate patenting three times more than a dollar of traditional corporate research and development does. This finding suggests that venture capital, even though it averaged less than 3% of corporate R&D from 1983 to 1992, drove a much greater share—about 8%—of US industry innovations over that decade."¹⁰⁶

Network providers cannot replicate the advantages associated with venture-capital funded firms, for example by having their own corporate

venture-capital funds. In recent years, a number of companies have created their own corporate venture-capital funds. While this may improve the situation compared to a world where only network providers can innovate, this will not necessarily solve all problems.

First, it is difficult to replicate the incentives and structural safeguards of independent venture capitalists within a firm.¹⁰⁷ For example, companies are often hesitant to pay their employees in a corporate venture-capital division the same share of the profits that an independent venture-capital firm would provide, since it may distort the company's pay structures. As a result, venture-fund managers within a firm may be less motivated than independent venture capitalists; it may also be harder for the firm to compete for the best fund managers. Empirical studies show that corporate venture funds often perform worse than independent funds and do not last as long—with one exception: corporate venture funds showed a comparable performance if there is a strategic overlap between the corporate investor and the funded firm. Thus, a few corporate venture firms associated with network providers would not match the breadth of expertise and investment activity that a large number of independent venture capitalist firms would provide.

Second, diversity among venture capitalists is as important as diversity among innovators. In the face of uncertainty, venture capitalists may have different views on a venture's potential. Having a large number of venture capitalists may be beneficial because it increases the chances that at least one venture capitalist may find the innovation worthwhile. Many companies that were eventually funded by venture capitalists (including eBay, Google, and Skype) were first rejected by other venture capitalists.¹⁰⁸

The Value of User-Driven Innovation

Scholars of innovation have long classified potential innovators on the basis of their functional relationship to the innovation—that is, on the basis of how they benefit from the innovation.¹⁰⁹ A “manufacturer” benefits from an innovation by selling it to others; a “user” benefits by using it herself.¹¹⁰ For example, 37signals (an independent software vendor that sells the project collaboration and management software it develops to customers) is a manufacturer; a bank that develops market-analysis software in-house to use the software in its day-to-day operations is a user.

Under architectures that centralize control over application development and deployment in the hands of the network provider, the network provider is a manufacturer with respect to most applications; it innovates to sell the resulting applications or services to consumers. In contrast,

under an end-to-end architecture, manufacturers (including the network provider) as well as users have the ability to innovate.

Research suggests that there are situations in which users may have an incentive to innovate but a manufacturer may not. In addition, innovation by users may provide specific social benefits that innovation by a network provider cannot replicate. These innovations and benefits will be realized under an end-to-end architecture, but they will not be realized under a non-end-to-end architecture in which users cannot innovate.

Because users and manufacturers have different functional relationships to innovation, their relative costs and benefits and the criteria they use to determine whether to innovate differ.

Consider the example of Mark Fletcher, who developed Bloglines, a Web-based news-aggregation service that was later acquired by Ask Jeeves. Fletcher, who was running an anti-spam company at the time, had bookmarked more than 100 sites that he visited every day to see whether they had been updated. Clicking on 100 bookmarks only to see that the corresponding website or blog had not been updated took up a considerable amount of his time, so he began to look for an alternative way of doing this. At the time, there were some programs that would aggregate the information from the different sites in one place; however, these programs, which ran on the user's computer and were not synchronized between different computers, were not useful to Fletcher, who was using several computers during the day. A software engineer with a background in building server applications, he figured that a Web-based aggregator would solve his problem. He decided to program one for himself in his free time.¹¹¹

David Heinemeier Hansson was developing a Web-based project-management application called Basecamp as a contractor for the small US firm 37signals while working toward a bachelor's degree in computer science and business at Copenhagen Business School. While programming the application, he realized that parts of the code constituted generic solutions to recurring problems and could be extracted for future use. The result was Ruby on Rails, a "toolbox" that can be used to build applications in the Ruby programming language more quickly and more effectively. Programmers at 37signals (where Heinemeier Hansson became a partner after graduation) have used Ruby on Rails to create other products, including a Web-based to-do list, a personal information manager, and an intranet for small businesses.¹¹²

In the late 1990s, legal scholars at Harvard Law School's Berkman Center for Internet and Society, unhappy with the choices offered by the commercial teaching software provided by the law school, developed software

applications that helped them teach better. For example, the online discussion forum provided by the commercial software offered by the law school did not encourage students to respond to others' contributions. To remedy this, Jonathan Zittrain, a Lecturer at Harvard Law School and Executive Director of the Berkman Center at the time, conceived an online tool that linked random pairs of class participants for a point-counterpoint exchange and worked with two students to develop a Web-based application with that functionality.¹¹³

Like Fletcher, Heinemeier Hansson, or Zittrain, users often discover opportunities for innovation during their normal activities.¹¹⁴ They experience a problem, and start thinking about a solution. They may have an incentive to realize their idea if the tools necessary to solve the problem are available to them, the benefits from solving and having solved the problem are greater than the associated costs, and existing solutions either aren't available or offer a lower net benefit.

A user may benefit from innovating in several ways. The user gets a solution that is tailored exactly to her needs.¹¹⁵ This may let her perform her task better or more efficiently, and thus may let her be more productive. The user may also derive benefits from the process of innovating itself, such as learning or a sense of fulfillment.¹¹⁶

The costs of innovating consist of the costs of investing the resources necessary to develop and realize the innovation and the associated opportunity costs of not being able to use the resources for other purposes. As we saw earlier, in the original architecture of the Internet the resource costs of developing a software application consist mostly of the costs of programming and testing the application.¹¹⁷ As a result, the only costs that the user who develops an application to solve her own problems incurs is the opportunity cost of the time spent designing, programming, and testing the application.¹¹⁸ While working on the innovation, she is unable to do something else. If she is working on it in her free time, she forgoes other recreational activities; if she is working on it as part of her employment relationship, she is giving up the opportunity to work on other employment-related tasks. As we saw in chapter 4, the widespread availability of open-source programming tools and open-source software components has considerably reduced the costs of developing new applications (even if they are Web-based), making it more likely that the benefits of developing an application will be larger than the costs.

Some of the most important applications in the history of the Internet have been developed by users. For example, throughout the 1970s and the 1980s, most of the programs used to read and send e-mail were

developed by researchers on the ARPANET who were unhappy with existing programs.¹¹⁹

Tim Berners-Lee developed the World Wide Web to better support the distributed community of CERN researchers and staff (box 8.2).¹²⁰

The dominant Web server on the Internet at the time of this writing, the Apache Web Server, was originally developed by a small group of webmasters who needed server software for the websites they were administering (box 8.12).¹²¹ And like Ruby on Rails, many of the major programming languages used in Web development today, including Perl, PHP, and Ruby, were originally developed by programmers who were unhappy with existing programming languages. Later, they often released their software in open-source form.¹²²

The likelihood that the expected benefits will be larger than the costs will be even higher for users who innovate on open-source software.¹²³ Like Brian Behlendorf and the webmasters who needed a server to run their sites, users contribute to open-source projects whose results they would like to use. For example, a lot of the programmers who contribute to the development of Ruby on Rails use the framework in their daily work.¹²⁴ More generally, when exploring the motivations of contributors to open-source projects, one study found that “facilitating my work due to better software” was ranked as the most important benefit.¹²⁵ In another study, 59 percent of contributors ranked use of the output they create as one of their three most important reasons to innovate.¹²⁶ Because the effort needed to produce the program is shared among the group of contributors, contributors to open-source software have to contribute only a portion of the effort needed to produce the full program, but get the benefits of being able to use a full program.

Other users innovate by modifying the source code according to their needs. The source code for open-source software can be downloaded from the Internet free of charge. Open-source licenses explicitly allow people to use and modify the program. By adapting open-source software to their specific needs, users can get a tailored solution without having to create a complete program from scratch.¹²⁷ Thus, a webmaster who needs a specific feature that the Apache server does not offer can download the Apache software and add the desired feature, even if she did not contribute to the Apache software in the first place. This lets her get a complete but tailored solution while incurring only the costs of the modification. Thus, relative to users who innovate on their own to produce an application they need from scratch, users of open-source software may incur lower costs for the same benefit.¹²⁸

Box 8.12

The Apache Web Server

A Web server is a software application that can be used to provide content on the World Wide Web. The architecture of the Web consists of two types of application programs that are implemented on end hosts and which communicate with each other to provide the overall functionality. A browser such as Mozilla's Firefox or Microsoft's Internet Explorer retrieves particular documents or content from the Web. Web server software, located on computers that host the documents and content available on the Web, services requests from browsers for particular documents or content. In performing these tasks, today's Web servers also perform additional functions such as security and authentication of users or gateways to databases. Thus, whereas the Web browsers are used by those who view content, Web servers are used by those who provide content.

In 1995, the most popular Web server was NCSA httpd, developed in 1993 by Robert McCool, an undergraduate systems administrator working at the National Center for Supercomputing Applications at the University of Illinois. McCool wrote the code at the request of Marc Andreessen, who was developing the Mosaic browser at the NCSA. Andreessen thought that the availability of good Web servers would facilitate the adoption of the Web and, in turn, of his browser. The NCSA made early versions of the server source code available for free and released them with a non-restrictive license.

Soon webmasters all over the world were using the software to run their sites. When they found bugs, they either reported them back to McCool at the NCSA or fixed them themselves, sharing their patches with McCool and other website administrators through a mailing list administered by CERN. When McCool left the NCSA to work for Netscape, in mid 1994, the NCSA stopped maintaining the server code. As a result, the number of patches that had not been integrated into the latest NCSA version of the source code grew rapidly. Webmasters willing to use the software had to download the latest NCSA version, try to find all available patches, and manually integrate them into the source code. Not only did this result in a lot of duplication of efforts, the same bug being fixed again and again; it also created conflicts between different patches. To solve this problem, one of the webmasters proposed to a small group of webmasters that they produce their own updated version of the NCSA server code. Brian Behlendorf, who was chief engineer at HotWired at the time, set up a mailing list to collect the patches and to coordinate work on the source code and a shared workspace. The webmasters collected the available patches, integrated them into the code, tested them on their servers, and released them to the public in April 1995. During the summer of 1995, Robert Thau designed a new server architecture and a new code base that made the code more modular and solved problems that had plagued the

Box 8.12

(continued)

existing version of the server. Version 1.0 of the Apache software, which was based on this code, was released to the public in December 1995. Although changes to the Apache versions of the source code are controlled by a core group of developers, anyone can download the source code and make modifications. Less than a year after the group's founding, the Apache Web server was the most popular server on the Internet. Not only was the software available for free; it also offered functionality that was not available in commercial servers offered by Netscape or Microsoft, such as the ability to host a large number of websites on a single server ("virtual hosting"). If a webmaster needed functionality that was missing, she was free to modify the source code and add the functionality.^a

a. The account in this box is based on the following sources: Moody 2001; Osterlie 2002; Apache Software Foundation 2008a,b.

In sum, there may be a variety of circumstances in which a user's benefits from innovating exceed the costs. This does not mean that users wouldn't prefer to buy the innovation from a manufacturer. In fact, a user would prefer to buy the innovation if the manufacturer's solution offered a higher net benefit.¹²⁹ The manufacturer, however, may not always have an incentive to innovate.

First, a manufacturer may fail to perceive the opportunity for innovation. Information about users' needs is costly to acquire. Whereas users discover their needs as a by-product of their normal activity at no incremental cost,¹³⁰ manufacturers need to spend considerable effort to acquire use-related information. Although there are many ways in which a manufacturer can acquire such information, including surveys, focus groups, and interacting with users,¹³¹ these come at considerable cost. Information related to users' needs usually is "sticky"—that is, costly to transfer from its point of origin at the user's site to the manufacturer in usable form.¹³² Thus, the relative costs of identifying users' needs are usually significantly higher for manufacturers than for users.¹³³

Second, even if a manufacturer is aware of users' particular needs, it may not necessarily have an incentive to meet them.¹³⁴ In making the decision to innovate, users focus on how valuable it is to them to be able use an innovation themselves. In contrast, a manufacturer benefits from an innovation by selling it to others; the potential benefits often rise with the number of potential customers. As a result, given limited resources, a

manufacturer will choose to work on those innovations that deliver the most expected profit. Working on projects that fulfill the needs of a large number of users will often be more profitable than working on projects that fulfill the needs of a small number of users.¹³⁵ Thus, even if the benefits of a project exceed its costs, a manufacturer may not necessarily pursue it.

In developing a product that appeals to as many customers as possible, manufacturers often focus on the needs of the *average* customer.¹³⁶ If customer heterogeneity is high, manufacturers may divide the market into segments of customers with broadly similar needs. Different products are then targeted to the average customer in each segment.¹³⁷ Still, this method may leave a considerable number of users' needs unserved.¹³⁸

All this suggests that there will be circumstances under which users may pursue an opportunity to innovate to meet their own needs and a manufacturer may not. The more specific the needs of a particular user and the smaller the potential market consisting of users with similar needs, the more likely such a situation is to occur. The ability to innovate themselves may also be particularly important to so-called lead users, who experience certain needs well in advance of most other users.¹³⁹ Since they are so much further ahead, they need solutions when the nature, risks, and potential size of the overall market are difficult or impossible to forecast. As we saw earlier, this uncertainty often reduces the incentives of manufacturers to pursue the innovation in question.¹⁴⁰

Enabling users to innovate offers three specific social benefits that innovation by manufacturers alone does not offer. First, innovation by users meets demand that would otherwise have remained unserved. Network architectures that do not let users innovate leave users at the mercy of products offered by innovating manufacturers. In an end-to-end architecture, users whose needs are not met by existing products can help themselves and innovate. If users have the ability to innovate and at least some of them have an incentive to do so, a higher fraction of consumers' needs will be met under an end-to-end architecture than under an architecture that rules out innovation by users. The size of such benefits is difficult to estimate. Surveys indicate, however, that today's standard commercial products may, on average, leave between 46 percent and 54 percent of customer needs unserved.¹⁴¹ The size of this potential benefit increases with the heterogeneity of consumers. The greater that heterogeneity and the higher the uncertainty about consumers' needs, the more likely it is that some consumers' needs may not be served.¹⁴²

Second, depending on the application in question, the adoption of the innovation by the user may have other beneficial effects. If the

programmers at 37signals use Ruby on Rails to write new applications, their productivity increases.¹⁴³ If the legal scholars who developed teaching software use it in their teaching, the quality of their classes increases.

Finally, unlike manufacturers, users often choose to voluntarily reveal their innovations to others for free.¹⁴⁴ This may be done in a variety of ways. Some users let others use the application for free without giving away the program itself. For example, Mark Fletcher let others use Bloglines for free, but did not make the software or its source code available for download.¹⁴⁵ Others make the application software available for download, but not the source code. Others also make the source code available for free, enabling others to modify the software according to their needs.¹⁴⁶ For example, Tim Berners-Lee published the specifications for the protocols underlying the World Wide Web on the first Web server and made the source code for the first browsers and server publicly available. In 1993, CERN released the property rights to the protocols of the World Wide Web into the public domain, making it possible for anyone to use the protocols for free.¹⁴⁷ The Apache project was run as an open-source project from the beginning (box 8.12). David Heinemeier Hansson released his Ruby on Rails Web application framework in open-source form in July 2004; it is now maintained by a group of programmers including Heinemeier Hansson. Anyone can download the Ruby on Rails source code from the Ruby on Rails website for free.¹⁴⁸ Similarly, the programmers who originally developed the popular programming languages Perl, PHP, and Ruby to meet their own needs later made them available in open-source form for free. Free revealing is an inherent feature of open-source development projects; contributors to those projects invest their own private resources in developing source code and make their contributions publicly available by posting them on the projects' websites. Under the open-source software licenses, anyone can download, use, and modify open-source software for free. The size of these activities should not be underestimated. For example, in August 2008, Sourceforge.net, an infrastructure provider and repository for open-source software projects, listed more than 180,000 open-source software projects and more than 1.9 million registered users.¹⁴⁹ Although free revealing was at first surprising to economists, researchers have made progress in explaining why users may have an incentive to freely reveal their innovations (box 8.13).

If users voluntarily reveal their innovation for free, others can reap the benefits associated with use of the innovation without incurring the costs of innovating. In contrast, manufacturers often do not find it in their interest to freely reveal their innovation to others.¹⁵⁰ From the point of

Box 8.13

The Theory of Free Revealing

At first, the fact that users freely reveal their innovations puzzled economists. After all, conventional economic theory assumes that an innovator, in order to have sufficient incentives to innovate, must have the right to exclude others from using his innovation. Otherwise, he would not be able to reap the financial rewards necessary to cover the costs of innovation, which would remove his incentives to innovate.^a If people will be able to use an innovation developed by others for free, why should anyone incur the costs of developing the innovation? On this reasoning, society rewards innovators with intellectual-property rights—limited sets of rights that give them a temporary monopoly over their innovation.^b In this context, uncompensated spillovers of proprietary knowledge to third parties constitute a loss to the innovator, since they reduce his ability to profit from the innovation.^c

Recent research has made some progress in understanding the incentives that drive users to freely reveal their innovations.^d Explanations usually start with the insight that for the user innovator the main benefit from innovating is the ability to subsequently use the innovation. Thus, in order to make free revealing rational, the associated benefits do not have to cover the costs of innovating – these costs usually are covered by the benefits associated with using the innovation. Instead, the benefits associated with free revealing must exceed the costs of free revealing. If the costs of free revealing are low, the associated benefits need not be very large.^e Thus, innovation and free revealing by users are easier to explain than innovation and free revealing by other types of non-commercial innovators.

The costs of free revealing consist of the costs of making the innovation available and the costs associated with giving up or not enforcing intellectual-property rights to the innovation.

Because software is a digital good, an innovator who is connected to the Internet can make it available for download by posting it on a website, or can distribute it using peer-to-peer file-sharing applications. Thus, the costs of reproducing and distributing the innovation are very low.

The opportunity costs of not enforcing intellectual-property rights to an innovation consist of (1) forgone profits that might have been available from selling the innovation and (2) any advantage that competitors of the innovator can gain through “free riding” on the invention. For users, both of these costs will often be low. Often the transaction costs and risks associated with licensing or selling the innovation are quite high. As a result, benefiting from the commercial exploitation of intellectual property will often not be a practical alternative for users.^f Whether the innovator loses a competitive advantage if others are able to adopt the innovation for free depends on the degree of rivalry between the innovator and potential adopters and on the importance

Box 8.13

(continued)

of the innovation for any competitive advantage in the market in which the innovator and potential adopters compete. This suggests that free revealing by users may be more common if the perceived rivalry between the innovator and potential adopters is low.⁸ Indeed, an empirical study indicates that the willingness of sports enthusiasts to freely reveal sports-related innovations decreases with an increase in competition between members of the sporting community.^h

Thus, if the innovation can be made available on the Internet and the perceived rivalry between the user innovator and potential adopters is low, the costs of free revealing will be very low. As a result, a small benefit from free revealing may be sufficient to motivate that user innovator to make the innovation available.

Recent research has identified a variety of such benefits that may operate simultaneously. First, in many cases, at least some adopters of an innovation may provide valuable feedback or suggestions for improvement. This benefit has been highlighted in studies of open-source projects and online games.^l Second, in the case of open-source software, free revealing is a prerequisite to having the functionality developed by the innovator included in the standard version of the open-source software that is maintained and distributed by the open-source project. Consider a webmaster who has developed a new feature for the Apache Web server. If the code for the feature is incorporated in the standard version, it will continue to be included in future updates of the program. Otherwise, the webmaster may not automatically be able to use the feature with future versions of the program, and may even have to rewrite it.^l Third, the innovator may also expect that if he offers his innovation today and thereby benefits others, he may be able to benefit from another user's innovation in the future.^k Finally, free revealing may also serve to increase the innovator's reputation among peers. This will be particularly relevant if there is a community of users, such as the community of those contributing to a particular open-source project or the community of online gamers of a particular computer game. In the case of open-source software, the innovator may also be interested in establishing a reputation among potential employers or venture capitalists.^l

Researchers also suggest that user innovators will be less concerned about free riders than conventional economic theory assumes. First, as indicated above, while free riding on the innovative effort itself, at least some adopters may perform testing and other important functions.^m Second, conventional economic theory assumes that innovators and free riders benefit equally from using the innovation. This assumption may not be correct in all cases. For example, innovators may derive a higher benefit from using the innovation,

Box 8.13

(continued)

since it exactly matches their needs.ⁿ They may also derive benefits from the process of innovating, such as learning, enjoyment, or a sense of ownership and control.^o Studies suggest that learning is an important output of research and development projects. Knowledge acquired through a project may enable an individual to better benefit from innovations in the future.^p The resulting combination of intrinsic rewards and more tangible direct and indirect benefits will often be sufficient to cover the low costs of free revealing.

a. See, e.g., Arrow 1962.

b. See, e.g., Arrow 1962. For a short introduction to this line of reasoning, see Carlton and Perloff 2005, chapter 16. For an in-depth analysis of the economic rationale underlying intellectual-property law, see Landes and Posner 2003.

c. See, e.g., Tirole 1988, p. 400.

d. On free revealing by users in general, see Harhoff, Henkel, and von Hippel 2003 and von Hippel 2007. As has become apparent, free revealing plays an important role in open-source projects. Research aimed at understanding the motivations to contribute to open-source projects is therefore applicable in the present context. For attempts to reconcile the empirical findings on open-source with economic theory, see Benkler 2002, Lerner and Tirole 2002, and von Hippel and von Krogh 2003.

e. See, e.g., von Hippel 2007, p. 304; von Hippel and von Krogh 2003, pp. 214–215.

f. For arguments along these lines, see von Hippel 2007, pp. 304–306; Harhoff, Henkel, and von Hippel 2003, pp. 1754–1756; von Hippel and von Krogh 2003, pp. 214–215. For an analysis of the circumstances under which user innovators may switch roles and become manufacturers, see Shah and Tripsas 2004.

g. See, e.g., Harhoff, Henkel, and von Hippel 2003, p. 1757; von Hippel 2007, p. 307.

h. Franke and Shah 2003, pp. 170–172.

i. See, e.g., von Hippel and von Krogh 2003, p. 218; Jeppesen and Molin 2003, pp. 374–376.

j. See, e.g., Harhoff, Henkel, and von Hippel 2003, p. 1759.

k. Franke and Shah (2003, pp. 172–173) call this motivation “generalized exchange behavior.” See also Harhoff, Henkel, and von Hippel 2003, p. 1757.

l. See, e.g., Lerner and Tirole 2002, pp. 212–220.

m. See, e.g., von Hippel and von Krogh 2003, p. 218.

n. See, e.g., Harhoff, Henkel, and von Hippel 2003, p. 1760.

o. See, e.g., von Hippel and von Krogh 2003, pp. 216–217.

p. See, e.g., Cohen and Levinthal 1990.

view of potential adopters, however, buying an innovation is more costly than getting it for free. As a result, there may be less diffusion of the innovation if only the manufacturer innovates than if a user innovates and makes the innovation available for free.¹⁵¹ Because widespread adoption of an innovation throughout the economy is a prerequisite for the innovation to positively affect economic growth,¹⁵² increased adoption resulting from free revealing by users may provide significant social benefits. For example, releasing Ruby on Rails as open-source software enabled many developers to create Web-based applications with less time and effort. The framework has been widely adopted. Among the best-known applications are Twitter (which enables users to send short, text-based messages to others who have signed up to receive them), the Yellow Pages website, and the Web-based productivity applications developed by 37signals.¹⁵³ A number of open-source programs have been diffused widely; for example, according to the July 2008 Netcraft survey, 49.5 percent of the nearly 175 million Internet domains surveyed used Apache Web server software.¹⁵⁴ The Firefox browser, an open-source program, had a market share of 21 percent in November 2008, which made it the second most popular browser after Internet Explorer (which had 70 percent of the market).¹⁵⁵ Sendmail, an open-source software program for routing e-mail on the Internet, was estimated in 2008 to handle about 65 percent of all Internet e-mail traffic.¹⁵⁶ Linux, an open-source computer operating system for personal computers, is estimated to have between 7 million and 16 million users worldwide.¹⁵⁷ All these products compete with commercial off-the-shelf alternatives. Firefox, Sendmail, and Linux are not necessarily examples of user-driven innovation alone. And not all open-source software exemplifies user-driven innovation; companies have developed hybrid models that enable them to benefit from open-source software.¹⁵⁸

In addition, free revealing often reduces the costs of second-generation innovators who are interested in improving an innovation. Instead of licensing the right to use the innovation for a fee or spending time and effort creating a comparable solution on which to build, innovators interested in improving an existing application can directly build on it and incur only the incremental costs associated with improving it.

The Possibility of Innovators' Contracting with Network Providers

So far we have assumed that, in the core-centered architecture, network providers and their employees are the only remaining innovators. One may ask whether the few network providers that control innovation can replicate the advantages of decentralized innovation by a diverse set of

Table 8.2

Different environments for application innovation and deployment.

	End-to-end network	Non-end-to-end network with network provider contracting	Core-centered network
Control over application innovation	Application developers	Network providers	Network providers
Control over application deployment	Users	Network providers	Network providers
Innovator pool	Large and diverse	Large and diverse	Small and homogeneous

innovators, for example by contracting with independent innovators (table 8.2).¹⁵⁹ (Thinking about this question will also help us understand whether and why it is important if application developers other than the network provider can innovate independently without needing permission or cooperation from the network provider.) Contracting with independent innovators may enable a network provider to benefit from opportunities discovered by such innovators, to take advantage of their differing capabilities, motivations, or cost structures, or, in some architectures, to overcome the fact that only a limited number of projects can be pursued within the firm.¹⁶⁰ Though such a strategy would increase the amount of innovation over what network providers alone would realize, it would not create as much innovation as a decentralized environment for innovation.¹⁶¹

First, if there is uncertainty, any increase in the number of decision makers that must approve an innovation reduces the chances that the innovation is realized.¹⁶² A network provider will see no need to contract for applications that it perceives as infeasible, as not viable, or as counter to its strategic interests. These projects would be realized under an end-to-end architecture, but not under contracting.

Second, an innovator has to disclose its innovation as part of the contract negotiations. If the innovation is not protected by intellectual-property rights, there is a danger that the network provider may appropriate the innovation without paying for it, which may reduce the innovator's incentive to innovate in the first place.

Third, bargaining costs and strategic behavior may prevent a contractual solution.

Fourth, the incentives for independent innovators who can directly commercialize their innovation are higher owing to the possibility of

exceptionally large gains. If the innovators depend on contracting with a network provider in order to commercialize their innovation and cannot gain access to the market on their own, they do not have any bargaining power. In fact, the network provider will be in a monopsony or oligopsony position, which leads to very low prices. In this case, the innovators will have to bear the risk of failure or bargaining breakdown, and will receive only modest compensation if they succeed. Such an incentive structure will probably not be sufficient to motivate innovators or their investors to put up with the risk.

Finally, if innovators have to contract with the network provider before they can innovate, they may be less able to react to new developments once they have started their project. Throughout the history of the Internet, successful innovators have often changed course repeatedly—for example, in response to feedback from consumers, or in response to an unexpected scarcity of funding. For example, in January 1999 Meg Hourihan and Evan Williams founded Pyra Labs to develop Web-based project-management software for geographically distributed project members to use in managing and coordinating their work. Users, however, were much more interested in Blogger, an application that enabled them to post entries to weblogs. Pyra Labs had made Blogger available for free to attract users to their project-management product.¹⁶³ Since they did not have enough resources to focus on both products, they finally decided to focus their efforts on Blogger (box 8.14).

Similarly, Caterina Fake and Stewart Butterfield, the founders behind Flickr, had not planned to develop a website for online photo sharing with social networking elements. They had founded their company, Ludicorp, to build a Web-based online game for a very large number of players. A shortage of funds motivated them to switch their focus to creating, instead, an application that would enable people to chat about photos. In response to consumer feedback, this application then evolved, through a series of changes, into a platform for sharing photos (box 8.3).

More generally, starting with one approach and then adapting it as events unfold may be the only way to successfully navigate fundamental uncertainty.¹⁶⁴ In a network that requires innovators to coordinate their activities with the network provider, this may not be possible. The transaction-cost and coordination-cost theories discussed in chapter 5 predict that deciding what to do in response to new developments is much more difficult and more time-consuming across firm boundaries than within a single firm. This is because firms have differing perspectives on how to react, and

Box 8.14

Blogger

Meg Hourihan and Evan Williams founded Pyra Labs in January 1999 to develop Web-based project-management software that would enable geographically distributed project members to manage and coordinate work.^a In 1998, the first weblogs were started. But weblogs were the province of the technically sophisticated. The aspiring weblogger had to set up and find a host for a website and had to maintain the software using an FTP client or command-line prompts. A few weblogs, like Arts and Letters Daily, had captured some mass attention, but it was still relatively rare to keep or read a weblog.^b

Williams had written some software for his personal weblog that automated the task of posting entries to a blog. The members of the Pyra Labs team used this software to post thoughts and information to an internal weblog that they used to coordinate their work, and later wrote software that let them cross-post some of these entries to the company's external weblog. Thinking that this tool might be attractive to website developers (the audience they were targeting with their main application), they decided to add it to their project-management software as an additional application and give it away for free, hoping that it would attract users to their main product.

People began using Blogger immediately. It became much more successful than the Pyra Labs team had expected, which created a problem for the company. People were using Blogger instead of Pyra Labs' "real" product. Though the team was convinced that users would be willing to pay for the project-management application, they could not figure out how to make money from Blogger. They tried to work on both products for a while, but this strained the resources of their small team. After Blogger's user base grew steadily for more than 9 months, they decided to focus their efforts on Blogger and drop their original project.

a. Livingston 2007, pp. 111–125; McKinnon 2001; Mead 2000.

b. Hourihan 2005; Mead 2000; Galant 2005.

the lack of efficient mechanisms for inter-firm decision making and dispute resolution may make the differences difficult to resolve.

Thus, relative to architectures in which innovators can innovate independently, architectures that force innovators to contract or otherwise coordinate with a network provider before they can innovate not only constrain independent innovators' ability and incentives to start a project. They also limit innovators' ability to react to new developments during the lifetime of the project—a limitation that is particularly detrimental under uncertainty.

The Costs of Diversity

Diversity is not without costs. Each approach incurs costs, and not every project will turn out to be successful. Because successful projects cannot be identified in advance, these costs seem to be unavoidable.¹⁶⁵ For the innovators' decision to innovate to be rational in expectation, the innovators must, on average, be sufficiently compensated from successful projects to compensate for the risk of losses associated with unsuccessful projects.¹⁶⁶ Empirical evidence on investment in high-technology ventures indicates that this is the case. For example, one study showed that, of the projects under analysis, the median project yielded less than the capitalized cost of capital funds invested, but that losses on the majority of projects were more than offset by gains from the most successful projects.¹⁶⁷ From society's point of view, the social costs of failed projects may be reduced by two factors. For people experimenting in their free time, the main cost is the opportunity cost of not doing other things that one normally does in one's free time. If the innovator had otherwise engaged in unproductive uses of her time, such as watching television, the social cost of the failed experiment may not be very high.¹⁶⁸ More generally, failed projects may still produce social benefits through spillovers. Other innovators may learn a valuable lesson from one innovator's failure—a lesson that may allow them to save the expense associated with trying the same approach themselves, or to use insights offered by the failed approach to improve their own product.¹⁶⁹

In sum, if there is uncertainty, or if users' needs are heterogeneous, a larger and more diverse group of innovators that independently decide on their approaches to innovation will realize a larger and more diverse number of applications than a few network providers that control which innovative ideas can be realized. The larger and more diverse number of innovative projects, in turn, guarantees a more complete search of the problem space and increases the chance that beneficial applications or approaches will be detected. It increases the probability of success and increases the expected quality of the best results. Thus, diversity increases the amount and quality of innovation. If users' needs are heterogeneous, diversity will also result in a greater variety of products that better meet users' needs.

Effect of Differences in Control over Deployment

In the end-to-end architecture, innovators independently choose which applications to realize; in the core-centered architecture, only network providers make this selection. In the end-to-end architecture, users decide

which applications are deployed and used by installing them on their computers and using them; in the core-centered architecture (and in the controllable architectures), network providers control which applications can be deployed and used. So far, we have focused on how differences in control over application development affect innovation. But what is the effect on innovation of differences in control over deployment?

Users' and network providers' decisions to deploy applications may differ for two reasons. First, network providers' interests may differ from users' interests, which can lead them to reject applications that users would find attractive. (As was shown in chapter 6, factors such as a concentrated market structure or market power due to switching costs or incomplete information may enable network providers to reject applications without being disciplined by the threat of customers' switching to another provider.) Whereas users choose the applications that best suit their needs, network providers may use different criteria. For example, as we saw in chapter 6, under certain conditions network providers may have an incentive to exclude an application that competes with one of their own applications.¹⁷⁰ Apple's behavior toward iPhone applications illustrates this possibility. Before any application can be sold in the iPhone App Store, Apple must approve it. Since the iPhone App Store is the exclusive distribution medium for iPhone applications, this enables Apple to control which applications can be deployed.¹⁷¹ Apple has rejected several applications for the iPhone, claiming that they duplicated functionality provided by existing Apple applications. For example, MailWrangler, an application that enabled users to manage their Gmail accounts from within the iPhone browser, was rejected for duplicating the iPhone's e-mail application. Another application, Podcaster, let users listen to podcasts and download podcasts over the air directly to the iPhone. In September 2008, Apple rejected that application, arguing that users could already download podcasts on their computer and load them on their iPhones. In December 2008, Apple itself introduced Podcaster's functionality. In both cases, users, developers, and observers argued that the functionality differed from and met their needs better than Apple's existing applications.¹⁷² Thus, Apple blocked applications that competed with its own products, or with products it hoped to create, even though at least some users clearly preferred the blocked product. Similarly, when choosing to exclude an application in order to manage bandwidth, network providers may consider whether the application is easy to detect, whether their preferred vendor's products can detect the application, or whether blocking or throttling the application has a noticeable effect on traffic

(ironically, the last criterion makes popular applications particularly attractive targets for this type of bandwidth management). As Comcast's blocking of the popular file-sharing application BitTorrent shows, the fact that an application is popular with users does not necessarily prevent the network provider from interfering with it. Instead, the network provider may simply try to interfere with the application in a way that is less noticeable by the user.

Second, decisions by users and by network providers to deploy applications may differ for another reason. Even in cases where network providers would like to deploy applications that users want, they are not necessarily able to do so. As we saw above, new applications are often afflicted with considerable uncertainty. No one knows which applications or features users will find attractive. Often users themselves do not know whether they like a specific application or find it useful until they have tried it or seen others using it. Thus, letting users choose applications replaces network providers' guesses about what users may like with a decision by users, who best know their current preferences and needs. It gives applications whose usefulness or attractiveness is not immediately apparent a chance, and it allows users' preferences and practices to evolve as a result of exposure to or experimentation with a new application. In addition, giving application developers direct access to users lets them experiment and change their product offerings in response to user feedback. The stories of PayPal and Flickr have shown how important this can be.¹⁷³ Being able to choose themselves is also important for users or user groups whose needs are idiosyncratic, since their needs may not be known to or may not be important to the network provider.

Overall Effect of Architectural Differences on Application Innovation

Thus, the mechanism that produces innovation in applications under an end-to-end architecture has two components: widespread experimentation by a large and diverse group of innovators who independently select whether to realize their innovative ideas and user choice among the resulting applications. Under uncertainty or consumer heterogeneity, this mechanism will produce more and better results than innovation in network architectures that concentrate control over innovation or deployment in the hands of a few network providers.

Together with the results of chapter 7, these insights help us predict how the differences between architectures based on the broad version of the end-to-end arguments and architectures deviating from it affect

the overall amount and quality of innovation. The overall amount of innovation is determined jointly by the effects of changes in the costs and benefits of innovation (which may affect network providers and independent innovators differently), changes in the size and diversity of the innovator pool, and changes in the allocation of control over innovation and deployment.

With an architecture based on the broad version of the end-to-end arguments such as the Internet's original architecture taken as a baseline, making the network more opaque increases the costs of innovation for network providers and independent innovators alike (although ultimately network providers are better able to deal with the situation; in the extreme case, they are the only remaining potential innovators). The private benefits of application-level innovation will usually stay the same; they may increase or decrease slightly if the reduction in the number of innovators reduces competition in the market for an application.¹⁷⁴ Relative to the end-to-end architecture, this reduces the number of innovations that innovators will find worth realizing for both network providers and independent innovators. As we saw in chapter 4, the increase in costs may particularly affect innovators' ability to continue to serve niche markets or low-value markets, and may reduce their willingness to take risks. As the pool of innovators becomes smaller and less diverse, and as control over application innovation and deployment shifts to network providers, the amount and the quality of innovation are reduced further, and more user needs are left unserved.

Again taking the end-to-end architecture as a baseline, increasing the controllability of the network increases network providers' general profits and their expected benefits from their own application-level innovation, while independent innovators' expected benefits decrease. While network providers' costs stay the same, independents' costs of innovation increase. As a result, network providers' incentives to innovate increase, while independent innovators' incentives to innovate decrease. As costs and the requirements that potential innovators must meet increase, the size and the diversity of the innovator pool decrease. As this chapter has shown, the increase in innovation by a few network providers cannot offset the reduction or elimination of innovation by a large number of independent innovators.

Thus, under uncertainty or consumer heterogeneity, deviations along both dimensions reduce the overall amount and quality of innovation, and leave more user needs unserved. Relative to a fully opaque architecture, network providers' incentives to innovate will be higher under a

fully controllable architecture. Also, although their incentives to innovate decrease, not all independent innovators will lose their ability to innovate. As a result, the overall amount of innovation will be higher under a fully controllable architecture than under a fully opaque one. Since the core-centered architecture combines the characteristics of a fully controllable and a fully opaque architecture, it will produce the least amount of innovation.

9 Public and Private Interests in Network Architectures

Different network architectures shape the economic environment for application innovation in different ways. In particular, a network architecture's compliance with or deviation from the broad version of the end-to-end arguments affects who can innovate and what incentives they have to do so. Ultimately (other things such as the set of actors exposed to the architecture and the set of constraints [i.e., laws or norms] under which they operate all being equal), these differences will result in different amounts, different qualities, and different kinds of innovation in applications.

But just because there are differences, legislators and regulators do not have to care about them. Therefore, this chapter evaluates how the differences among network architectures that comply with or deviate from the broad version of the end-to-end arguments relate to the public interest. If the values fostered by architectures based on the broad version are in the public interest, can we rely on the market to produce these architectures, or is there a wedge between the public's and network providers' private interests in network architectures?

Public Interests in Network Architectures

Social Benefits

As we saw in chapter 8, architectures based on the broad version of the end-to-end arguments provide an economic environment that is more conducive to application innovation than architectures that deviate from it. Apart from differences in the costs and benefits of innovation, the architectures allow different types of innovators to innovate. If there is technological uncertainty or market uncertainty, or if users' needs are heterogeneous, allowing innovators other than network providers to innovate increases the amount and the quality of innovation. If users' needs are heterogeneous, increasing the size and diversity of the innovator pool

will also result in higher product variety that better meets users' needs. In the current Internet, technological uncertainty, market uncertainty, and user heterogeneity are still high. The technology of the network and the technologies for the development of applications are still evolving, creating considerable technological uncertainty. As more people and more businesses connect to the Internet, the population of Internet users will become more diverse. In an ideal situation, if a country manages to bridge the digital divide, the heterogeneity of Internet users will mirror the heterogeneity of society. As a result, the heterogeneity of users' needs is bound to increase. Thus, the conditions under which increases in the size and in the diversity of the innovator pool under an end-to-end architecture result in increases in the amount and the quality of innovation and in applications that better meet users' needs are still met. As a result, other things being equal, an architecture that is based on the broad version of the end-to-end arguments will foster more and better application innovation than an architecture that deviates from that version.

But is more innovation in applications socially beneficial? This question can be approached in several ways. First, one may ask whether the amount of innovation is generally lower than the social optimum. In this case, more innovation would be socially beneficial.¹ In dealing with such questions, economists often note that the link between innovation and social welfare is theoretically ambiguous.² On the one hand, some economic models highlight the possibility that firms, in their desire to capture the rents from innovation, may increase the level of investment in research and development above the socially efficient amount.³ On the other hand, innovators may not be able to completely appropriate the social gains from innovation, which would lead them to invest less than the socially optimal amount in innovation.⁴ For example, innovators may not be able to capture all the consumer value created by their innovation, or other firms may be able to exploit the innovation without compensating the original innovator, either of which will lead to uncompensated spillovers. While the theory is ambiguous, empirical studies indicate that there is too little innovation, because private firms are typically unable to appropriate all social gains from the innovation.⁵

Second, one may ask whether in the specific case of the Internet there is likely to be less innovation than the socially optimal amount. The Internet is a platform product that may be used with a large number of complementary products.⁶ Innovation in platform products and in complementary products is subject to two types of externalities that are likely to reduce the amount of innovation below the social optimum.⁷ The first externality

operates vertically between the platform product and each complementary product; the second operates horizontally among different complementary products.

Owing to the complementarity between the platform product and complementary products, innovation in complementary products usually increases demand for the platform product, and vice versa. If the platform product and the complementary product are developed by different economic actors, the innovator in a complementary component does not appropriate his innovation's positive effects on the platform product, and vice versa.⁸ On the Internet, new applications usually make Internet services more attractive, but an independent application developer does not capture this increase in value.

Innovation in one complementary product usually increases demand for the platform product, which may in turn positively affect demand for other complementary products. If different economic actors pursue innovation in the different components, each actor does not appropriate his innovation's positive effects on the other components. As a result, each actor's incentives to innovate will be lower than the social optimum. In the case of the Internet, application developers do not capture the beneficial effect of their innovation on other, unaffiliated applications.

A common solution to the problems caused by such externalities is integration among all affected parties. The resulting integrated entity internalizes the externalities and therefore has higher incentives to innovate.⁹ In the context of the Internet, this is not a feasible solution, since a few network providers will not be able to identify and realize all beneficial uses of the platform.¹⁰

All this suggests that there will be too little application innovation, and that an increase in application innovation is socially beneficial. Moreover, application innovation is crucial for enabling the Internet to realize its economic, social, cultural, and political potential.

As a general-purpose technology, the Internet has the potential to contribute disproportionately to economic growth.¹¹ Fostering innovation in applications is critical to realizing this potential. Technological progress is the most important engine of growth for modern economies. Economists have estimated that as much as 70 percent of the growth in output per hour in the United States between 1950 and 1993 can be attributed to technological growth.¹² Exactly how and to what degree specific technological advancements contribute to economic growth, however, is less clear.

Research in economics indicates that technological inventions do not contribute equally to economic growth. Instead, over extended periods of

time, technical progress and economic growth seem to be driven by a few general-purpose technologies; some examples are the steam engine, the electric motor, semiconductors, and information technology.¹³ General-purpose technologies offer generic functionalities that can potentially be applied in a large number of sectors within the economy. As the use of a general-purpose technology spreads throughout the economy, use of the technology increases productivity in the sectors in which the technology is applied. At the same time, new applications of the technology or adoption of the technology in additional sectors of the economy increase the returns to innovation in the general purpose technology, triggering new advances in the general-purpose technology itself. These advances, in turn, may spawn the adoption of the general-purpose technology in additional sectors of the economy, or may lead to new or improved applications in sectors that already use the technology. Thus, the adoption of general-purpose technologies exhibits increasing returns to scale. The ongoing dynamic interactions among new or improved uses of the technology, adoption of the technology in additional sectors of the economy and advances in the general purpose technology can create enormous increases in economic growth.¹⁴

Owing to the general nature of a general-purpose technology, the mere existence of such a technology is not sufficient to have a positive effect on economic growth. A general-purpose technology's effects on growth stem from its adoption in more and more sectors of the economy and from the resulting increases in productivity. Owing to the general nature of the technology, however, its potential applications and uses are not immediately obvious. Instead, realizing a general-purpose technology's inherent promises in a specific sector of the economy requires a considerable amount of innovative activity in order to identify and realize potential uses. Thus, adoption of a general-purpose technology in a specific area is an important innovative activity in its own right; for this reason, such activity is often called *co-invention*.¹⁵

As a result, the rate at which a general-purpose technology can affect economic growth depends on the rate of co-invention, not primarily on the rate of technological innovation in the general-purpose technology itself. Thus, the cost of co-invention is an important determinant of the speed with which the social benefits of the general-purpose technology can be realized.¹⁶ In fact, the empirically found delay with which firms' investments in information technology lead to increases in economic growth¹⁷ is usually explained by the high costs of co-invention—that is, by the costs of finding the best ways to apply the new technology in a firm's daily

operations, the costs of developing the appropriate software, and the costs of changing organizational structures and processes in response to the new opportunities.¹⁸

Thus, on the one hand, general-purpose technologies have the potential to contribute disproportionately to economic growth—that is their promise. On the other hand, the rate at which a general-purpose technology can contribute to this growth is limited by the rate at which *new* uses of the technology can be identified and realized.

These insights help us think about the importance of application innovation for economic growth.¹⁹ As a general-purpose technology, the Internet has the potential to contribute disproportionately to economic growth. The ability to communicate cheaply and cost-effectively with computers all over the world may be usefully applied in a large number of contexts. The higher the number of uses, the higher the aggregated increases in productivity and the higher the effect of the Internet on economic growth.²⁰ The rate at which the Internet can contribute to economic growth, however, depends on the rate of co-invention—that is, the rate at which potential uses for the Internet are identified and applications that enable or support these uses are developed, deployed, and used.²¹ Measures that increase the cost of co-invention or otherwise reduce the amount of co-invention can harm social welfare significantly. Specifically, increasing application-level innovation increases economic growth; in contrast, limiting application-level innovation may significantly limit the Internet's ability to contribute to economic growth.

The importance of innovation in applications goes beyond its role in fostering economic growth. As the literature on general-purpose technologies reminds us, the Internet, as a general-purpose technology, does not create value through its existence alone. It creates value by enabling users to do the things they want or need to do.²² Applications are the tools that let users realize this value. For example, the Internet's political, social, or cultural potential—its potential to improve democratic discourse, to facilitate political organization and action, or to provide a decentralized environment for social and cultural interaction in which anyone can participate²³—is tightly linked to applications that help individuals, groups, or organizations do more things or do them more efficiently, and not just in economic contexts but also in social, cultural, or political contexts.²⁴ For example, e-mail, instant messaging, Internet telephony, video chat, or social networking applications let us maintain or deepen our existing relationships with family members, friends, and colleagues, but also establish new relationships with people who share similar interests or goals, or with

people who are interesting to us for other reasons.²⁵ Other applications enable us to organize and coordinate a multitude of tasks—for example, shopping, a class reunion, a fundraiser for a preschool, or a major political campaign—more efficiently.

The World Wide Web, weblogs, open-source programming languages, and audio and video processing software enable anyone who is willing and able to invest the time and effort necessary to master these applications to become a creator of information products at costs low enough to support a wide range of motivations and strategies. E-mail, mailing lists, content versioning systems, and wikis allow loosely joined and geographically distributed individuals to work together to create all sorts of information products, whether they are open-source programs or online encyclopedias. Such applications greatly increase our capacity to produce things alone or in groups, outside the realm of the market. As Yochai Benkler has shown, this capacity to be a speaker, not just a listener, and to actively shape our environment instead of just being at the receiving end, is central to the Internet's potential to enhance individual freedom, to provide a platform for better democratic participation, to foster a more critical and self-reflective culture, and potentially to improve human development everywhere.²⁶

Across all these domains, new and better applications may open up new or better opportunities for users, increasing the Internet's potential to create value. Innovation in applications alone is not sufficient to create value; applications merely increase the potential for value creation. The value promised by applications is realized when users adopt and use these applications to realize their goals.

The kind of value that is realized may differ depending on the type of use. The value of consumptive uses (or uses preparing for consumptive uses) is realized when the user consumes the good. For example, I am entertained when I watch my favorite television show online. The use of a search engine to find the product I want to buy or an online shop to buy the product creates value by preparing my ultimate consumptive use of the good in a way that is more cost-efficient or time-efficient for me, or that helps me find a product that better meets my needs.²⁷

The value of productive uses is more difficult to determine and to measure.²⁸ Consider someone who improves an article on Wikipedia or contributes an article to a website devoted to a certain subject. This person will realize certain benefits from doing so, including the pleasure of being creative and the potential to become recognized as an expert in the field. But the value created by such use of the Internet goes beyond that. People

who read this person's article may learn something. People can benefit from his article even if they are not themselves Internet users. For example, a teacher could use the article to teach her class, or could print it out and distribute it to her students as supplemental reading. The article's beneficial effect may even affect those who never read the article, whether online or offline, if the higher level of education on this topic ultimately helps to improve societal conditions.

Similarly, political bloggers may realize some value resulting from their own blog entries. The political value created beyond that may be relatively tangible when the blogging directly influences the way political events unfold. For example, in 2002, Trent Lott, the US Senate's Republican Majority Leader, resigned from his position after he had commented at Senator J. Strom Thurmond's hundredth-birthday party that, in the words of a journalist, "the United States would have avoided 'all these problems' (Lott) if then-segregationist Strom Thurmond had been elected president in 1948."²⁹ The mainstream media originally ignored Lott's remark, although it had been broadcast on the cable channel C-SPAN. They began to focus on the remark only after it had triggered intense debate and criticism among bloggers. In other cases, bloggers have contributed "talking points" that were later used by members of the US Congress in speeches. Bloggers also made it more difficult for President George W. Bush to gain some support from Democrats for his proposed Social Security reforms by publicly identifying and criticizing Democrats who were thinking of supporting Bush's proposal.³⁰ Debates in the "blogosphere" also may help to highlight interesting or emerging issues for journalists working for traditional media, which in turn may frame the public discourse on those issues.³¹ Most generally, blogs and other forms of online discourse may improve the political discourse on a subject, thereby benefiting even members of society who have not participated in the deliberations.³²

Thus, productive Internet uses often create positive externalities that benefit not only other Internet users but also people who do not use the Internet.³³

Whether users can realize the value promised by particular applications, or, more generally, whether the Internet can fulfill its economic, social, cultural, and political potential, depends not only on the availability of applications, but also on the way in which the architecture of a network shapes the environment for the use of the network.

As we have seen throughout this book, a network's architecture shapes the economic system for the design, development, and production of the network's components by imposing constraints on those who design,

develop, and produce the components. Similarly, an architecture affects the social systems for the use of the network—the actors in these systems, the relationships among them, the governance structures through which they interact, and their behavior—by imposing constraints on those using the network. Of course, architecture is not the only constraint that shapes the options available to users of the system and affects the costs and benefits associated with these options. Architectural constraints interact with other constraints (including laws, norms, and more traditional economic constraints) to establish the set of constraints under which system users operate. The ultimate effect of an architecture on the environment for the use of the system at a specific place and time depends not only on the set of constraints, but also on the characteristics of the actors exposed to the architecture and on their existing and expected relationships with other actors. Depending on these other factors, the same architecture may result in very different actual environments for the use of the network.³⁴

Whereas other scholars have highlighted the role of law, particularly intellectual-property law, in realizing the Internet's political and cultural potential,³⁵ I focus in this book on the specific effect of architecture. To isolate the effect of architecture, I neglect the effect of potential differences in actors or non-architectural constraints and focus on the technological possibilities afforded by or constrained by different architectures without implying that such possibilities can be or will be realized by everyone or everywhere. Although tracing how exactly design principles such as the broad version of the end-to-end arguments shape the environments for the economic, social, cultural, and political uses of the Internet is beyond the scope of this book, a few observations can be offered.

First, network architectures influence who controls which applications and what content can be used. In a network with an architecture based on the broad version and with a deployment context in which users control their end hosts, users can determine independently which applications or what content they want to use. Since the network is application-blind, network providers cannot interfere with users' choices. In contrast, network providers whose networks deviate from the broad version have at least some control over the use of their network. As we saw in chapter 7, this control is maximized in a fully controllable architecture. Most generally, the value of the Internet for users is maximized when they can do the things they need or want to do, using the applications or accessing the content that best suits their needs. Users know best what these are. In contrast, as we saw in chapters 6 and 8, if network providers can control the use of the network, decisions about which applications and what

content to support or exclude may be driven by motivations that are not necessarily identical with users' preferences, preventing users from acting on their preferences and using the Internet in the way that creates the most value for them. For example, if I am working on an open-source project that uses BitTorrent to distribute its source code, and the network provider chooses to single out BitTorrent to manage bandwidth on its network, I am not able to use the application that best meets my needs or to use the Internet in the way that is most valuable to me. Thus, network architectures (such as architectures based on the broad version of the end-to-end arguments with a deployment context in which users control the end hosts) that allow users to decide how they want to use the network will let users realize more of the Internet's potential value for themselves than architectures in which network providers control the use of the network. I am not claiming that users' choices maximize the Internet's value for society. As we saw above, users do not appropriate many of the positive benefits that their productive uses create for other users or for those who do not use the Internet. At the same time, under flat-rate pricing users will not sufficiently consider the negative effects of their uses on other users (for example, negative effects due to congestion). Whether the positive or the negative externalities dominate on balance is difficult to determine. Identifying and implementing ways to encourage users to consider at least some of the negative externalities without overly stifling positive externalities would increase the likelihood that the balance is positive.³⁶ I argue, though, that having users choose applications or content provides more value to society as a whole than having network providers choose for them. This is because, taken together, network providers' limited ability to completely appropriate their users' benefits from the different uses or to appropriate the positive externalities these uses create for users of other Internet service providers or for those not using the Internet, as well as the various other biases and wedges between users' and network providers' interests and the public's and network providers' interests (discussed in chapters 6 and 8 and throughout this chapter), create a larger wedge between the public's interests and the network providers' private interest than is created by the fact that in countries with flat-rate pricing users do not internalize the congestion externalities during times of congestion.

Second, network architectures influence the environment for the use of the network by affecting the ease with which users (whether individuals, groups, firms, or other organizations) can adopt and use new applications that increase their productivity. The ease of adoption, in turn, influences

the rate at which the Internet can contribute to economic growth.³⁷ As we saw in chapters 4 and 7, network architecture influences the costs and requirements for adopting new applications, for example by influencing whether new applications can be adopted incrementally. In this respect, a network architecture based on the broad version establishes the lowest hurdles for adopting applications by letting users adopt new applications incrementally (i.e., independently of other users) at very low costs—the costs of installing the application on their computer.

Third, network architectures influence the Internet's ability to realize its social, political, or cultural potential in subtler ways. As other scholars have shown, the individual's ability to speak and be heard, to be a producer and not just a consumer, and to have access to a wide variety of diverse sources that are not selected or controlled by a central gatekeeper who has its own motivations are central to the Internet's political and cultural potential.³⁸ All these aspects may be affected by the ways in which a network architecture shapes the environment for the use of the system.

Consider some of the ways in which network architectures shape the environment for political uses of the Internet. A network architecture based on the broad version of the end-to-end arguments is not optimized in favor of specific applications. In particular, it is not optimized in favor of end-user applications that receive a lot of data but do not send a lot of data. Such an environment equally supports an individual's ability to speak and listen. In contrast, an architecture that is optimized in favor of applications that download a lot of data but do not upload a lot of data (for example, by providing asymmetric bandwidth to and from the home) favors listening over speaking, consuming over producing, and consumptive uses over communicative uses (such as video chatting or video conferencing) that entail sending and receiving large amounts of data.³⁹

More generally, given that network providers aim to maximize their profits, there is a danger that if network providers are allowed to optimize the network in favor of specific applications they will optimize the network in favor of uses that create observable value that they can appropriate over uses that create less observable and appropriable benefits. As we saw above, the value of consumptive uses is realized by the user itself and is relatively well defined, whereas productive uses often create external benefits for other users (and even for non-users) that neither the productive user nor the network provider can appropriate. This may lead network providers to optimize the network in favor of consumptive uses over productive uses.⁴⁰

Architectures that enable network providers to control the applications and the content on their network give them the power to shape the infor-

mation environment according to their private interests. If I am interested in content that my network provider happens to restrict, my ability to educate myself, my ability to contribute to a discussion on this subject, and my ability to make informed decisions are impeded. Instead, ISPs gain the power to shape public discourse based on their own interests and idiosyncratic content policies. This may affect users in their capacity as speakers and listeners.

Network architectures deviating from the broad version of the end-to-end arguments may also make it more difficult for non-commercial actors to participate in democratic or cultural discourse. For example, by using peer-to-peer file-sharing applications instead of a central server to distribute information goods, these actors can reduce their costs of distribution by shifting some of these costs to users of the file-sharing application who contribute their own bandwidth to assist in the distribution of the file. Network architectures in which network providers manage bandwidth by throttling or excluding these file-sharing applications may reduce the ability of these actors to participate in democratic or cultural discourse by making it more costly for them to make their information available to others.

In sum, network architecture affects the Internet's potential to create economic, social, cultural, and political value not only by fostering or impeding application innovation, but also by shaping the environment for the use of the network. In particular, applying the broad version of the end-to-end arguments ensures user choice, non-discrimination, and non-optimization all of which contribute directly to characteristics of the Internet (such as enabling users to listen, speak, consume, and produce, and to do so without intervention by the network provider) that are central to the Internet's social, cultural, and political potential.⁴¹

Social Costs

Using the broad version also has social costs.

First, as we saw in chapter 2, optimizing the network for particular applications or uses may create cost savings or performance gains for the applications for which the network is optimized. Keeping the network general, as the broad version requires, means forgoing these benefits.

Second, as we saw in chapters 6 and 7, deviating from the broad version by increasing the controllability of the network increases network providers' profits. Lower profits may reduce network providers' incentives to deploy more and better broadband networks. I am not convinced that the relatively lower profit in a network architecture based on the broad version

reduces network providers' profits so significantly that they will no longer be able to deploy more and better broadband networks, or that network providers would really use the additional profits to deploy more networks instead of using the money for other things such as increasing dividends. But until we have data with which to better assess whether this is a real problem, it is safer to assume that network providers' incentives to deploy more and better broadband networks will be reduced to some degree.

Third, some scholars have argued that the end-to-end arguments constrain the needed evolution of the network's core by preventing Quality of Service, restricting the amount of state in the network, or requiring the network to be simple. As we saw in chapter 3, these assumptions are unfounded. The broad version provides much more room for network evolution than its critics recognize. It does, however, constrain the evolution of the network to some degree by requiring the network to be general, or by allowing only some forms of Quality of Service.

Finally, critics argue that the broad version makes it more difficult, or impossible, to make the network secure. Some seem to think that the broad version of the end-to-end arguments requires all functions related to security to be implemented at the end hosts. On this perception, some have argued that a network based on the broad version makes it impossible to defend against distributed denial-of-service attacks, in which distributed attackers flood the target (which may be a link or an end host) with unwanted traffic that exceeds the target's capacity to handle it, making it impossible for the target to properly handle its legitimate traffic. This argument is based on the view that distributed denial-of-service attacks can be identified only in the network, since only the network can recognize traffic patterns that may indicate an attack.⁴²

Others have argued that the broad version of the end-to-end arguments, by assigning responsibility for security to the end hosts, creates challenges for users who have little knowledge of how to best secure their computers. Jonathan Zittrain has argued that the existence of a large number of unsecured end hosts may facilitate a catastrophic security attack that will have widespread and severe consequences for affected individual end users and businesses. In the aftermath of such an attack, Zittrain claims, users may be willing to completely lock down their computers so that they can run only applications approved by a trusted third party.⁴³ Given that general-purpose end hosts controlled by users rather than by third-party gatekeepers are an important component of the mechanism that fosters application innovation in the Internet,⁴⁴ Zittrain argues, following the broad version of the end-to-end arguments may threaten the Internet's ability to support

new applications more than deviating from the end-to-end arguments and implementing some security functions in the network, because, according to Zittrain, following the broad version will ultimately result in a world in which end hosts are effectively controlled by third parties, through the sequence of events just described.⁴⁵ Again, this argument relies heavily on the assumption that the end-to-end arguments prevent the implementation of security-related functions in the network.⁴⁶

The perception that the broad version of the end-to-end arguments requires “security” to be implemented at the end hosts is too strong.⁴⁷ The broad version provides guidelines for the placement of individual functions, not groups of functions belonging to a certain category. Thus, whether the broad version requires a function to be implemented at the end hosts or in the network must be determined function by function.⁴⁸

For example, if it is true that distributed denial-of-service attacks can be identified and stopped only in the network,⁴⁹ the broad version clearly allows the implementation of the associated functions in the network. After all, the broad version does not prevent functions from being implemented in the network if they cannot be completely and correctly implemented at the end hosts only.⁵⁰

In contrast, according to the broad version, a function (such as encryption) that can only be completely and correctly implemented end-to-end between the original source and ultimate destination of data should not be implemented in the network. This is because “a function or service should be carried out within a network layer only if it is needed by all clients of that layer, and it can be completely implemented in that layer.”⁵¹

Thus, depending on the function, the broad version may not require that the function be implemented at the end hosts. But even if the broad version requires a function to be implemented at the end hosts, considerations such as those advanced by Zittrain may require a deviation from the broad version. More generally, if end users’ inability to secure their end hosts makes it impossible to provide “effective” security by implementing functions solely at the end hosts, it may be necessary to implement them in the network even if the broad version would generally require these functions to be implemented on end hosts.⁵² As we saw in chapter 2, the broad version is based on the assumption that long-term evolvability, reliability, and application autonomy are more important than the performance gains or cost savings resulting from the short-term optimization of the system. For reasons that were set out in detail in chapter 2, factors that were considered in this trade-off usually cannot justify a deviation from this rule. For example, the performance gain from implementing

a certain application-specific function in the network cannot usually justify a deviation from the broad version, as the loss of these benefits has already been factored into the trade-off underlying the decision to use the design principle. In contrast, the effect of the broad version on the security of the system was not similarly considered as part of the original trade-offs (i.e., the trade-off underlying the design principle and the trade-off underlying the decision to use this design principle for a particular system). Insofar as maintaining the security of the Internet is a value that fundamentally affects the usefulness of the Internet for its users, and ultimately for society as a whole, deviating from the broad version of the end-to-end arguments for certain security-related functions may be justified.

This does not mean that any implementation of security-related functions in the network is automatically justified. Some solutions may affect the evolvability of the network and its ability to foster innovation more than others, and it is worth investing the time, money, and effort required to find solutions that maintain the Internet's openness to new applications as much as possible.⁵³ As a negative example, firewalls (which deviate from the broad version) have been particularly harmful for application development, and, together with network-address translators (which also deviate from the broad version), have made it difficult if not impossible to deploy new transport-layer protocols or certain types of new applications.⁵⁴ Thus, whether a specific security-related function should be implemented in the network although the broad version of the end-to-end arguments requires that this function be implemented at the end hosts can only be determined case by case after careful consideration of the consequences for the evolvability of the network. The discussion above shows, however, that the broad version is able to interact with and adapt to other requirements in a way that is flexible enough to maintain the design principle's usefulness in the face of changing requirements for the Internet.

In sum, the broad version of the end-to-end arguments constrains solutions to security problems much less than is often assumed. Still, protecting the evolvability of the system constrains the range of potential solutions more than architectures deviating from the broad version, in which concerns about evolvability do not play a role.

Trade-Offs

Thus, there are four potential social costs of using the broad version of the end-to-end arguments: the forgone benefits of short-term optimization (e.g. performance gains or cost savings), some limits on the evolvability of

the network's core, some constraints on how security problems can be solved, and potentially lower incentives for network providers to deploy more and better broadband networks relative to network providers in more controllable network architectures.

These social costs must be traded off against the social benefits identified above: other things being equal, network architectures based on the broad version result in more and better application innovation and in applications that better meet the needs of Internet users. As we saw, fostering innovation in applications is not only critically important for economic growth. More applications also increase the Internet's potential to create value in the social, cultural, and political domains. In addition, network architectures based on the broad version let users realize more of the Internet's potential value, both for themselves and for society as a whole. Finally, the broad version directly influences characteristics of the environment for the social, cultural, and political uses of the Internet that are central to the Internet's potential to enhance individual freedom, provide a platform for better democratic participation, foster a more critical and self-reflective culture, and potentially improve human development everywhere.⁵⁵

How should we compromise among these values? It may help to think about them in two steps. In a way, the first three costs constitute the price of having a system that can evolve. A system can either be optimized for the short-term or be evolvable, but not both. The trade-off closely resembles the technical trade-off underlying the broad version as a design principle that trades off the benefits of long-term evolvability against the advantages of short-term optimization. Is this fundamental trade-off still appropriate? If we believe that all important applications have been realized, there is no need to incur the costs of keeping the Internet open for new applications. If, however, we believe that there are many more potential applications waiting to be discovered and developed, it is a comparably low cost to forgo the benefits of optimization, or to live with some limits on the evolvability of the network's core.⁵⁶

It would be premature to say that the only improvements left to make on applications are incremental. As the Internet permeates more and more areas of society and the economy, its potential uses come to mirror the diversity and variety of human life. Apart from being used to support existing activities, Internet applications may enable activities that were not possible before. In addition, new peripherals and advances in the services provided by the Internet infrastructure (e.g., the mobile Internet, the potential introduction of Quality of Service, or increases in available

bandwidth and computer processing power) continuously create new opportunities for applications that take advantage of these features. Together, these ongoing technical developments at all layers of the Internet, in addition to increases in the number and the diversity of users and potential uses, make it likely that existing applications have only begun to tap the Internet's vast potential.

This leaves the more fundamental trade-off between broadband deployment and the benefits associated with the broad version of the end-to-end arguments.

Without access to broadband networks, users cannot realize the value of the broadband Internet and cannot benefit from the new applications that better broadband networks enable. Thus, widespread broadband deployment is clearly an important social goal. At the same time, for a variety of reasons (including spillovers), network providers do not capture all of the value created by broadband networks. All this suggests that network providers' private interests in deploying broadband networks may be lower than the social optimum.⁵⁷

Thus, both social benefits *and* social costs of the broad version represent important social values, which makes it difficult to resolve the trade-off. Exploring whether the social benefits associated with one architecture could be realized even if this architecture is rejected may help. If we maintain the broad version of the end-to-end arguments so as to realize its social benefits, are there other ways to solve the problem of network deployment? Although it is non-trivial to finance and foster investment in infrastructure, the problem can be solved in a number of ways. If we truly believe that network providers need more profits to motivate them to deploy broadband networks, tax cuts on broadband investments or direct subsidies may be an option. An alternative solution may be to think about public provision of infrastructure.

If we let network providers increase the controllability of their networks in order to get more broadband networks with higher capacity, can we realize the benefits associated with the broad version in other ways? As we saw in chapter 8, under uncertainty or consumer heterogeneity, the two crucial ingredients for creating successful innovation in applications are (1) independent, low-cost application innovation by a large and diverse set of innovators and (2) user choice among the resulting applications. Letting network providers discriminate against applications or letting them increase the costs of innovation through access charges breaks this mechanism. Recreating a similar mechanism through other means (e.g., through subsidies targeted at potential application innovators) does not

seem feasible. On the one hand, in an architecture based on the broad version, almost anyone is a potential innovator. On the other hand, no one can identify successful innovators and innovative ideas in advance.

Similarly, both the Internet's value for individual users and for society and the Internet's potential to realize important social, cultural, or political values depend fundamentally on user choice, non-discrimination, and non-optimization of the network, all of which are achieved by applying the broad version of the end-to-end arguments. Thus, letting network providers increase their profits by increasing the controllability of the network removes the very features of the network that make the Internet valuable.

Thus, whereas it is possible to have a network architecture based on the broad version, realize the associated social benefits, and then solve the deployment problem through other means, it is not possible to have a controllable architecture and realize the benefits associated with the broad version through other means.

Under these circumstances, sacrificing the very aspects that drive the Internet's value in order to get more broadband networks seems too high a price to pay; as Tim Wu put it, it is like selling the painting to get a better frame.⁵⁸ All this suggests that the trade-off among the social costs and benefits associated with the broad version is positive, so that the broad version of the end-to-end arguments is in the public interest.

Network Providers' Private Interests in Network Architectures

If using the broad version of the end-to-end arguments is in the public interest, do network providers share this interest, or do their private interests make them prefer architectures that deviate from the broad version?

As we saw in chapter 6, increasing network providers' ability to monitor and control applications and content on their networks increases their profits. They can increase the profitability of their own applications or content by excluding or discriminating against competing applications. In addition, the more sophisticated pricing strategies supported by more controllable architectures let them extract more of the value realized by users or by providers of applications and content. Network providers seem to be moving rapidly in this direction. They increasingly deploy devices for deep packet inspection in their networks that let them monitor and control the applications on their network. These devices can be used to slow down or exclude specific applications and content, to implement sophisticated pricing schemes, or to offer Internet-service packets that

allow consumers to use some applications but not others.⁵⁹ Large network providers in the United States, in the United Kingdom, and in Germany have publicly stated their desire to charge applications or content providers for the right to have their applications or content travel over the network providers' "last-mile" networks to reach the network providers' Internet-service customers.⁶⁰ And incumbent network providers (including AT&T and Verizon in the United States and BT in the United Kingdom) have begun to deploy in their wireless and wireline networks a technology called *IP Multimedia Subsystem (IMS)* that adds to conventional IP networks functionality that gives network providers fine-grained information about and perfect control over the applications and the content on their networks. IMS also provides sophisticated charging capabilities that let the network provider charge both end users and providers of applications or content.⁶¹ White papers by industry vendors laud IMS's ability to support flexible monetization models that convert the network from a "data highway to a data tollway."⁶² Thus, in the terminology of chapter 7, IMS is an example of a "fully controllable architecture."

At the same time, the network has become more opaque. Recent years have seen a proliferation of technical solutions, driven by the short-term interests of particular actors, that do not take account of their effects on the long-term evolvability of the Internet. They solve the problem at hand, but at the same time they reduce the generality of the Internet, with significant costs for application-level innovation.⁶³ For example, residential broadband networks provide asymmetric bandwidth for uploads and downloads, creating problems for applications that send and receive equal amounts of data.⁶⁴ Network-address translators and firewalls deployed by network providers and users have made it increasingly difficult to deploy new transport-layer protocols or new applications whose behavior deviates from that of traditional client-server applications.⁶⁵ Network providers are using deep packet inspection to single out and slow down specific applications in order to manage bandwidth on their networks.⁶⁶ Though asymmetric bandwidth seems to have been motivated by the available cost savings, the other deviations from the end-to-end arguments seem to occur as a by-product of some actors' attempts to solve specific technological problems without considering the effect on the evolvability of the Internet as a whole. Often, as in the case of network-address translators and firewalls, the negative consequences take some time to become apparent.

Thus, the evidence suggests that network providers have an interest in deviating from the broad version. This may be surprising. After all, the broad version fosters application innovation. More applications make a

network provider's Internet service more attractive, which should enable the network provider to sell more Internet service or increase its price. But if following the broad version increases network providers' profits, shouldn't this lead them to *prefer* the broad version over architectures that deviate from that version? This line of thought neglects that the potential increase in profits from more application innovation is not the only aspect network providers consider when choosing network architectures. Network providers compare the private costs and benefits of complying with the broad version with the private costs and benefits of deviating from it. Although more application innovation may increase their profits, increasing the controllability of the network may increase their profits even more. And whereas the economic and non-economic benefits of application innovation for society weigh heavily in the trade-off among the social benefits and the social costs, network providers consider their private costs and benefits, which may be different.

Although network providers receive some benefits from application innovation, two factors will lead them to undervalue the benefits of application innovation compared to the social optimum. First, they do not appropriate all the gains from application innovation. Second, the gains they are able to capture are uncertain and will be realized in the future, which may lead them to take even less account of those gains.

Network providers do not appropriate all of the gains from application innovation if they innovate themselves, much less if the application has been developed by someone other than the network provider. If consumers value the application differently, network providers cannot appropriate the entire consumer surplus associated with the use of the application. As we saw in chapter 6, this inability is particularly marked in an architecture that is based on the broad version of the end-to-end arguments, which forces the network provider to charge a uniform per-packet transport price that does not vary depending on the application being used. Because the network provider isn't able to collect access charges under an end-to-end architecture, it cannot extract all the value that application developers realize from the application, nor can it capture the positive externalities resulting from its users' productive uses that accrue to users of other Internet service providers and even to people who do not use the Internet at all. Finally, it does not capture the increase in the value of other network providers' Internet service resulting from additional application innovation. Thus, a network provider's private gain from an application innovation will often be much lower than the social benefits associated with that innovation. As a result, network providers value the

increase in application innovation brought about by the broad version less than is socially optimal.

In addition, intertemporal biases (that is, biases affecting decisions involving trade-offs among outcomes that will have their effects at different times)⁶⁷ may bias network providers or their decision makers toward short-term outcomes. Whereas the private benefits of deviating from the broad version are realized immediately, most of the private benefits of complying with the broad version are uncertain and will be realized in the future. Given this intertemporal distribution of network providers' private benefits, a bias in favor of short-term gains would further exacerbate the wedge between network providers' interests and the interests of the public.

First, research in behavioral economics shows that individuals often discount future benefits more than rational discounting would suggest—a bias known as “hyperbolic discounting.”⁶⁸ In the extreme case, this bias may cause individuals deciding whether to deviate from the broad version of the end-to-end arguments to prefer the smaller *current* private benefits associated with deviating (i.e., the performance gains, or the increase in profits from discrimination or pricing strategies that extract more of the value realized by users or independent application developers) over comparatively larger *future* private benefits associated with following the broad version (such as the increase in profits from Internet service due to increases in the number and variety of applications), even if rational discounting would lead them to prefer the future benefits. If under rational discounting the private current benefits of deviating from the broad version were larger than the private future benefits of using it, this bias would further reinforce network providers' preference for deviating from the broad version.

Second, since the future applications that may make network providers' Internet service more valuable are not known, it is difficult to know their worth. Uncertainty about the future further reduces the weight that future developments receive in current decision making.⁶⁹

Third, researchers suggest that firms may generally be biased toward the short term.⁷⁰ For example, theoretical research and some empirical evidence suggest that the stock market favors short-term benefits over long-term gains.⁷¹ If the stock market exhibits such a bias, we should expect network providers to follow the same valuation in order to please the stock market. Alternatively, managers' desire to enhance their reputation quickly may lead them to focus on actions that create high and observable short-term gains, without caring about the long-term effect, if they expect to

have left the firm or their specific position when the long-term consequences of their actions occur.

Finally, network providers that currently dominate the market may not know whether they will be able to sustain their lead. For example, technical developments may reduce the costs of deploying broadband networks. Lower costs may motivate additional firms to enter the market, or may motivate regulators to artificially introduce competition. In both cases, network providers' expected share of benefits from innovation in applications will be lower in the future, when they have a less dominant position, which again would lead them to discount their share of future benefits from application innovation more.

All this suggests that network providers will often resolve the trade-off between the increased profits, performance gains, or cost savings from deviating from the broad version of the end-to-end arguments and the potential loss of application innovation in favor of deviating. Thus, the public's and the network providers' interests in network architecture diverge, creating a market failure regarding the evolution of the architecture.

Conclusion

Many people have a pragmatic attitude toward technology: they don't care how it works, they just want to use it. With regard to the Internet, this attitude is dangerous. As this book has shown, different ways of structuring the Internet result in very different environments for its development, production, and use. If left to themselves, network providers will continue to change the internal structure of the Internet in ways that are good for them, but not necessarily for the rest of us—individual, organizational or corporate Internet users, application developers and content providers, and even those who do not use the Internet. If we want to protect the Internet's usefulness, if we want to realize its full economic, social, cultural, and political potential, we need to understand the Internet's structure and what will happen if that structure is changed.

The End-to-End Arguments

The original architecture that governed the Internet from its inception to the early 1990s was based on a design principle called *the end-to-end arguments*. There are two versions of the end-to-end arguments, and both of them shaped the Internet's original architecture. The narrow version was first identified, named, and described by Saltzer, Reed, and Clark in 1981.¹ The broad version was the focus of later papers by these authors.² Although Saltzer, Reed, and Clark never explicitly drew attention to the change in definition, there are real differences between the two versions—differences in scope, in content, and in validity—that make it preferable to distinguish between the two.

The narrow version applies solely to functions (called *end-to-end functions*) that can only be completely and correctly implemented end-to-end—that is, between the original source and ultimate destination of data. This version can be summarized as follows: A function should not be

implemented in a lower layer if it cannot be completely and correctly implemented at that layer. Sometimes an incomplete implementation of the function at the lower layer may be useful as a performance enhancement.³ The narrow version prescribes that, for reasons of correctness, these functions must be implemented at a layer where a complete and correct implementation is possible—that is, at a higher layer that operates end-to-end between the original source and ultimate destination of data. Also implementing a necessarily incomplete version of these function at a lower layer is redundant and usually constitutes an inefficient use of system resources, but the narrow version recognizes that in some cases an additional, necessarily incomplete implementation at the lower layer may be justified as a performance enhancement.

Whereas the narrow version applies only to functions that cannot be completely and correctly implemented at lower layers, the broad version provides a general rule for distributing functionality in a multi-layer system and for dividing functionality between end systems and the network's core in multi-layer communication networks. According to the broad version, lower layers of the system—the network's core—should provide only general services that can be used by all applications. The network should not be optimized to better support specific higher-layer applications. Though this may increase the performance of particular applications, it is an unnecessary and therefore inefficient feature for applications that do not need this function; it may even rule out the implementation of applications not foreseen at the time of design. Finally, the broad version advises that application-specific functionality be concentrated in the higher layers of the system, at the end hosts. The broad version does not prevent functions from being implemented in the network if they cannot be completely and correctly implemented at the end hosts only.

In the context of the broad version, “end hosts” and “core of the network” denote a purely functional relationship between users and providers of communication services. Whereas end hosts run application programs that use the communication services provided by the network, computers in the core of the network provide these services. The terminology does not refer to topological relationships or to administrative ownership and control.

Whereas the narrow version, if it applies, must be followed for reasons of correctness, the broad version has other justifications. For all functions other than end-to-end functions that must be implemented end-to-end for reasons of correctness anyway, the broad version's design rules emerge from a trade-off. They prioritize long-term system evolvability, application autonomy, and reliability over short-term performance optimizations.

Thus, it is appropriate to use the broad version only if the system's requirements imply that this trade-off is justified. If the system's requirements imply a different resolution of this trade-off, the broad version should not be applied.

Although the broad version of the end-to-end arguments does not explicitly prohibit the core of the network from being aware of the applications running over it or being capable of controlling their execution, applying this design principle results in a network that does not have this functionality. Thus, though the broad version does not directly require the network to be non-discriminatory or "neutral," there is a causal relationship between the broad version and "network neutrality." A neutral network (that is, a network in which network providers cannot interfere with the applications and content on their network) is a direct consequence of applying the broad version. This characteristic represents a benefit for independent application developers, for users of the Internet, and for society as a whole. From the point of view of network owners, it constitutes a cost of architectures based on the broad version.

The End-to-End Arguments and the Original Architecture of the Internet

The original architecture of the Internet resulted from a combination of a variant of the layering principle called *relaxed layering with a portability layer* and the end-to-end arguments. Layering, in turn, is a special form of modularity.

The architecture of the Internet consists of four layers: the link layer (the lowest layer), the Internet layer, the transport layer, and the application layer.

The application of "relaxed layering with a portability layer" imposes constraints on the interactions among protocols belonging to different layers and on the number of protocols that may populate a layer. The Internet layer is designated as the portability layer. Layers above the Internet layer are allowed to invoke the services of protocols in the same layer or in any layer below them, down to the Internet layer, but they are not allowed to directly invoke the services of protocols in a layer below the Internet layer. Lower-layer protocols are not allowed to invoke the services of higher-layer protocols. Whereas the Internet layer is restricted to a single protocol (the Internet protocol), other layers may host a variety of different protocols.

Designating the Internet layer as the portability layer influenced the design of the Internet protocol, since a portability layer aims to provide a technology-independent interface to the services of the network.

During the initial design of the Internet's architecture, the end-to-end arguments shaped the division of functionality between the Internet layer and the transport layer. Whereas the narrow version required implementing functions such as reliable data transfer, duplicate suppression, and reordering end-to-end between the original source and ultimate destination of data (i.e., end-to-end between end hosts), the broad version influenced the division of functionality between the Internet layer and the transport layer.

The Internet's designers chose to implement such end-to-end functions as reliable data transfer, duplicate suppression, and reordering at the transport layer. This design fulfils the narrow version's requirement for an end-to-end implementation of these functions if the transport-layer protocols operate end-to-end between the original source and ultimate destination of data and no events falling into the responsibility of the function occur after the data leaves the transport layer.

The broad version of the end-to-end arguments shaped the Internet's original architecture by determining the following design choices. First, the functionality required to transfer data from host to host is separated from the end-to-end functionality required to control the end-to-end transmission of data. The two groups of functionality are placed in separate layers: the Internet layer and the transport layer. The Internet layer is the highest layer running on computers in the core of the network and provides a general service that is broadly useful across applications: unreliable connectionless datagram service. The transport layer is the lowest layer that only has to be implemented on end hosts and is the lowest layer implementing application-specific functionality.⁴ Second, functionality provided end-to-end by the transport layer is not duplicated in the Internet layer. In particular, the Internet layer does not implement additional hop-by-hop error control.

Apart from guiding the division of functionality between the Internet layer and the transport layer, both versions of the end-to-end arguments influenced, and continue to influence, the design of individual applications.

If the transport-layer protocols do not operate end-to-end between the original source and the ultimate destination of data, or if events falling into the responsibility of the end-to-end function occur after data leaves the transport layer, the narrow version requires end-to-end functions to be implemented in layers above the transport layer, ultimately in the application itself.

The broad version constrains the design of individual applications by requiring that application-specific functionality be implemented in

higher layers at the end hosts, not in the core of the network. Within these boundaries, designers are free to choose any application designs they like.

The use of the end-to-end arguments and of relaxed layering with a portability layer shields transport-layer protocols and application-layer protocols from changes below the Internet layer. At the same time, since the broad version of the end-to-end arguments concentrates application-specific functionality in higher layers, lower-layer protocols are not affected by changes in applications or other protocols above the Internet layer. Thus, the Internet layer divides the protocol stack into two quasi-independent subsystems that can evolve independently.

The Broad Version of the End-to-End Arguments and Innovation in the Original Architecture of the Internet

Network architectures (and the design principles that shape them) influence the economic system that develops and produces the Internet in a variety of ways. In particular, they affect which actors can innovate and what incentives they have to do so.

Applying the broad version of the end-to-end arguments creates an environment that is more conducive to application innovation than architectures that deviate from this design principle. The gradual transition from a network architecture based on the broad version of the end-to-end arguments to one that implements a considerable amount of application-specific functionality in the network's core significantly increases the costs of innovation for independent application developers. It reduces their incentives to innovate, limits the range of potential innovators, and gradually shifts control over the development and deployment of application-level innovation from application developers and users to network owners. In a network architecture based on the broad version of the end-to-end arguments, anyone with access to an end host and the ability to learn a programming language is a potential innovator. At the same time, innovators independently decide how they want to innovate. Users independently decide which applications they want to deploy. In contrast, in a network with a significant amount of application-specific functionality in the network's core, network owners may be the only remaining potential innovators. They also control which applications get deployed. At the same time, the costs of adopting new applications rise for both users and for network owners. As we saw in chapters 8 and 9, these changes significantly reduce the overall amount, the type, and (potentially) the quality

of innovation, at significant costs to society. If there is market uncertainty, business uncertainty, or user heterogeneity, the effects of differences in the size and diversity of the innovator pool and of differences in control over application development and deployment are particularly profound.

Thus, the mechanism that produces innovation in applications under an end-to-end architecture has two components: widespread experimentation by a large and diverse group of innovators who independently select whether to realize their innovative ideas and user choice among the resulting applications. Under uncertainty or user heterogeneity, this mechanism will produce more and better applications than innovation in network architectures that concentrate control over innovation or deployment in the hands of a few network providers. If users' needs are heterogeneous, it will also produce more diverse applications that better meet users' needs.

The broad version's various influences on innovation in applications can be traced back to two features of end-to-end architectures. First, developing and deploying applications requires changing only end hosts that want to use the application. It does not require any changes to the network's core. As a result, innovators can decide independently whether to realize their ideas for an application, and can do so at low costs. Second, the network cannot discriminate against the applications running over it and cannot exclude applications. As a result, independent application developers need not fear that their application will be excluded or discriminated against, or that the network provider will try to extract their profits through access charges. Through mechanisms that were examined in chapters 4–6 and 8, these features influence the ability of economic actors other than network owners to participate in application-level innovation, their incentives to do so, and the types of applications they will develop.

Though the broad version of the end-to-end argument also affects innovation in the core of the network, it constrains the design and evolution of the core of the network less than is often assumed. The broad version requires the network to provide only general services that are useful for a large variety of existing and potential future applications. It prohibits optimizing the network in favor of specific applications and implementing application-specific functionality in the network's core. It does not require the core of the network to be "stupid" or simple. It does not constrain the amount of state in the network. Finally, it does not generally prevent Quality of Service, although it does constrain how Quality of Service can be realized. Thus, it leaves considerable room for innovation at the Internet layer and below.

There are, however, two other mechanisms that make it very difficult to change the core of the network. First, the large number of higher-layer protocols that use the Internet protocol constrain its potential evolution by creating large costs of system adaptation. Second, the deployment context provided by the commercial Internet constrains the Internet layer's evolution even more than an analysis based on the structural dependencies among the Internet's components would suggest.

Contrary to other potential network architectures, the original architecture of the Internet facilitates innovation below the Internet layer by designating the Internet layer as a portability layer. Since the Internet protocol provides a technology-independent interface to the services provided by the network, and because higher-layer protocols are not allowed to directly access protocols below the Internet layer, the underlying physical network technologies can evolve freely without requiring changes in protocols above the Internet layer as long as the service interface provided by the Internet layer stays the same.

End-to-End Arguments in the Current Internet

In recent years, the architecture of the Internet has evolved in ways that challenge both versions of the end-to-end arguments.⁵ For one thing, many application services have evolved, or are evolving, toward a more distributed structure with application-level intermediaries that are topologically in the network and under the administrative control of a network provider or a third-party service provider. In addition, recent years have seen a proliferation of intermediary devices in the path between two end hosts that are not an intended part of the application executed between the two hosts, but perform functions other than the normal, standard functions of an IP router on the datagram path between a source host and a destination host.

Mature application services often evolve toward a more distributed structure, with application-level intermediaries that are topologically in the network and under the administrative control of a network provider or a third-party service provider (figure 3.7).⁶ E-mail servers and Web caches are examples of such intermediaries. (For a description of the structure of e-mail, see box 3.6.) If such an intermediary is an integrated part of the application, correctly terminates the protocol stack, and implements its functionality above the Internet layer, this evolution does not pose any problems for the broad version of the end-to-end arguments.⁷ After all, designers are free to design their applications in any way they want, as

long as the application-specific functionality is completely contained in layers above the Internet layer. Since all the application-specific functionality is restricted to end hosts, an application with such intermediaries does not interfere with the deployment of new applications over the Internet.

From the perspective of the narrow version of the end-to-end arguments, these developments are highly relevant. Implicitly, the original architecture of the Internet was based on the assumption that application-layer and transport-layer protocols would operate directly between the two ultimate end systems of an application session (figure 3.3). Under these assumptions, the implementation of end-to-end functions (i.e., of functions that can only be completely and correctly implemented between the original source and ultimate destination of data) in the transport layer met the requirements of the narrow version of the end-to-end arguments for most applications, if no additional events relevant to the correct execution of the functions occurred after the data left the transport layer.

Today these assumptions are not automatically justified. The distributed application services described above consist of a concatenation of application-layer connections (and, consequently transport-layer connections); as a result, the transport-layer protocols no longer operate end-to-end between the ultimate end systems of the application. With respect to the path between the original source and ultimate destination of data, the transport-layer protocols operate hop-by-hop (figure 3.4). In this context, we cannot assume that end-to-end functions are executed correctly just because the application layer uses a transport-layer protocol that implements these functions. Instead, the narrow version of the end-to-end arguments requires that the end-to-end functions be implemented at a layer operating end-to-end between the original source and the ultimate destination of data, ultimately in the application itself.⁸

Though the increasing delegation of functions to parts of an application that are under the control of third parties such as network providers or service providers does not affect how easily the network can evolve, it raises important questions about the autonomy, the control, and the trust of end users.⁹ Some have suggested that these new concerns make the old functional distinction between “end hosts” and “network” as users and providers of communication services underlying the broad version irrelevant.¹⁰ I do not agree with this assessment. The broad version focuses on the evolvability of the network. In this context, a functional interpretation of “end hosts” and “the network” is still appropriate. As we have seen throughout this book, keeping the network general and open for new applications is still an acute and important problem. Just because new developments

raise new issues that are orthogonal to the question of network evolvability does not mean that the old questions have become irrelevant.¹¹

Furthermore, recent years have seen a proliferation of intermediary devices in the path between two end hosts that are not an intended part of the application, but which perform functions other than the normal, standard functions of an IP router on the datagram path between a source host and a destination host.¹² Most of these devices are topologically close to the edge of the network.

A large group of devices are explicitly aimed at controlling the execution of applications running over the network. Not surprisingly, these devices violate the broad version of the end-to-end arguments: they operate at the Internet layer, but they access parts of the datagram that contain the messages passed between higher-layer protocols. They then use the information gained through this layer violation to determine how to treat the datagram. To determine the appropriate treatment, they must implement application-specific functionality in the network.¹³ Their reasons for existing differ. Some, such as firewalls, are intended to improve security. Others, such as devices for deep packet inspection, are explicitly intended to give network owners the ability to influence how applications run over their network.

Finally, the perceived shortage of IPv4 addresses has led to the emergence of so-called network-address translators (NATs) that let a number of end hosts simultaneously share the same Internet address.¹⁴ NATs operate at the Internet layer and are intended to be transparent to the applications that traverse them. To perform their function, they access and modify parts of the datagram that contain the messages passed between higher-layer protocols—a clear layer violation.¹⁵ In addition, they need to know how higher-layer protocol data messages are formatted in order to function correctly, which violates the broad version of the end-to-end arguments.¹⁶

Whereas devices that let network providers influence the applications and content on their networks affect application innovation by reducing independent application and content developers' incentives to innovate, devices such as NATs and firewalls directly impede the development and deployment of new applications and transport-layer protocols by increasing the costs of developing applications and forcing innovators to coordinate their innovations with others. The effect of these devices goes beyond their ability to block applications they are intended to block.¹⁷ They can have unwanted side effects. Designers developed these devices with the needs of client-server applications such as the World Wide Web

in mind. Applications with different characteristics often cannot pass through the device; in this case, users behind the firewall or NAT will not be able to use the application. For example, peer-to-peer applications, applications that use UDP, and applications that use one signaling connection to set up a second connection may have difficulties passing through NATs or firewalls.¹⁸ To let these applications traverse the device, software specifically written for each particular application must be installed at the device. This not only complicates application design; it also may be very difficult to realize in practice. Often, commercial NATs and firewalls are closed boxes running proprietary software. As a result, application designers who want to let their applications traverse through such a device have to convince the device's vendor to write and distribute the necessary software. Obviously, device vendors may not be very motivated to do so for new applications with an unknown market base.¹⁹ To deal with these problems, developers of affected applications develop complex workarounds or try to design their application in ways that make it as similar to HTTP as possible. This significantly impedes application development—for example, the developers of Skype spent a considerable amount of the overall development time developing mechanisms that would enable Skype to work through NATs and firewalls.

Moreover, NATs and firewalls have made it almost impossible for new transport-layer protocols to be deployed.²⁰ The designers of transport-layer protocols face a “chicken and egg” problem: an application using this protocol can run on a computer only if the protocol is deployed on that computer (usually transport-layer protocols are implemented as part of the operating system).²¹ Since they would like their application to work for all potential users, application designers will not write applications using the new protocol unless it is widely deployed and can be expected to work end-to-end. Operating system vendors will implement a new transport-layer protocol only if they expect application developers to use it. And NAT and firewall vendors do not have an incentive to add support for a new transport protocol if it is not part of a commonly used operating system. Without support, the new transport-layer protocol will not be able to work for users behind firewalls and NATs, and this prevents application developers from using it. Even if firewall and NAT vendors decide to add support for the protocol to their devices, it will take a long time until devices with the new software have been ubiquitously deployed.

These examples show how seemingly minor deviations from the broad version to solve specific technical problems can add up to severely constrain the evolvability of the Internet's architecture.

Though the broad version does not establish any requirements for the end hosts, a sufficient number of end hosts must be general (that is, able to support a large variety of applications), and must be under the control of application developers and end users, if the mechanism that produces application innovation under an end-to-end architecture is to work. This suggests that widespread replacement of general-purpose end hosts controlled by users with specialized appliances controlled by the manufacturer or the network provider (e.g., mobile phones, video game controllers, TiVos), should it occur, might be problematic.²²

Policy Implications

The architecture of the Internet changes in ways that deviate from the broad version of the end-to-end arguments. These changes have direct consequences for public interests because Internet architectures based on the broad version of the end-to-end arguments shape how the Internet is developed, produced, and used. As a general-purpose technology, the Internet does not create value through its existence alone. It creates value by helping users do what they want to do, or letting them do so more efficiently. Applications are the tools that let users realize this value. By fostering innovation in applications, the broad version of the end-to-end arguments increases the potential of the Internet to create even more value in the future. In addition, fostering application innovation is critically important for economic growth.

In a network based on the broad version, users, not network providers, decide how to use the network. Network providers cannot interfere with these uses, nor can they optimize the network for specific uses. By enabling users to use the Internet in the way that creates the most value for them, the broad version not only maximizes the value of the Internet for users, it also increases the Internet's overall value to society.

But the social value of architectures based on the broad version goes beyond that. The Internet has the potential to enhance individual freedom, provide a platform for better democratic participation, foster a more critical and self-reflective culture, and potentially improve human development everywhere.²³ The Internet's ability to realize this potential, however, is tightly linked to features—user choice, non-discrimination, non-optimization—that result from the application of the broad version of the end-to-end arguments. Though a network based on the broad version may reduce network providers' incentives to deploy more and better broadband networks to some degree, letting network providers deviate from the

broad version by increasing the controllability of the network considerably reduces the Internet's value to society. Given that there are other ways to foster broadband deployment that are not similarly harmful, I resolve this trade-off in favor of the social benefits resulting from the broad version.

Network providers' interests point in a different direction. They capture only a small part of the social value resulting from the broad version, and the fact that this value is uncertain and will be realized in the future leads them to further discount their private benefits associated with respecting the broad version. They have an incentive to increase the controllability of the network to make more profits. They also have an incentive to optimize the network for specific applications or find easy solutions for technical problems, making the network more opaque. Insofar as network providers control the evolution of the network's core, it is not surprising that the network is moving away from its original end-to-end design.

Given this gap between network providers' private interests and the public's interests, we face an important choice. Leaving the evolution of the network to network providers will significantly reduce the Internet's value to society. If no one intervenes, network providers' interests will drive networks further away from the broad version. With this dynamic, doing nothing is not enough to preserve the status quo, let alone to restore the full potential of the Internet.

If the Internet's value for society is to be preserved, policy makers will have to intervene. How to best do so without over-restricting the evolution of the network or locking in design choices that may not remain appropriate in the future is a difficult question. Law and architecture may interact in different ways. Some issues can be solved through regulation after technology has been deployed. For example, even if a network contains devices that let network providers control the applications and content on their networks, non-discrimination rules can forbid them from using these devices. In contrast, some issues can be solved only at the level of technology. For example, an optimization in favor of specific applications, such as the asymmetric bandwidth to and from most residential homes, can be overcome only through new technology; it cannot be undone by a law. In these cases, regulators need to shape the technology *before* it is deployed.

All this suggests that potential policy interventions will have to be more sophisticated than simply requiring network providers to adhere to the broad version of the end-to-end arguments, or to maintain the Internet's original architecture.

But what does all this mean for the future of the broad version as an engineering design principle? After all, though we may object to regulators

imposing specific design principles through regulation, network architects do have to choose design principles that help them realize their goals.

Design principles are tools to achieve certain ends: an architecture with a specific set and prioritization of characteristics. The broad version of the end-to-end arguments may not be the only design principle that can create an architecture that is evolvable or that realizes the social benefits that the broad version makes possible. We know, however, that the broad version has these effects. At the same time, the broad version provides much more flexibility for the evolution of the network's core than is often assumed. As the discussion of its potential interaction with security requirements in chapter 9 showed, the broad version is also able to react to other requirements that were not considered in the trade-offs underlying this design principle. Since the fundamental trade-off underlying the broad version is still appropriate (there are many more applications that have yet to be invented, which makes the long-term evolvability of the network more important than short-term performance optimizations), the broad version is at least a serious contender to be one of the design principles for the future Internet. Still, even if the broad version is chosen to shape the future Internet, its interaction with changed or new requirements will most likely result in an architecture that differs from the Internet's original architecture.

Though the focus of this book was on the effect of architecture on the economic system, the relationship between architecture and the economic system is not a one-way street. If architecture has economic implications, we can expect economic actors to begin exploiting the economic effect of architecture by engaging in strategic design—that is, attempting to shape architectures in their favor. The move by network providers toward next-generation networks and technologies such as the IP Multimedia Subsystem is an example of strategic design. From the perspective of network providers, the IP Multimedia Subsystem corrects perceived deficiencies of the Internet's original architecture (such as the lack of control and sophisticated charging capabilities) while maintaining the Internet's perceived benefits (e.g., the benefits of innovation by actors other than the network provider) (table C.1).²⁴

Traditionally, actors have tried to influence laws, norms, or economic constraints to improve their positions. Architecture has become another tool that actors can use to further their interests.

For example, Google is currently trying to recreate the conditions that characterized the economic environment for the development, production, and use of the original Internet for the mobile Internet (tables C.2

Table C.1

Differences between network architectures.

	Intelligent telephone network	IP multimedia subsystem	Original Internet
Potential innovators	Few	Many	Many
Control over network use	Network providers	Network providers	Users
Sophisticated charging capabilities	Yes	Yes	No

Table C.2

Differences between the traditional mobile Internet and Google's vision of the mobile Internet.

	Traditional mobile Internet	Google's vision of mobile Internet	Original Internet
Specificity of end hosts	Application specific	General	General
Control over end hosts	Network providers, handset makers	Application developers and users	Application developers and users
Control over network	Network providers	Users	Users

Table C.3

Differences in end hosts for the mobile Internet.

	General	Application-specific
Controlled by users	Android	
Controlled by network providers and handset makers	Apple iPhone	Traditional cell phone

and C.3). In the past, the economic environment in which the mobile Internet was developed, produced, and used differed considerably from the economic environment in which the original Internet was developed, produced, and used, both at the handsets and in the network.²⁵ In the original Internet, users control which applications are running on their end hosts. Standardized application programming interfaces that give developers access to the full capabilities of the end hosts are available at low cost. In contrast, mobile phones have been traditionally controlled by handset providers and carriers. Users usually cannot freely install applica-

tions on their cell phones in the same way they can on their personal computers. Application programming interfaces for handsets were proprietary, varied widely, and were costly to access, and often they provided only limited access to the phone's technical capabilities. Whereas users of the Internet can freely attach their computers to the Internet and can freely decide how they want to use the network, mobile carriers contractually or technically restrict which phones can be used on their networks and control the use of the network through contractual or technical means. As a result, in the mobile network, it has been difficult to develop new applications; network providers control the handsets and the use of the network. In contrast, networks based on the broad version of the end-to-end arguments make it easy to develop applications and enable users, who control their end hosts and the use of the network, to freely choose application and services.

Why is Google seeking these changes? If browsing on a cell phone becomes as convenient as browsing on a PC, Google will benefit from consumers' cell-phone searches just as it benefits from their PC searches.²⁶ To realize its goal, Google uses a mix of architectural, legal, and economic strategies.

At the core of Google's architectural strategy is Android, a new software platform and operating system for mobile phones that Google developed in a business coalition with handset makers, hardware vendors, and software vendors. Android is available to anyone for free and lets programmers create applications in the well-known Java programming language. Though carriers and handset makers can still cripple the functionality of a phone, Android is designed to give control of cell phones to users and application developers.

Some consequences of the broad version cannot be recreated by Android. Making it easier for developers to develop applications and letting users install them does not help users if carriers do not let the user's phone access the mobile network. Android also cannot protect against discrimination by the network. Google's legal strategy tries to solve these problems through regulation. Working with partners, Google has been lobbying heavily for regulations that would let people use any cell phone on a mobile network as long as it does not harm the network and would ban mobile network providers from interfering with the applications and content on the network. When the Federal Communications Commission decided to auction off spectrum in the 700-MHz band, Google asked for the rules just described as well as for rules that would require the winners of the spectrum auction to license the spectrum to other providers on a wholesale

basis in order to increase competition in the market for mobile carriers. (These latter regulations would have influenced the market structure in the market for mobile networks, which according to popular theory, would also reduce carriers' incentives to discriminate.) Google promised to bid for a portion of the 700-MHz spectrum in exchange for the imposition of its desired regulations.²⁷ When the FCC imposed two of the desired regulations on one block of spectrum in the 700 MHz auction, triggered by a certain reservation price, Google participated in the auction and bid the price up to the reservation price to make sure the regulations were triggered.²⁸

These architectural and legal strategies were complemented by a more traditional economic strategy. Google got involved in the market for broadband wireless Internet service by investing in Clearwire, a company that provides wireless broadband Internet services, alongside with cable companies and Intel. Through its influence on the strategies of one wireless broadband provider, Google may be able to cause mobile carriers to change their behavior.

The IP Multimedia Subsystem was originally developed for mobile networks. Thus, in a way, wireline network providers try to make the traditional wireline Internet more similar to the closed and controlled wireless cellular Internet, while Google tries to make the cellular Internet more like the traditional Internet.

Influencing architecture can be a powerful way to pursue one's interests. Corporate actors such as network providers or Google have recognized this. Those protecting the public interest need to do so, too.

Notes

Introduction

1. Miniwatts Marketing Group 2009. This number reflects the estimated number of Internet users as of June 30, 2009. (N.B.: In the United States, a billion is a thousand million, i.e., 10^9 .)

2. This argument was pioneered by the law professors Mark Lemley and Lawrence Lessig in an *ex parte* submission to the Federal Communications Commission in the AT&T-MediaOne merger proceeding (Lemley and Lessig 1999; see also Lemley and Lessig 2001). On the end-to-end arguments and network neutrality, see Lessig 2001; Lessig 2002; Wu and Lessig 2003; Wu 2003a; Lessig 2006; Lessig 2008b.

3. Network neutrality rules prevent network providers from interfering with the applications and content on their networks. Open access rules require the owners of broadband networks to allow independent Internet Service Providers to offer their services over these networks. Rules regarding network management in broadband network constrain network providers' ability to manage their networks. They may, for example, require network providers to use non-discriminatory mechanisms for managing their networks, if possible. For more on these debates, see chapter 6 below.

4. Since the 1990s, scholars of management strategy have explored the economic effect of modular and integrated architectures. See Langlois and Robertson 1992; Garud and Kumaraswamy 1993; Garud and Kumaraswamy 1995; Ulrich 1995; Sanchez 1995; Chesbrough and Teece 1996; Sanchez and Mahoney 1996; Baldwin and Clark 1997; Sanchez 1999; Baldwin and Clark 2000; Schilling 2000; Fixson and Sako 2001; Schilling and Steensma 2001; Christensen, Verlinden, and Westerman 2002; Langlois 2002; Sako 2002; Sturgeon 2002. The effect of the Internet's original architecture on innovation was first highlighted by Lemley and Lessig (1999). For a more detailed analysis, see Lessig 2001.

5. Researchers of business strategy have long recognized the strategic importance of architecture. See Ferguson and Morris 1993; Morris and Ferguson 1993; Gawer and Cusumano 2002; Baldwin and Clark 2006a; Woodard 2006; Woodard 2008.

6. More precisely, when talking about the original architecture of the Internet, I refer to the network architecture that was specified in the DARPA Internet Program protocol specifications for the Internet Protocol and the Transmission Control Protocol, RFC 791 (Postel 1981a) and RFC 793 (Postel 1981b). David Clark described this architecture in an important 1988 article on the design philosophy of the DARPA Internet protocols (D. D. Clark 1988).

7. See, e.g., Carpenter 1996, paragraph 2; Computer Science and Telecommunications Board and National Research Council 2001, pp. 36–38; Braden et al. 2000, p. 15; Moors 2002; Blumenthal and Clark 2001, p. 70; Clark et al. 2002, section 4.1.

8. Lemley and Lessig 2001, p. 928.

9. Solum and Chung 2004, p. 845.

10. Ibid., p. 844.

11. Sandvig 2002, p. 21.

12. Sandvig 2006, p. 235.

13. See, e.g., Yoo 2004, pp. 26, 41–46.

14. Saltzer, Reed, and Clark 1981. The 1981 paper was a conference paper. When referring to the original paper that first identified the end-to-end arguments, researchers usually cite the revised version (Saltzer, Reed, and Clark 1984) that appeared in the *ACM Transactions on Computer Systems* in 1984. I generally follow this convention too.

15. See, e.g., Reed, Saltzer, and Clark 1998.

16. A network that provides “Quality of Service” (QoS) offers different types of service to different data packets. For example, it may guarantee a minimum bandwidth or maximum delay, or it may give some traffic priority over others without giving absolute guarantees. See Peterson and Davie 2007, pp. 499–525.

17. In part III of the book, this thread focuses on how the broad version of the end-to-end arguments affects application-level innovation and neglects other aspects of the effect of the Internet’s architecture on innovation explored in that part. Readers interested in this thread should skim chapter 1 for an introduction to the framework that underlies the more detailed analysis covered by this thread.

Chapter 1

1. This observation is often summarized as “code is law” (Lessig 1999a, p. 6). The argument is developed in detail in Lessig 1999a. For a shorter exposition of the argument, see Lessig 1999b. Lessig builds on the work of a number of scholars; see Mitchell 1995, p. 111; Katsh 1996, pp. 335, 338; Johnson and Post 1996, pp. 1378–1387; Boyle 1997, p. 177; Reidenberg 1998, p. 553; Shapiro 1998, pp. 715–723.

2. The slogan is usually attributed to Mitchell Kapor, co-founder of the Electronic Frontier Foundation. See Barlow 1992, p. 25; Lessig 1999a, p. 242, note 7 to chapter 1.

3. This use of the word follows Lessig's terminology. In general, Lessig (1999b, p. 507) uses 'architecture' to refer to "the physical world as we find it, even if 'as we find it' is simply how it has already been made." In the context of the Internet, Lessig (*ibid.*, pp. 506, 509) uses 'architecture' and 'code' interchangeably to denote "the software and hardware that constitute cyberspace as it is."

4. This definition is equivalent to one often used in regard to software architecture: "The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them. 'Externally visible' properties refers to those assumptions other components can make of a component, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on." (Bass, Clements, and Kazman 1998, p. 23) This definition is widely quoted in the literature on software architecture—see van Vliet 2000, pp. 255–256; Bosch 2000, p. 11; Bengtsson et al. 2000, p. 3; Hofmeister, Nord, and Soni 2000, pp. 4, 6. For alternative definitions, see Bass, Clements, and Kazman 2003, pp. 23–24. The architecture of a complex software-intensive system usually consists of several structures that represent different aspects of the architecture. Each structure describes the system in terms of components, their externally visible properties, and the relationships among them, but each structure addresses different engineering concerns (e.g., the conceptual decomposition of the system, the design's concurrency and synchronization aspects or the mapping of the software onto the hardware elements). The representation of such a structure is usually called a *view* of the architecture. The terminology and the number of views differ across authors. This book's analysis focuses on the view of architecture that describes the decomposition of the system into subsystems and modules and the management of dependencies between them. Hofmeister, Nord, and Soni (2000, pp. 11–13, 97–124) call this view "the module view." Kruchten (1995, pp. 45–46) calls it "the development view." For more on the use of views in software architecture design, see Hofmeister, Nord, and Soni 2000, pp. 9–16; Bass, Clements, and Kazman 2003, pp. 35–42.

5. This definition does not prescribe a particular method of decomposition. Often, decomposition takes place on a functional basis, as in engineering design and traditional software design. In a functional decomposition, the overall functionality of the system is decomposed into a number of subfunctions, each of which solves part of the problem. These subfunctions, or subfunctions thereof, are then assigned to components. Thus, when functional decomposition is used, designing the architecture consists of allocating functionality to structure. The above definition also accommodates an object-oriented approach to design, which concentrates on the data on which the system will operate. As a result, the system is decomposed into collections of objects and relationships among them. (For a short overview of object-oriented analysis and design, see van Vliet 2000, 351 ff.)

6. In definitions of software architecture, this is often called the “physical view” (Kruchten 1995, pp. 46–47) or the “deployment view” (Bass, Clements, and Kazman 2003, pp. 38–39). In the terminology of Hofmeister, Nord, and Soni (2000, pp. 11–12), the mapping of functional components to the hardware is part of the “execution view”.
7. Ulrich and Eppinger 2000, pp. 182–183.
8. The hierarchical structure described in the text is a common feature of complex systems. See Simon 1969, pp. 86–88.
9. Christensen and Rosenbloom (1995, p. 238) describe this structure as a nested hierarchy: “In other words, products which at one level can be viewed as complex architected systems act as components in systems at a higher level. Viewed in these terms, a given system-of-use comprises a hierarchically nested set of constituent systems and components.”
10. See Bass, Clements, and Kazman 2003, p. 72.
11. For a detailed explanation of these and other qualities in the context of software architecture, see Bass, Clements, and Kazman 1998, pp. 75–88. For an overview of existing taxonomies of quality attributes in software engineering, see van Vliet 2000, pp. 110–119. Definitions of quality characteristics for software are also contained in ISO/IEC Standard 9126: Software Quality Characteristics and Metrics (ISO 1997), which is reproduced in part on pp. 117–119 of van Vliet 2000.
12. The following two paragraphs draw heavily on p. 78 of Bass, Clements, and Kazman 1998.
13. This is due to the overhead incurred in inter-component communication.
14. Because it is possible to counter good architecture with bad subsequent design or implementation, “a good architecture is necessary, but not sufficient, to ensure quality” (Bass, Clements, and Kazman 1998, p. 32). See also Hofmeister, Nord, and Soni 2000, p. 3.
15. On conflicts between quality attributes and the resulting need for trade-offs, see Bass, Clements, and Kazman 1998, pp. 127–129; Bosch 2000, p. 31. For an abstract overview of different ways to resolve conflicts between requirements or qualities, see Hofmeister, Nord, and Soni 2000, p. 6.
16. For a detailed analysis of this trade-off in the context of a client-server architecture, see Kazman et al. 1998, sections 4–10 .
17. Conceptually, this decision is not a part of architectural design, but belongs to the specification of requirements. Because some conflicts between quality requirements become apparent only during architectural design, there probably will be some iteration between the requirements specification and architectural design.

18. The exact nature of a design principle's architectural guidelines may differ. Whereas some design principles contain abstract rules for allocating functionality to components, others are mainly concerned with structural choices. For example, much research in software architecture focuses on identifying, analyzing, and documenting "architectural styles" (or "architectural patterns"), i.e., design principles that influence the structure of an architecture. A specific architectural style consists of a collection of component types, a description of the potential patterns of interaction among them, and a set of constraints on how these elements can be combined. The constraints may limit the behavior of components, restrict the kind of interaction allowed among the different types of components, or impose restrictions on the topological layout of the system. Architectural styles usually do not describe how system-specific functionality is allocated to component types. Other design principles may provide guidelines on how to allocate the functionality of the system under design to instances of the component types given by the architectural style. The seminal work on architectural styles (or architectural patterns) is Shaw and Garlan 1996. For encyclopedic coverage, see Buschmann et al. 1996 and Schmidt et al. 2000. The research on architectural patterns has been strongly influenced by the work of the architect Christopher Alexander. (See Alexander et al. 1977; Alexander 1978.) For an introduction to Alexander's work for software engineers, see Lea 1994.

19. How using the design principle affects the properties of the resulting system may be described in more or less detail. In the literature on software architecture, descriptions of a design principle usually include an informal description of the advantages and disadvantages of using it, an account of how it has been used historically, and qualitative reasoning to explain why its use results in a system with the specified properties. (For some examples concerning architectural styles, see Shaw and Garlan 1996, chapter 2. For a description of different architectural styles and the corresponding rules of thumb for choosing an architectural style, see chapter 5 of Bass, Clements, and Kazman 1998 or pp. 270–281 of van Vliet 2000.) There are efforts, however, to associate design principles with frameworks for more detailed qualitative or quantitative reasoning. For example, research on "attribute-based architectural styles" seeks to link the description of an architectural style with an analysis of that style with respect to a particular quality attribute such as modifiability or performance (see Klein and Kazman 1999). Research on "quality attribute design primitives" or "architectural tactics" seeks to describe architectural building blocks that are smaller and more fundamental than architectural styles and to combine them with a quality-attribute-specific analysis. On quality attribute design primitives, see Bass, Klein, and Bachmann 2000. On architectural tactics, see Bachmann, Bass, and Klein 2002 and chapter 5 of Bass, Clements, and Kazman 2003.

20. For case studies demonstrating this point, see Shaw and Garlan 1996, pp. 33–38; Bass, Clements, and Kazman 1998, pp. 117–121; van Vliet 2000, pp. 258–270. For additional examples, see Shaw and Garlan 1996, chapter 3.

21. See Bass, Clements, and Kazman 1998, pp. 113–117.

22. In practice, compliance with design principles may be very difficult to enforce. In major software projects, system architects can force designers to abide by the design principles by rejecting designs that violate the chosen design principles. In larger systems without a hierarchical relationship between system architects and component designers, system architects may be unable to enforce the design principles and are dependent on designers following them on a voluntary basis. In case of a system that is standardized in a standard-setting organization, the organization can reject standards solutions that violate the design principles, but this does not necessarily prevent someone from developing such solutions outside the standard-setting process. For example, though the Internet Engineering Task Force may reject certain technical solutions, that does not prevent individual vendors from incorporating these solutions into their products or network providers or users from deploying them. The emergence and deployment of devices for deep packet inspection is an example of this phenomenon.

23. The constraints described in the text are examples of different types of constraints that affect human behavior. Laws include constitutional and judicial rules, rules that directly regulate behavior, and economic rules such as the legal rules governing property rights (North 1990; Posner 1998b). Laws are an example of “formal constraints” (North 1990, chapter 6); other formal constraints include regulations, bylaws, and individual contracts. Social norms are informal rules that govern behavior within a community (Ellickson 1991; Posner 2000). Contrary to law, a norm is “neither promulgated by an official source, such as a court or legislature, nor enforced by the threat of legal sanctions, yet is regularly complied with” (Posner 1997, p. 365). Together with codes of conduct and conventions, social norms are a type of “informal constraint” (North 1990, chapter 7). The technical and natural environment includes the climate or natural resources in a given area (Diamond 1997), the current technical, scientific, economic, political, and cultural knowledge about the world (Nelson and Winter 1982; Rosenberg 1982; Freeman and Louca 2001; Perez 2002; Nelson 2005), the architectural layout of cities (Lessig 1998; Katyal 2002), and the digital environment established by the hardware and software of the Internet (Lessig 1999a). See, generally, North 1990 and Lessig 1998.

24. See North 1990; Hall and Soskice 2001.

25. The framework described in the text is used by researchers in economics (North 1990; Aoki 2001; Furubotn and Richter 2005; Greif 2006), political science (Hall and Taylor 1996; Thelen 1999; Peters 2005); law (Sunstein 1996a; Lessig 1998; Benkler 2006), and sociology (Powell and DiMaggio 1991; DiMaggio 1998; Scott 2000).

26. For an overview of the literature, see Lessig 1998.

27. See Posner 1996b.

28. For general analyses of the issues that may arise if law regulates other structures of regulation to reach policy goals, see Lessig 1998, pp. 686–691; Lessig 1999b, pp. 536–541. For an analysis of the negative consequences of various efforts to regulate technical aspects of the Internet to reach specific goals, see Solum and Chung 2004.

29. For these and other ways of influencing market constraints, see Viscusi, Harrington, and Vernon 2005, pp. 357–368.

30. Weakening the effect of a norm and providing incentives to break it is the first step toward changing a norm. See Sunstein 1996b, p. 930; Posner 1996a, p. 1729; Lessig 1995, pp. 1010–1012. For general discussions of different techniques of norm regulation, see Lessig 1995; Posner 1996a; Sunstein 1996b; Cooter 1998; Posner 1998a.

31. For general overviews of methods by which governments can influence technical systems in cyberspace, see Reidenberg 1998; Lessig 1999b; Kesan and Shah 2005. For critical reviews of specific efforts to regulate behavior by directly or indirectly regulating technical aspects of the Internet, see Zittrain 2003; Solum and Chung 2004. Similarly, governments can influence the physical environment in order to affect behavior in real space. For some interesting historical examples, see Lessig 1999a, pp. 91–92. Different ways in which government can influence the architecture of cities, neighborhoods, and individual buildings to reduce criminal activity and some of the associated problems are analyzed in Katyal 2002.

32. The architecture of a system can also influence other economic and non-economic activities, for example the use of the system, or political speech. See the discussion in chapter 9 below.

33. This is the standard definition of ‘innovation’ used in the literature. See Hall 1994, p. 2; Beije 1998, pp. 1–2.

34. See, e.g., Hall 1994, pp. 17–19.

35. This is the central assumption underlying economic theory: People react to changes in the costs and benefits associated with certain behavior in a way that is observable and predictable, and economic theory can be used to predict how people react to such changes. While different economic theories differ in their predictions as to how exactly people react to changes in incentives, they agree that, on average, an activity that becomes less costly is likely to be pursued to a greater extent and that an activity that becomes more costly will be reduced or eliminated. See Jolls, Sunstein, and Thaler 2000, p. 17.

36. North 1990, p. 5; Arrow 1974, p. 224; Thelen 1999, pp. 377–378.

37. Economists usually treat resources and cost structures as constraints. Since they are specific to the individual actor, I mention them here. See also Ostrom 2005, p. 828; Furubotn and Richter 2005, pp. 308–309.

38. See, generally, Ostrom 2005, pp. 828–829; Furubotn and Richter 2005, pp. 3, 308–309.

39. On the need to consider the joint effect of all constraints to determine the effect on an individual, see Lessig 1998, pp. 663–664; Lessig 1999b, pp. 509–510.

40. For an in-depth analysis of this issue from an economic perspective and pointers to the literature, see Landes and Posner 2003.

41. For a detailed discussion of this possibility, see chapter 5. This part of the framework also allows the integration of insights from sociology about the effect of existing social or economic relationships on economic behavior. See, e.g., the research on embeddedness (e.g., Granovetter 1985; Uzzi 1996) or on the importance of organizational fields for firms' organizational choices (e.g., DiMaggio and Powell 1983). On the relevance of embeddedness within organizations, cultures, or network connections for innovation, see Saxenian 1994; Castilla et al. 2000; Saxenian 2002; Powell et al. 2002; Powell and Grodal 2005; Porter, Bunker Whittington, and Powell 2005.

42. For a similar argument, see Lipsey, Carlaw, and Bekar 2005, p. 16.

43. Venture capitalists specialize in providing funding to high-risk, early-stage ventures. See Gompers and Lerner 2006. Normal banks are less well equipped than venture capitalists to identify promising investment opportunities and are less likely to finance new ventures even if they are promising. If the existing framework of constraints (or, in the terminology of the new institutional economics, the institutional framework) is sufficiently flexible, though, new forms of financing may arise in response to changes in the architecture. See Baldwin and Clark 2000, pp. 416–417. On the importance of an economic system's ability to adapt to new circumstances for the performance of an economy, see North 1990, pp. 81–83, 135–137.

44. For a theory along these lines, see Baldwin and Clark 2000. Based on the theory of complex adaptive systems and the theory of real options, they develop a theory of how designers' search for value drives architectural evolution in modular architectures, which in turn drives industry evolution; they use this theory to explain the evolution of the computer industry. I will discuss the relationship between architectures and real options in chapter 4.

45. More precisely, when talking about the original architecture of the Internet, I refer to the network architecture that was specified in the DARPA Internet Program protocol specifications for the Internet Protocol and the Transmission Control Protocol, RFC 791 (Postel 1981a) and RFC 793 (Postel 1981b). As I have already mentioned, David Clark described this architecture in his 1988 article on the design philosophy of the DARPA Internet protocols (D. D. Clark 1988).

46. In reality, this may happen if an architecture that has been in use in a specific country is changed while the other constraints stay the same.

47. Rational-choice theory is the theory that economists have traditionally used to model human behavior (see Kreps 1990, chapters 2–4). Though the details may differ, rational-choice theory usually assumes that human actors have well-defined interests (“preferences”) that can be described by individual utility functions, and that they seek to maximize their utility. Bounded rationality takes account of the fact that human actors may have limited knowledge and limited cognitive capabilities (Williamson 1975; Williamson 1985; Simon 1957). Behavioral economics takes account of the systematic deviations from rational-choice theory that have been found empirically (Kahneman and Tversky 2000; Camerer, Loewenstein, and Rabin 2004).

Chapter 2

1. System architects create the architecture of a system. Software development projects for larger systems often have dedicated “software architects.” For systems developed through standardization, the architecture is often developed as part of the standardization process; in this case, the system architects are those who work on defining the architecture within the standard-setting process.

2. To some readers, characterizing design principles as “rules that system architects must follow” may seem too strong. After all, if the system architects choose the design principles, they can also choose to deviate from them. Although this is correct, deviating from a design principle often compromises the ability of the final system to realize the qualities that the design principle fosters. Thus, though system architects can decide for themselves how closely they want to comply with the constraints imposed by the principle, this should be a deliberate decision that seriously considers the consequences. Seemingly minor deviations may add up and may severely compromise the system’s ability to meet its quality goals.

3. This analysis of modularity draws heavily on the ideas Carliss Baldwin and Kim Clark develop in their 2000 book *Design Rules*. For an application of Baldwin and Clark’s approach to software design, see Sullivan et al. 2001.

4. At first sight, this description looks different from the one usually provided in the computer science literature, where modularity entails decomposing the system into smaller components with well-defined interfaces. (See van Vliet 2000, p. 299.) Smaller components with well-defined interfaces, however, result from the second step of a modular design (described in the text) that addresses and resolves all remaining dependencies among components. Though the traditional definition does not contain any rules concerning the decomposition of the system, descriptions of modularity usually proceed to argue that effective or good modularity is reached by decomposing the system in a way that minimizes dependencies between modules, which is equivalent to the first step of designing a modular architecture as described in the text. See Pressman 1997, pp. 357–361; Sommerville 1996, p. 218. Thus, the definition used in the text is equivalent to the traditional definition of a

“good modular architectural design.” As described here, modularity as a design strategy is not restricted to particular approaches to design. For example, components with well-defined interfaces that realize the principle of information hiding also are characteristic of object-oriented systems (Sommerville 1996, p. 215). The definition of ‘modularity’ used in the text is often used to describe modularity in the context of physical products too (Sanchez and Mahoney 1996, pp. 65–66; Baldwin and Clark 2000, pp. 63–64). For a review of the literature on different definitions of the concept of modularity in the context of assembled hardware products, see Fixson 2001.

5. See Baldwin and Clark 2000, p. 63; Schilling 2000, p. 312; Ulrich and Eppinger 2000, p. 184.

6. See Baldwin and Clark 2000, pp. 63–64. Abstraction, information hiding, and the separation of concerns are fundamental principles of software design. On abstraction in software design, see Bass, Clements, and Kazman 1998, p. 125; van Vliet 2000, pp. 296–299. On separation of concerns, see Bass, Clements, and Kazman 1998, p. 124. The seminal paper on information hiding is Parnas 1972.

7. With respect to the following, see Baldwin and Clark 2000, pp. 72–76.

8. My interpretation of interfaces as the resolution of all remaining cross-component interdependencies follows Baldwin and Clark 2000, pp. 70–76. See also Sullivan et al. 2001, section 3.2. Sullivan et al. apply Baldwin and Clark’s approach to software design.

9. For example, a module’s interface describes the services provided by the module and how they can be invoked and specifies all functional and non-functional properties of the module that may be relevant for other modules using its services. This may include statements about the quality of service provided by the module or its performance, availability, or security. Access to the services of a module can occur only through its interfaces. As a result, each interaction between two modules is reflected in the interface of the module whose services are used. In present-day software architecture design, one also specifies the services the module needs from the system to perform its services as part of its interface definition. This part of the interface definition is usually called the *required interface*. In this case, each interaction is reflected in the corresponding interfaces of both modules: once in the “provided interface” of the module that provides the service, and once in the “required interface” of the module that uses the service (Hofmeister, Nord, and Soni 2000, p. 110; Bosch 2000, pp. 220–224). This fact is subsumed in the text for expository clarity. Changes to the provided or required part of a module’s interface have very different consequences: changing the interface provided by a module requires subsequent changes to all modules that use this interface. In contrast, if the internal implementation of a module uses existing interfaces in a new or different way, the “required interface” of this module has to change to reflect the change in dependencies, but this does not affect any module in the rest of the system.

10. See Ulrich 1995, p. 435; Baldwin and Clark 2000, pp. 52, 91; Sanchez 1995, pp. 145–147. For an analysis of the organizational implications of this fact, see chapter 5 below.
11. Changes to an architecture's visible information are usually very costly, as all modules that depended on the changed information have to be changed as well to enable the system to continue to operate. See chapter 4 below.
12. Baldwin and Clark 2000, p. 52.
13. See Baldwin and Clark 2000, pp. 68–89; Sullivan et al. 2001, sections 5 and 7.
14. Thus, a modular design introduces a hierarchical dependency into the system. Hidden design parameters depend on visible design parameters (Baldwin and Clark 2000, pp. 64–76).
15. For an example, see Sullivan et al. 2001, sections 5.1. and 5.3. For an empirical study of the relationship between the level of coupling and the evolvability of a system, see MacCormack, Rusnak, and Baldwin 2007.
16. Stevens, Myers, and Constantine 1999, p. 241. For factors balancing this effect, see *ibid.*, pp. 241–243.
17. See e.g., Bass, Clements, and Kazman 1998, pp. 32, 78, 79.
18. Similarly, for physical products, modularity may improve local performance (e.g., because of the ability to replace modules with better versions or due to the reduction in complexity in the design of individual modules), but may hurt global performance by preventing the joint optimization of the design of different modules necessary to optimize global performance with respect to heat or space by hiding aspects of a module's implementation, see Ulrich 1995, p. 432–434.
19. The terminology sometimes differs. See Sanchez and Mahoney 1996, p. 65 (“Methodology of Constrained Optimization”); Ulrich and Eppinger 2000, p. 184 (“Integral Architecture”); Baldwin and Clark 2000, pp. 49–52, 64–68 (“Inter-connected Design”). For descriptions of this approach in the context of physical products, see Baldwin and Clark 2000, pp. 49–52, 64–68; Sanchez 1999, p. 95; Ulrich and Eppinger 2000, p. 184.
20. See Pressman 1997, pp. 359–361; Sullivan et al. 2001, section 3.1. For sources of interdependencies between components of a physical system, see Baldwin and Clark 2000, pp. 34–39; Sanchez 1999, p. 93, note 3.
21. On physical products see Baldwin and Clark 2000, pp. 39–40. On software see Sullivan et al. 2001, section 3.1.
22. This process will usually require a considerable amount of “cycling” and iteration. See Baldwin and Clark 2000, pp. 49–52; Sanchez 1999, p. 95.
23. See Sanchez 1999, p. 95.

24. This strategy is known as “geometric nesting.” See Ulrich 1995, p. 433.
25. The use of layering is very common in modern software development. See Hofmeister, Nord, and Soni 2000, chapter 5. For discussions of layering in the context of architectural styles, see Shaw and Garlan 1996, p. 25; Bass, Clements, and Kazman 1998, pp. 100–101; Bosch 2000, pp. 119–120; van Vliet 2000, pp. 277–280; Sommerville 1996, pp. 232–233.
26. Hofmeister, Nord, and Soni 2000, p. 101.
27. In modern software architecture design, the description of a layer’s interface to lower layers is also part of a layer’s visible information. The interface to lower layers describes which lower-layer services or functionality the layer requires to correctly fulfill its responsibilities. While the interface to higher layers is called the *provided interface* (it specifies the services provided by the layer), the interface to lower layers is called the *required interface* (it specifies the services required by the layer). See Hofmeister, Nord, and Soni 2000, p. 113. The required interface is not important in the context of this book and is therefore neglected in the text for expositional clarity. (See note 9 above.)
28. See Bass, Clements, and Kazman 1998, p. 100; Hofmeister, Nord, and Soni 2000, p. 113; Bosch 2000, p. 119.
29. See Bass, Clements, and Kazman 1998, p. 100; Bosch 2000, p. 119.
30. See Bass, Clements, and Kazman 1998, p. 83. Following Clark (1997), a portability layer is also called a *spanning layer*. See, e.g., Messerschmitt 2000, pp. 185–186; Kavassalis and Lehr 2000, pp. 209–211.
31. See Tanenbaum 2008, pp. 4–6, 63–64.
32. In early operating systems, applications were sometimes able to access hardware functions directly. For example, in MS-DOS, application programs were able access hardware-specific device drivers to write directly to the hardware. See Silberschatz, Galvin, and Gagne 2005, p. 59.
33. See Shaw and Garlan 1996, p. 25; Hofmeister, Nord, and Soni 2000, p. 102. In the context of networking, see Comer 2000, p. 178; Peterson and Davie 2007, p. 20.
34. This problem is often mentioned in discussions of layering in the context of networking. See Comer 2000, p. 192; Kurose and Ross 2008, p. 48. It may be overcome by offering additional, potentially optional interfaces that selectively expose additional lower-layer information; see Eggert and Eddy 2006. See also Coulson et al. 1999; Herbert 1990.
35. See Bosch 2000, p. 121; Klein and Kazman 1999, p. 63. In the context of networking, see Peterson and Davie 2007, p. 37. See also Clark and Tennenhouse 1990.

36. For example, by associating processes (in some operating systems, a process is called a *thread*) with events instead of with layers, the amount of time-consuming context switches between processes may be reduced. A context switch occurs when the operating system “stops one process from executing on the CPU and starts up another one” (Peterson and Davie 2007, p. 37). This technique is often used in implementations of layered network protocols. Instead of implementing one process per protocol, protocol implementors associate processes with the messages that pass through the layers. (See *ibid.*, pp. 37–39.) It is also possible to increase performance by reducing the amount of copying when passing messages between layers. (See *ibid.*, pp. 39–40.) For an in-depth treatment of network architecture and design in the context of high-speed networking, see Sterbenz and Touch 2001.

37. Today, the use of layered network protocols is the generally accepted approach to network architecture. For discussions of layering and protocols in the context of network architecture, see Peterson and Davie 2007, pp. 20–25; Tanenbaum 2003, pp. 26–37; Comer 2000, chapter 11; Kurose and Ross 2008, pp. 45–51. For an alternative proposal that seeks to achieve modularity through a non-layered protocol paradigm called *role-based architecture*, see Braden, Faber, and Handley 2003. The proposal is intended to address the problems of layer violations and unintended feature interactions between layers that plague the current architecture of the Internet.

38. With respect to the following, compare Peterson and Davie 2007, p. 9; Tanenbaum 2003, p. 19; Kurose and Ross 2008, pp. 9–11; Blumenthal and Clark 2001, pp. 71–72, 80–81.

39. This description glosses over the fact that in some network architectures, the end hosts may not only use, but also offer network services. In the Internet, the computers on the network (or end hosts) also participate in the operation of the network through the protocols at the Internet layer and below. Thus, one may say that in the Internet, computers on the network form the network, too. A more precise description of the Internet would focus on layers: the layers up to the Internet layer are “in” the network, while the layers above the Internet layer are “on” the network. See, e.g., Comer 2000, p. 186; Sterbenz and Touch 2001, p. 350.

40. Throughout this book, “the core of the network” will be used to denote the set of computers “in the network,” or, in the case of the Internet, the lower layers up to, and including, the Internet layer. (See preceding note.) For a similar use, see Blumenthal and Clark 2001, pp. 71–72. Sometimes (but not in this book), “core network” is used to denote the part of a hierarchical telecommunications network that provides the highest level of aggregation, such as the backbone network, as opposed to the intermediate part of the entire network, the backhaul, that connects the core network with the access networks or edge networks.

41. See Blumenthal and Clark 2001, pp. 80–81. The functional, topological, and administrative views of the meaning of “in the network” are explicitly distinguished

by Sterbenz and Touch (2001, p. 350). According to Sterbenz and Touch (*ibid.*), “the topological view has to do with whether or not functions and resources are collocated with network nodes or located with subscriber end systems.”

42. The networking literature uses the word ‘protocol’ in different ways. Following the terminology used by Peterson and Davie (2007, pp. 20–25), this book uses ‘protocol’ to denote the collection of abstract entities at a certain layer that are distributed among different computers and cooperate with each other provide a particular service to the layer above. Others use ‘protocol’ more narrowly, to denote the rules governing the interaction among the communicating entities on different computers (see e.g., Tanenbaum 2003, pp. 36–37; Kurose and Ross 2008, pp. 8–9). Thus, in the terminology of this book, they use ‘protocol’ to denote a protocol’s horizontal peer interface, as defined below.

43. Protocol peers are the instances of a protocol on different computers that communicate with each other. They are not the same as peer-to-peer applications. Peer-to-peer applications are applications in which the computers running the application act as both client and server. In contrast, in client-server applications, a computer running the application acts either as a client or as a server.

44. On the different uses of ‘protocol’, see note 42 above.

45. When protocols are defined in the Internet Engineering Task Force, only the peer interface, that is the form and meaning of the messages exchanged between the communicating protocol entities, and the services the protocol provides to higher layers are specified. The specification of the exact interface through which higher-layer protocols can invoke the services is left to those who implement the protocol. On IP, see Postel 1981a, section 3.3; Comer 2000, p. 413. On TCP, see Postel 1981b, section 1.4; Comer 2000, p. 216.

46. In the terminology of network engineering, these messages that protocol peers at a particular layer (layer n) send to each other are called *layer n -protocol data units*. See Sterbenz and Touch 2001, p. 43.

47. In the terminology of network engineering, the part of a message in layer $n - 1$ that consists of data to be delivered to layer n (that is, that consists of either the complete layer n -message or parts of it), is called the *layer $n - 1$ payload*. See Sterbenz and Touch 2001, p. 43.

48. In the terminology of network engineering, the part of a message in layer $n - 1$ that tells the receiving-layer $n - 1$ peer what to do is called a *header* or a *trailer*, depending on whether it is inserted before or after the payload. See Sterbenz and Touch 2001, p. 43. Thus, the layer $n - 1$ protocol data unit is created by adding a header or a trailer to the layer $n - 1$ payload—that is, to the layer n protocol data unit, or parts of it. Because the layer n protocol data unit is encapsulated in the layer $n - 1$ protocol data unit created by the layer n protocol, the process of protocol data unit creation is also called *encapsulation*. See Peterson and Davie 2007, pp. 24–25.

49. Comer 2000, pp. 187–189; Kruse 1999, abstract and section 2; Sterbenz and Touch 2001, pp. 42–43.

50. Some researchers seem to imply that the layering principle prohibits the modification of higher-layer messages by lower-layer protocols. (See Kruse 1999, section 2.) This statement, however, is not accurate. In order to perform their lower-layer services, lower layers may temporarily modify the messages passed to them by a higher layer. For example, lower layers may split or compress the messages passed to them by a higher layer in order to perform their services. (See Peterson and Davie 2007, pp. 24–25.) The layering principle requires that all these modifications are completely reversed before the message is delivered to the higher-layer protocol at the receiving computer. For example, the Internet Protocol may split a data packet passed to it by a transport-layer protocol and encapsulate it into several datagrams. The Internet Protocol implementation at the receiver, however, will completely reverse this split and deliver the complete data packet to the transport-layer protocol at the receiver. (See Peterson and Davie 2007, pp. 239–242.) Thus, it seems more accurate to say that the layering principle prohibits lower-layer protocols from permanently modifying higher-layer messages.

51. See Peterson and Davie 2007, p. 22; Tanenbaum 2003, pp. 26–30; Sterbenz and Touch 2001, p. 44; Kurose and Ross 2008, pp. 47–51.

52. See Comer 2000, p. 187; Tanenbaum 2003, p. 30.

53. Tanenbaum 2003, p. 448; Peterson and Davie 2007, pp. 24–25; Kruse 1999, abstract and section 2.

54. The applicability of the end-to-end arguments is not restricted to communication networks. In fact, according to Reed, Saltzer, and Clark (1998, p. 70) the end-to-end arguments originally arose from the authors' research on secure operating system kernels and on end-to-end transport protocols in local area networks and the Internet. For additional examples of end-to-end arguments in the design of other systems, see Saltzer, Reed, and Clark 1984; Reed, Saltzer, and Clark 1998.

55. The terms “end hosts” and “core of the network” and the differences between higher-layer and lower-layer protocols were explained in the section of this chapter titled “Modularity and Layering in Network Architectures.”

56. With respect to the International Standards Organization's Open Systems Interconnection reference model (the transport layer and higher layers are typically implemented on end hosts, not on the intermediate switches or routers), see Peterson and Davie 2007, p. 28; Tanenbaum 2003, p. 40; Sterbenz and Touch 2001, pp. 41–42. With respect to the architecture of the Internet (end hosts implement all layers, while IP routers typically implement only lower layers, up to and including the Internet layer), see pp. 51–52 of Kurose and Ross 2008. In practice, routers may implement higher layers to terminate routing protocols such as BGP or management protocols.

57. Saltzer, Reed, and Clark 1981. The 1981 paper was a conference paper. When referring to the original paper that first identified and described the end-to-end arguments, researchers usually refer to the revised version (Saltzer, Reed, and Clark 1984) that appeared in *ACM Transactions on Computer Systems* in 1984. See, e.g., Moors 2002, p. 1214. I generally follow this convention too. For a discourse-oriented analysis of the end-to-end arguments and their subsequent role in political, social, and legal debates, see Gillespie 2006.

58. See Peterson and Davie 2007, pp. 52–53, 387.

59. See Carpenter 1996, section 2; Computer Science and Telecommunications Board and National Research Council 2001, pp. 36–38; Braden et al. 2000, p. 15; Moors 2002; Blumenthal and Clark 2001, p. 71; Clark et al. 2005, section VI.A.

60. Reed, Saltzer, and Clark 1998.

61. Blumenthal and Clark 2001; Clark and Blumenthal 2007.

62. Saltzer 1999; Reed 2000.

63. I first described the differences between the two versions in chapter 6 of van Schewick 2004.

64. Peterson and Davie (2007, p. 387) use a similar phrase, rephrasing the definition given in the original paper (Saltzer, Reed, and Clark 1984, p. 278): “The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)”

65. Reed, Saltzer, and Clark 1998, p. 69 (emphasis added).

66. Reed, Saltzer, and Clark (1998, p. 69), after referring to the 1984 paper, simply introduce the “new” second version quoted above as “the end-to-end principle.”

67. For two recent exceptions, see Moors 2002 and Kempf and Austein 2004. Moors notes that the original paper emphasized an end-to-end argument “relating to correctness of function,” referring to the version called *the narrow version* in the text. In addition to this version, Moors identifies a number of additional end-to-end arguments in the two end-to-end papers by treating each line of reasoning against the low-level implementation of functions as a separate “end-to-end argument.” As a result of this increase in the number of end-to-end arguments, however, the notion of “end-to-end arguments” becomes arbitrary and loses any prescriptive value. This does not seem desirable, given that it is possible to identify two distinct design principles that provide clear guidelines for design in their respective areas of applicability and that are supported by the various arguments presented in the two end-

to-end papers. According to the interpretation advanced in the text, the different arguments against the implementation of application-specific functionality in lower layers of the system are relevant in two ways: first, in the context of the narrow version, they need to be considered in order to decide whether an additional, necessarily incomplete low-level implementation of end-to-end functions is justified. This decision is the trade-off mentioned in the original paper. See the discussion of the narrow version below. Second, in the context of the broad version, these arguments form the justification of the design rule established by the broad version. See the discussion of the broad version below. Kempf and Austein (2004, section 1) also note changes in the interpretation of the end-to-end principle: “The end-to-end principle was originally articulated as a question of where best not to put functions in a communication system. Yet, in the ensuing years, it has evolved to address concerns of maintaining openness, increasing reliability and robustness, and preserving the properties of user choice and ease of new service development as discussed by [Blumenthal and Clark (2001)]; concerns that were not part of the original articulation of the end-to-end principle.” According to Kempf and Austein (2004, section 2.2), “the original articulation of the end-to-end principle—that knowledge and assistance of the end point is essential and that omitting such knowledge and implementing a function in the network without such knowledge and assistance is not possible—took a while to percolate through the engineering community, and had evolved by this point to a broad architectural statement about what belongs in the network and what doesn’t.” This observation is similar to the distinction between the narrow and the broad version made by this book. Contrary to this book, however, Kempf and Austein associate the end-to-end principle as a broad architectural statement with the question of “what kind of state is maintained where”—a point of view that is criticized in chapter 3 below. Finally, although Kempf and Austein emphasize the continuing importance of the end-to-end principle for the architecture of the Internet, it is difficult to see to which of the different interpretations of the end-to-end principle this statement is meant to apply. Thus, they fail to provide clear guidelines for the application of the end-to-end principle that can be used in network design.

68. See Peterson and Davie 2007, p. 387.

69. The perception of the end-to-end arguments in policy debates has been deeply influenced by Mark Lemley and Lawrence Lessig’s description of “the end-to-end principle” in their 1999 *ex parte* letter to the FCC (Lemley and Lessig 1999, sections 16–17): “16. The end-to-end principle organizes the placement of functions within a network. It counsels that that [sic] ‘intelligence’ in a network be located at the top of a layered system—at its ‘ends’, where users put information and applications onto the network—and that the communications protocols themselves (the “pipes” through which information flows) be as simple and general as possible. 17. One consequence of this design is a principle of nondiscrimination among applications. Lower level network layers should provide a broad range of resources that are not

particular to or optimized for any single application—even if a more efficient design for at least some applications is thereby sacrificed.” (See also Lemley and Lessig 2000, pp. 8–9.) As will become apparent below, this description refers to the broad version of the end-to-end arguments. For similar descriptions, see Lessig 2001, pp. 34–37 (“End-to-end says to keep intelligence in a network at the ends, or in the applications, leaving the network itself to be relatively simple”); Yoo 2002, p. 270; Solum and Chung 2004, p. 844. For descriptions of the broad version of the end-to-end arguments in the context of descriptions of the Internet’s architecture, see Computer Science and Telecommunications Board and National Research Council 2001, p. 36 (“Aimed at simplicity and flexibility, this argument [i.e., the end-to-end argument] says that the network should provide a very basic level of service—data transport—and that the intelligence—the information processing needed to provide applications—should be located in or close to the devices attached to the edge of the network”), Carpenter 1996, section 2.1 (“However, in very general terms, the community believes that the goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end [*sic*] rather than hidden in the network”), and Carpenter 1996, section 2.3 (“The network’s job is to transmit datagrams as efficiently and flexibly as possible. Everything else should be done at the fringes[,] in the context of the discussion of “the end-to-end argument”).

70. See Carpenter 1996, section 2.3.

71. Kempf and Austein (2004) advance this interpretation. See note 67 above.

72. The terminology does not reflect the differences in the content of the rules.

73. A complete comparison of the two versions can be found in the subsection of this chapter titled “Comparison of the Two Versions.”

74. In the paper, “communication system” is specified as follows (Saltzer, Reed, and Clark 1984, p. 278): “In a system that includes communications, one usually draws a modular boundary around the communication subsystem and defines a firm interface between it and the rest of the system. When doing so, it becomes apparent that there is a list of functions each of which might be implemented in any of several ways: by the communication subsystem, by its client, as a joint venture, or perhaps redundantly, each doing its own version.”

75. *Ibid.*, p.278.

76. Peterson and Davie (2007, p. 387) use similar phrasing.

77. The division of functionality between the layers has implications for the division of functionality between end hosts and the core of the network. The choice between layers, however, is not always equivalent to the choice between end systems and core of the network. See figure 2.6.

78. Although texts describing the narrow version agree that it only applies to certain functions (see Saltzer, Reed, and Clark 1984, p. 278; Moors 2002, p. 1214), the

literature does not contain a general description of the type of functions and of the circumstances under which the argument applies. Instead, texts list one or more specific functions to which the argument applies. For example, according to Saltzer, Reed, and Clark (1984), the (narrow) version applies to error control, “encryption, duplicate message detection, message sequencing, guaranteed message delivery, detecting host crashes and delivery receipts” (p. 278). Kempf and Austein (2004, section 2.1) and Peterson and Davie (2007, p. 387) also list specific functions to which the narrow version applies. A more general characterization of functions to which the argument applies seems desirable to enable designers to determine the applicability of the argument in their specific case. Building on the examples highlighted by the literature, the following treatment aims at providing such a general description.

79. Sterbenz and Touch (2001, pp. 39–41, 346–348) emphasize this case in their treatment of the end-to-end arguments.

80. Sterbenz and Touch 2001, pp. 39–41, 346–347; Moors 2002, p. 1214. For an empirical study confirming the introduction of errors by faulty hardware or software in routers or end hosts, see Stone and Partridge 2000.

81. Sterbenz and Touch 2001, pp. 39–41, 346–347. Even with end-to-end error control, data integrity cannot be absolutely guaranteed. This is due to the probabilistic nature of integrity checks. See Stone and Partridge 2000, section 2; Moors 2002, p. 1215.

82. Sterbenz and Touch 2001, pp. 39–41, 346; Moors 2002, p. 1215.

83. This case generalizes the example of reliable file transfer as well as other examples provided in the original end-to-end paper: as Saltzer, Reed, and Clark (1984, p. 280) note, even if the communication system can guarantee reliable data transmission within its boundaries, it cannot protect the application against threats to data integrity that occur outside the boundaries of the communication system. A similar reasoning about the possibility of threats to the function outside the boundaries of the communication system underlies the argument with respect to delivery guarantees (p. 282), secure transmission of data (pp. 282–283), or suppression of duplicate messages (p. 283).

84. *Ibid.*, pp. 278–280.

85. *Ibid.*, pp. 282–283.

86. This choice is also related to trust. See Moors 2002, pp. 1215, 1218–1219; Clark and Blumenthal 2007, pp. 2–6.

87. For example, according to the original paper, a file-transfer application must implement end-to-end error control itself in order to eliminate all threats to data integrity that may occur after the data leaves the communication system. For example, data may get corrupted when it is stored on the disk (Saltzer, Reed, and

Clark 1984, p. 280; Border et al. 2001, section 4.1.3). In spite of this insight, most applications today do not implement such functionality, but use the reliable-data-transfer functionality provided by transport-layer protocols such as TCP. On error control in higher layers, see box 3.2 below.

88. Saltzer, Reed, and Clark 1984, pp. 280–282.

89. Sometimes, the additional lower-layer implementation may reduce the performance of the system. For example, encrypting data at a lower layer that has already been encrypted at a higher layer may reduce the strength of the end-to-end encryption (Sterbenz and Touch 2001, pp. 423–424).

90. For a more detailed overview of the positive and negative effect of local implementations on performance, see Moors 2002, pp. 1216–1217.

91. Saltzer, Reed, and Clark 1984, p. 281. See also Sterbenz and Touch 2001, pp. 347–348.

92. See Saltzer, Reed, and Clark 1984, p. 281; Moors 2002, pp. 1216–1217. See also Wischik 2007, pp. 8–9.

93. For a discussion of these examples, see Sterbenz and Touch 2001, pp. 347–348.

94. Saltzer, Reed, and Clark 1984, pp. 281–282.

95. *Ibid.*, p. 281.

96. See, e.g., the abstract of the original paper (Saltzer, Reed, and Clark 1984, p. 277): “This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument, suggests that functions placed at the lower levels of a system may be redundant or of little value when compared with the cost of providing them at that low level.”

97. See, e.g., the quotations from Reed, Saltzer, and Clark 1998 and Blumenthal and Clark 2001 in the text. See also Saltzer 1999 (“The end-to-end argument says ‘don’t force any service, feature, or restriction on the customer; his application knows best what features it needs, and whether or not to provide those features itself.’”); Reed 2000 (“In that paper [meaning Saltzer, Reed, and Clark 1984] we argued that many functions can only be completely implemented at the end points of the network, so any attempt to build features in the network to support particular applications must be viewed as a trade-off. Those applications that don’t need a particular feature will have unnecessary costs imposed on them to support the other applications that benefit. We argued that building in such functions is rarely necessary, and that systems designers should avoid building any more than the essential and common functions into the network.”); Clark et al. (2005), p. 471 (“One of the most respected and cited of the Internet design principles is the end-to-end arguments which state that mechanism should not be placed in the network if it can be placed at the end

node, and that the core of the network should provide a general service, not one that is tailored to a specific application [Saltzer, Reed, and Clark 1984].")

98. Blumenthal and Clark 2001, p. 71.

99. Reed, Saltzer, and Clark 1998, p. 69.

100. *Ibid.*, p. 70.

101. Blumenthal and Clark 2001, p. 80 (citing personal communication with Jerome Saltzer).

102. Reed 2002.

103. Reed, Saltzer, and Clark 1998, p. 70; Reed 2000.

104. Saltzer, Reed, and Clark 1984, pp. 281, 284–285; Reed, Saltzer, and Clark 1998, p. 70; Reed 2000; Moors 2002, p. 1217.

105. Reed, Saltzer, and Clark 1998, p. 70; Blumenthal and Clark 2001, p. 71; Reed 2000; Moors 2002, pp. 1217–1218; Clark et al. 2005, p. 472.

106. Though humans can hear a broader frequency range, this is the range which they need so that they can understand speech clearly.

107. See Moors 2002, p. 1217, citing Lucky 1997.

108. Apart from asymmetric bandwidth, dynamic IP addresses, network-address translators (NATs) and firewalls are examples of this trend. See Minar and Hedlund 2001, pp. 13–15. On NATs and firewalls, see my concluding chapter.

109. See Kurose and Ross 2008, p. 84.

110. See *ibid.*, pp. 14–15.

111. The problems resulting from asymmetric bandwidth were clearly described in several reports by the Computer Science and Telecommunications Board and National Research Council (1994, p. 87; 1996, pp. 52, 71–77, 89–93).

112. The text assumes that the network only offers a single service to all applications, so that all applications have to use that service. It may be possible to offer different functions in the network among which applications can choose. Such a solution increases the complexity of the network and makes it more difficult to predict how the network will behave. This lower transparency constitutes a cost that is borne by all applications. Whether such a solution complies with the end-to-end arguments, has to be decided on a case-by-case basis. See Reed, Saltzer, and Clark 1998 (commenting on active networking and end-to-end arguments). A current example of a network service that is only provided to transport-layer connections that explicitly request it is Explicit Congestion Notification (ECN), where routers signal congestion to transport-layer connections by setting a code point in the IP packet, see Ramakrishnan, Floyd, and Black 2001.

113. Saltzer, Reed, and Clark 1984, pp. 281, 286–287; Reed, Saltzer, and Clark 1998, pp. 69–70; Moors 2002, p. 1217.

114. Saltzer, Reed, and Clark 1984, pp. 286–287.

115. *Ibid.*, p. 281; Moors 2002, p. 1217.

116. See Peterson and Davie 2007, p. 558. The source of this example is Moors 2002, p. 1217.

117. Reed, Saltzer, and Clark 1998, p. 70.

118. Not every application of the broad version automatically results in decentralized control by users (instead of centralized control by a few network providers), though; it only results in decentralized control if end users control the end hosts. Thus, control over a network can be centralized even if the network is designed according to the broad version, if a central entity controls the end hosts. For example, an enterprise network based on the TCP/IP protocol suite where applications can only be installed by system administrators is based on the broad version, but centrally controlled.

119. Blumenthal and Clark 2001, p. 71; Clark et al. 2005, p. 472.

120. Blumenthal and Clark 2001, p. 71; Clark et al. 2005, p. 472. However, a network provider may prefer to provide functions in the network, where he controls their execution, instead of relying on users or third parties to execute them correctly. See Abbate 1999, pp. 156–161.

121. Blumenthal and Clark 2001, p. 71; Clark et al. 2005, p. 472; Computer Science and Telecommunications Board and National Research Council 2001, p. 37.

122. This is particularly common in contributions to the policy debate. See Wu 2003a, p. 146: “For these reasons, Internet Darwinians argue that their innovation theory is embodied in the ‘end-to-end’ design argument, which in essence suggests that networks should be neutral as among applications. [Saltzer, Reed, and Clark 1984]”

123. See Lemley and Lessig 1999: “16. The end-to-end principle organizes the placement of functions within a network. It counsels that that [*sic*] “intelligence” in a network be located at the top of a layered system—at its “ends,” where users put information and applications onto the network—and that the communications protocols themselves (the “pipes” through which information flows) be as simple and general as possible. 17. One consequence of this design is a principle of non-discrimination among applications.” (p. 8 in Lemley and Lessig 2000)

124. Blumenthal and Clark 2001, pp. 82–83.

125. In this context, “message” denotes the data that the higher-layer protocol wants to pass on to its peer.

126. For similar observations, see section 2 of Kruse 1999 and p. 846 of Solum and Chung 2004.

127. See the section of this chapter titled “Modularity and Layering in Network Architectures.” Such mechanisms would introduce a dependence on higher layers. That lower layers must not depend on higher layers, however, is the central requirement for a layered architecture. If it is violated, changes at higher layers may require subsequent changes at lower layers, if the lower layer is to function correctly. This is exactly what the layering principle wants to prevent.

128. This consequence of an end-to-end design was first identified by Mark Lemley and Lawrence Lessig in their 1999 *ex parte* letter to the FCC (Lemley and Lessig 1999, reprinted as Lemley and Lessig 2000). But because the end-to-end arguments are just a design principle and network providers are not required to comply with them, nothing prevents the network providers from deploying devices in the network that enable them to control the traffic on their network, even if these devices violate the broad version.

129. For an overview of the debate and pointers to the literature, see chapter 6.

130. For a representative example of this view with respect to open access regulation, see Lemley and Lessig 1999, reprinted as Lemley and Lessig 2000. With respect to network neutrality rules, see Wu and Lessig 2003; Wu 2003a.

131. For two representative examples of this view, see Speta 2000a,b.

132. The two arguments are mutually exclusive. If network owners do not have an incentive to discriminate against independent applications anyway, the imposition of the network neutrality regime that prevents such discrimination will not reduce their profits. If it does not reduce their profits, however, it cannot reduce their incentives to invest in upgrades of their network infrastructure in the future.

133. For a representative example of this view, see Thierer 2004.

134. See Odlyzko 1998, pp. 40–41; Blumenthal and Clark 2001, pp. 74, 96; Clark et al. 2005, p. 467. As Jonathan Zittrain has pointed out, this makes it more difficult to adequately protect PCs from viruses, spyware, and other malware. The resulting security risk may be exploited in a global security attack which harms Internet users all over the world. In the aftermath of such an attack, users may call for countermeasures which may result in a lock-down of PCs, making them unable to run applications that have not been certified by a gatekeeper before. Such a development would considerably reduce the ability of users and application innovators to experiment with new applications. For a detailed description of this argument and possible solutions, see Zittrain 2008. For an analysis of the effect of the broad version on security, see the subsection of chapter 9 titled “Social Costs.”

135. For a similar argument in a slightly different context, see p. 467 of Clark et al. 2005 and section 4.1.1 of Kempf and Austein 2004. A widespread move toward

“software as a service” and appliances, which enables software providers and appliance producers to control their product’s functionality and even change it retroactively, may reduce the ability of users and application developers to experiment with new applications. For an analysis of the problem, see Zittrain 2008, chapter 5. For potential solutions, see Zittrain 2008, part III.

136. See Blumenthal and Clark 2001, pp. 77–78, 86; St. Johns and Huston 2003, section 3; Clark et al. 2005, p. 472. For an overview of principles of traffic engineering in the Internet today, see Awduche et al. 2002.

137. See Peterson and Davie 2007, p. 633; Blumenthal and Clark 2001, p. 80; St. Johns and Huston 2003, section 4.

138. For example, the IP protocol header contains a “protocol” field that indicates which higher-layer protocol at the destination host to deliver the data to. In other words, the “protocol” field indicates the higher-layer protocol that invoked the services of IP at the originating host. Thus, if an application does not use a transport protocol but invokes IP directly, the IP header will carry information about the application. This field is used solely for de-multiplexing at the destination host and does not influence the treatment of datagrams within the network. The numbering of protocols used in the protocol field is standardized across the entire Internet. The relevant numbers are contained in an online database located at www.iana.org. See Tanenbaum 2003, p. 435; Peterson and Davie 2007, p. 239. For an overview of the use of port numbers in TCP and UDP, see Comer 2000, pp. 197–206, 216–218.

139. See Blumenthal and Clark 2001, pp. 77–80, 86. The Internet Architecture Board has recently commented on this practice and highlighted some of the long-term problems associated with mechanisms that are not restricted to gathering information, but are intended to filter or prevent the transmission of traffic based on this information (St. Johns and Huston 2003).

140. Kruse 1999, section 2.

141. The proliferation of such devices creates huge problems in the current Internet. For example, such devices may affect the end-to-end integrity of data, may prevent the use of end-to-end IPsec or require the maintenance of state in the device, violating the principle of fate-sharing and introducing single points of failure. Most important in the context of this book, these devices may prevent or impede the deployment of new applications. The analysis of these problems is beyond the scope of this book. For a short overview of the problems for innovation, see the concluding chapter of this book. The Internet Engineering Task Force has analyzed these issues in a number of requests for comments. See Kaat 2000; Carpenter 2000; Carpenter and Brim 2002. Legal scholars have analyzed the policy implications of requiring the use of packet-filtering devices for regulatory purposes. See Solum and Chung 2004; Zittrain 2003.

142. For example, there may be instances when the cost reductions resulting from a low-level implementation are so large and the costs to applications with different

needs are so small, that an implementation of application-specific functionality in the network may be justified. In their second joint paper, Reed, Saltzer, and Clark mention the provision of network-level multicast as an example of such reasoning (1998, p. 69, note 1).

143. For a more detailed version of this argument in a slightly different context, see Solum and Chung 2004, pp. 854–865.

144. Braden et al. 2000, pp. 6–9.

145. See the discussion in my concluding chapter.

146. This trade-off is particularly apparent in Reed, Saltzer, and Clark 1998 (p. 70), in Blumenthal and Clark 2001 (pp. 71–72), and in Reed 2000.

147. Yoo 2004, pp. 40–43. The conflict results from the different ways in which the narrow version and the broad version approach partial implementations of application-specific functionality in the network. See the discussion in the subsection of this chapter titled “Comparison of the Two Versions.”

Chapter 3

1. See Comer 2000, pp. 53–55.

2. See Comer 2000, p. 63.

3. Kurose and Ross 2008, pp. 526–544; Peterson and Davie 2007, pp. 137–143.

4. Kurose and Ross 2008, pp. 469–480; Peterson and Davie 2007, pp. 116–124.

5. See Peterson and Davie 2007, p. 234.

6. The global Internet is denoted by ‘Internet’ (capitalized); ‘internet’ (lower case) usually refers to an arbitrary collection of networks interconnected to provide a universal communication service. See Comer 2000, p. 1.

7. See Peterson and Davie 2007, p. 234.

8. The standards describing the protocols that are part of the Internet’s architecture are developed and standardized by the Internet Engineering Task Force. A list of the current “Official Internet Protocol Standards” is available at <http://www.rfc-editor.org>; the list is updated daily. The Internet-standards process used by the Internet Engineering Task Force is described in Bradner 1996. For a short overview of that process, see Computer Science and Telecommunications Board and National Research Council 2001, pp. 134–135. For introductions to the architecture of the Internet, see Comer 2000; Peterson and Davie 2007; Tanenbaum 2003.

9. The description in the text follows the TCP/IP reference model used by the Internet Engineering Task Force (see Peterson and Davie 2007, pp. 28–30; Tanenbaum 2003, pp. 41–44; Comer 2000, pp. 183–185). The terminology regarding the names

of the layers follows Comer 2000, pp. 183–185. As Comer notes, the link layer is sometimes called the *network interface layer*. An alternative reference model, the Open Systems Interconnection (OSI) Reference Model was developed by the International Organization for Standardization (ISO) (see Tanenbaum 2003, pp. 37–41; Peterson and Davie 2007, pp. 26–28). For a comparison of the two reference models, see Tanenbaum 2003, pp. 44–49; Comer 2000, pp. 185–186. For a discussion of the history of the OSI reference model, see Abbate 1999, pp. 167–179.

10. The link layer also includes protocols for transporting data packets across a point-to-point link. The use of the term “link layer” follows Comer 2000, p. 184. In the OSI reference architecture, this layer corresponds to two layers: the *physical layer* and the *data link layer*. The physical layer is responsible for transmitting raw bits over a communications link. The data link layer builds on the services of the physical layer to transmit data packets called *frames* from one end host to another end host on the same physical network. See Tanenbaum 2003, pp. 85–182 (discussing the physical layer) and 183–342 (discussing the data link layer).

11. On this division (end hosts implement all layers, while IP routers typically implement only lower layers, up to and including the Internet layer), see Kurose and Ross 2008, pp. 51–52. Although higher layers do not have to be implemented on routers, in practice routers implement higher layers to terminate management and routing protocols.

12. For a description of the service provided by IP, see Comer 2000, p. 97; Peterson and Davie 2007, pp. 236–237.

13. For empirical studies measuring the amount of packet re-ordering or packet loss in the Internet, see Paxson 1999; Jaiswal et al. 2007.

14. For overviews of IP interconnection and routing, see Comer 2000, pp. 56–60, 115–126; Peterson and Davie 2007, pp. 234–235, 250–254.

15. In reality, routers have an IP address for each physical network to which they attach. The same is true for “multi-homed hosts” that are connected to two or more physical networks. For details, see Comer 2000, p. 65. For overviews of IP addressing, see Comer 2000, pp. 63–75; Peterson and Davie 2007, pp. 248–250.

16. If the destination host is attached to the same physical network, the IP protocol at the source host does not send the datagram to the nearest router; it sends the datagram directly to the destination host, using the link-layer protocol of the physical network. This differs from networking technologies, such as GPRS or DOCSIS, in which two hosts can never send data directly to each other. By requiring hosts to send data via computers in the network, these technologies effectively make the existence of a network operator a technical requirement.

17. In many operating systems, a running application program is called a *process*, a *user process*, or a *task*. See Comer 2000, p. 197; Kurose and Ross 2008, pp. 85–86.

18. For a description of the mechanism (multiplexing and de-multiplexing) used to realize this goal, see Comer 2000, pp. 197–206, 243.

19. See Kurose and Ross 2008, pp. 195–199.

20. On this division (end hosts implement all layers, while IP routers typically implement only lower layers, up to and including the Internet layer), see pp. 51–52 of Kurose and Ross 2008. In practice, routers may implement higher layers to terminate routing protocols such as BGP or management protocols.

21. See the subsection of this chapter titled “The End-to-End Arguments and the Division of Functionality between the Internet Layer and the Transport Layer. The Broad Version.”

22. For descriptions of the service provided by TCP, see Peterson and Davie 2007, pp. 384–385, 387, 410–411; Comer 2000, pp. 209–211.

23. For descriptions of the service provided by UDP, see Peterson and Davie 2007, pp. 382–384; Comer 2000, pp. 197–207.

24. Thus, there is a difference between the application (the program that implements the protocol specification) and the protocol specification (which describes the protocol’s peer and service interface). See Kurose and Ross 2008, pp. 81–85.

25. More precisely, the format of the messages exchanged between different entities belonging to the protocol and the actions the receiver of a message is supposed to take are defined as part of the peer interface of the protocol.

26. The application layer in the reimbursement example in chapter 2 consists of only one application (reimbursement processing). Different protocol entities, such as the employee and the administrator in the reimbursement department, exchange messages according to rules, which constitute that protocol’s peer interface.

27. See Comer 2000, p. 577.

28. Following Clark (1997), a portability layer is also called a *spanning layer*. (See Messerschmitt 2000, pp. 185–186; Kavassalis and Lehr 2000, pp. 209–211.) The role of the Internet layer as a spanning or portability layer populated by a single protocol is discussed by Clark (1997, pp. 134–135) and by Kavassalis and Lehr (2000, pp. 209–211). The proposal for an Open Data Network Architecture for the National Information Infrastructure by the Computer Science and Telecommunications Board and National Research Council (1994, pp. 47–55) also proposes a technology-independent portability layer below the transport layer called the *open bearer service*, noting that the Internet Protocol is the analogue of the bearer service in the TCP/IP protocol suite. See Computer Science and Telecommunications Board and National Research Council 1994, p. 54.

29. See Comer 2000, pp. 575–579; Peterson and Davie 2007, pp. 28–30.

30. See Comer 2000, pp. 577–578; Peterson and Davie 2007, pp. 29–30; Computer Science and Telecommunications Board and National Research Council 2001, pp. 126–130. The “hourglass” metaphor is often ascribed to the Computer Science and Telecommunications Board and National Research Council 1994, pp. 51, 53. (See Clark 1997, pp. 134–135.) According to Carpenter and Brim (2002, section 7 at [HOURG]), however, it was first used by John Aschenbrenner in 1979, with reference to the ISO Open Systems Interconnection model.

31. On the effect of the use of the Internet layer as a technology-independent portability layer, see Clark 1997, pp. 134–135; Kavassalis and Lehr 2000, pp. 209–211; Computer Science and Telecommunications Board and National Research Council 2001, pp. 126–130. For a detailed discussion of the benefits of such an approach, see Computer Science and Telecommunications Board and National Research Council 1994, pp. 50–55.

32. Saltzer, Reed, and Clark 1981 (narrow version); Reed, Saltzer, and Clark 1998 (broad version).

33. This point is taken up in more detail in the section on misconceptions below.

34. See the subsection of this chapter titled “The End-to-End Arguments and the Division of Functionality. . . : The Broad Version” below.

35. See Sterbenz and Touch 2001, pp. 351–352.

36. This aspect of the narrow version (i.e., that a function must sometimes be implemented at a higher layer at the end host, potentially in the application itself to guarantee a correct execution, if threats to the function may occur after the data leaves the communications subsystem) was already well established when the Internet’s architecture was developed. In their paper describing the narrow version, Saltzer, Reed, and Clark (1984) describe a number of earlier systems that were designed during the 1960s and the 1970s in which the designers used this argument to justify the ultimate placement of the function under consideration in the application.

37. Comer 2000, p. 210.

38. Implementing functions such as reliable data transfer or duplicate suppression in TCP also complies with the broad version. These functions are application-specific functions, which, according to the broad version, should be implemented in a higher layer at an end host.

39. The approach is summarized in a critical statement by John Postel (1977a, p. 1): “Specifically we are trying to use TCP to do two things: serve as a host level end to end [*sic*] protocol, and to serve as an Internet packaging and routing protocol.” For technical descriptions of the initial monolithic protocol, see Cerf and Kahn 1974 (the first published description), Cerf 1977, and Cerf and Postel 1978.

40. Originally, the designers assumed that the Internetwork Transmission Control Protocol could be designed to support any needed type of service, not just reliable

data transfer. The initial TCP attempted to provide two types of reliable data transfer with different combinations of throughput and delay. The designers came to the conclusion that accommodating all types of services in a single protocol would be too complex. See D. D. Clark 1988, pp. 107–108.

41. Even if the implementation of the function constituted only a partial implementation of the function (e.g., in cases where the Internet layer does not operate end-to-end between the original source and ultimate destination of data, or if events falling into the responsibility of the function may occur after data leaves the Internet layer), the narrow version would still have allowed the implementation of the function at the Internet layer as a performance enhancement.

42. See discussion in the next subsection.

43. See discussion in the next subsection.

44. See references cited in note 39 above.

45. See discussion in note 40 above.

46. See generally Reed 2000, box 3.3 below, and references cited in notes 51–53 below.

47. See Reed, Saltzer, and Clark 1998, p. 70. Also see the discussion of the broad version in chapter 2 above.

48. See Sproull and Cohen 1978, p. 1382; Boggs et al. 1980, pp. 615–616. For an overview of the needs of different applications, see Kurose and Ross 2008, pp. 88–94; Peterson and Davie 2007, pp. 499–505.

49. See Saltzer, Reed, and Clark 1984, p. 282, note 1; Kurose and Ross 2008, p. 589.

50. See D. D. Clark 1988, pp. 108–109.

51. See Sproull and Cohen 1978, pp. 1377, 1379, 1382; Boggs et al. 1980, p. 615; Saltzer, Reed, and Clark 1984, pp. 284–285. Similarly, computer-to-computer communications that exchange single packets in a simple request-response pattern do not need a separate acknowledgment from a lower-layer protocol that the request was received correctly. If the receiver sends the reply, this is acknowledgment enough. If the receiver does not reply, the sender can simply send another request. See Clark, Pogran, and Reed 1978, p. 1511; Boggs et al. 1980, p. 615.

52. See Sterbenz and Touch 2001, pp. 131–132, where connectionless and connection-oriented packet switching are compared. Also see Peterson and Davie 2007, pp. 410–411, where the signaling overhead associated with setting up and closing connections in TCP is discussed.

53. See Clark, Pogran, and Reed 1978, pp. 1511–1512; Reed 2000. See also box 3.3 below.

54. Boggs et al. 1980, p. 613; D. D. Clark 1988, pp. 108–109, 112; Reed 2000. See also box 3.3 above. Critics have argued that the service provided by the Internet's original architecture—unreliable, connectionless, best-effort datagram service—is not general enough, because the Internet Protocol does not offer Quality of Service. This argument is discussed in box 4.3 below.

55. For a description of the mechanisms used by the Transmission Control Protocol to provide reliable data transfer based on the unreliable data transfer provided by the Internet Protocol, see Kurose and Ross 2008, pp. 252–260.

56. Following the meeting where the decision to split was made, the proposals for the headers for the two protocols were made in Cerf 1978a and in Cerf 1978b. The first full specifications of TCP and IP after the split were the specification of the Internet Datagram Protocol Version 4 (Postel 1979a) and of the Transmission Control Protocol—TCP (Version 4) (Postel 1979b). See also Cerf 1980a, abstract and p. 4. The User Datagram Protocol (UDP) was also discussed at the meeting on January 30–31, 1978; the first specification of UDP was prepared by David Reed and Jon Postel (Reed and Postel 1978). See Postel 1978a, pp. 12–13; Reed 2006a,b.

57. See Cohen 1979, pp. 178–179; Boggs et al. 1980, pp. 613–614; D. D. Clark 1988, pp. 108–109; Reed 2000. See also box 3.3 below.

58. See Cerf and Kirstein 1978, pp. 1397–1398; Cerf 1980b, p. 11; Boggs et al. 1980, pp. 615–616; D. D. Clark 1988, pp. 108–109.

59. This implies a need for some suitable form of addressing. See D. D. Clark 1988, p. 109.

60. D. D. Clark 1988, pp. 109–110; Peterson and Davie 2007, p. 237; Computer Science and Telecommunications Board and National Research Council 2001, p. 37, note 14; Computer Science and Telecommunications Board and National Research Council 1994, pp. 54–55; Kurose and Ross 2008, pp. 323–324.

61. Cerf 1980b, p. 11; D. D. Clark 1988, pp. 107–108; Carpenter 1996, section 2.3.

62. The term “fate-sharing” was coined by David Clark (1988, p. 108).

63. For example, under Jon Postel's (1977a, pp. 5, 7) proposal for splitting the Internetwork Transmission Control Program into two protocols, end-to-end error control would have been removed from the Internet layer and provided by individual higher-layer protocols at the end hosts, but the Internet Protocol would still have performed error control hop-by-hop.

64. “The internet protocol does not provide a reliable communication facility. There are no acknowledgments either end-to-end or hop-by-hop. There is no error control for data, only a header checksum. There are no retransmissions. There is no flow control.” (Postel 1979a, pp. 2–3) For the justification, see Postel 1981a, section 3.2: “There are some applications where a few data bit errors are acceptable while retrans-

mission delays are not. If the Internet Protocol enforced data correctness such applications could not be supported.”

65. See the subsection of this chapter titled “The End-to-End Arguments and the Division of Functionality. . . : The Narrow Version” above.

66. Solum and Chung 2004, pp. 844–845.

67. See also Felten 2003.

68. See the discussion of the layering principle in chapter 2 above.

69. See Saltzer, Reed, and Clark 1984, p. 287.

70. Reed, Saltzer, and Clark 1998, p. 69.

71. For a description of the X.25 protocol suite and its relationship to the ISO reference model of Open System Interconnection, see Comer 2000, pp. 182–183. For a comparison of the X.25 protocol suite and the TCP/IP protocol suite, see Comer 2000, pp. 185–186; Peterson and Davie 2007, p. 387.

72. See subsection “The End-to-End Arguments and the Division of Functionality . . . : The Broad Version” above.

73. This view is very common in the Internet community. Kempf and Austein (2004, section 2.2) mention it explicitly as one “formulation of the end-to-end principle”: “The end-to-end principle in this formulation is specifically about what kind of state is maintained where.” For another example of this view, see Gaynor 2003, p. 36.

74. Carpenter 1996, section 2.3.

75. See Chiappa 2002; Reed 2001.

76. David 2001, pp. 171–174; Frischmann 2005, pp. 1010–1011; Weiser 2008, pp. 277–278.

77. For a similar argument, see Reed 2001.

78. See Peterson and Davie 2007, pp. 499–525.

79. For a similar argument that also differentiates between the request for quality of service and the actual provision of the service, see Solum and Chung 2003, pp. 109–111.

80. See Kurose and Ross 2008, p. 199.

81. Blumenthal and Clark 2001, p. 80 (citing personal communication with Jerome Saltzer).

82. For a description of the corresponding technology, see Anderson 2007. For a specific example of such technology, see Cisco Systems 2005.

83. See the discussion of application-blindness in chapter 2 above. Also see Solum and Chung 2003, p. 114.

84. See Lessig 2001, pp. 37–38; Solum and Chung 2004, p. 829; Gaynor 2003, pp. 35–39.

85. Reed 2001.

86. Sandvig 2002, pp. 20–21. See also Gaynor et al. 2000, pp. 3, 4; Gaynor 2003, p. 45. Gaynor argues that applications with a decentralized management structure comply with the end-to-end arguments, while applications with a centralized management structure do not. According to Gaynor (2003, p. 26), “centralized management occurs when users cross organizational boundaries by accessing a server managed by someone unrelated to the users, or a centralized server manages the users [*sic*] data.”

87. In the text, the term “edge of the network” is meant to describe the collection of end hosts, not the access networks.

88. According to Gaynor, applications where a part of the application is controlled centrally will be less innovated upon. See Gaynor 2001, pp. 32–36; Gaynor 2003, pp. 47–72, 245–249; Gaynor et al. 2000. I am not sure this is necessarily correct. An application with a server that is controlled and operated by the application provider may be able to evolve more quickly (at least in the server part) than an application whose parts are distributed among a number of different parties, since changes are easier to deploy. See chapter 4 below.

89. See chapter 2 above. Also see Reed 2002.

90. Saltzer, Reed, and Clark 1981; Reed, Saltzer, and Clark 1998.

91. Work on the Internet’s architecture started in the early 1970s; the specifications of what I call the Internet’s “original architecture” were published in 1981 (Postel 1981a,b). The argument described in the text has been made by Solum and Chung (2004, pp. 844–845).

92. See, e.g., the introductory paragraph of the 1984 paper (Saltzer, Reed, and Clark 1984, p. 277): “This paper discusses one class of function placement argument that has been used for many years with neither explicit recognition nor much conviction. However, the emergence of the data communication network as a computer system component has sharpened this line of function placement argument by making more apparent the situations in which and the reasons why it applies. This paper articulates the argument explicitly, so as to examine its nature and to see how general it really is.”

93. Saltzer, Reed, and Clark 1984, pp. 285–287; Reed, Saltzer, and Clark 1998, p. 70.

94. See Reed 2000; Blumenthal and Clark 2001, pp. 71–72.

95. Saltzer, Reed, and Clark 1984, pp. 278–286.

96. Reed, Saltzer, and Clark 1998, p. 70.

Chapter 4

1. In a software-intensive system, the total amount of change corresponds to the total number of lines of code that need to be changed across all components. Thus, even if the overall number of lines of code that must be changed is the same, the costs of change are larger if these lines of codes are distributed over a number of components than if they are concentrated in one component.

2. Once a software system reaches a certain size, changing software components may become more expensive than changing hardware components, since testing becomes more and more expensive as a software system's size and complexity increase.

3. For example, some innovators may intend to sell their innovation; others may innovate to use it for themselves. Some actors may be motivated by pecuniary rewards; others may be motivated by intrinsic hedonic rewards or social-psychological rewards. For a discussion of how actors may differ in their preferences for these types of rewards and an analysis of the effect of these differences, see Benkler 2002, pp. 426–434.

4. For a similar assumption, see p. 68 of Lipsey, Carlaw, and Bekar 2005.

5. The costs of realizing an innovation differ from the costs of adopting an innovation. The costs of adopting an innovation are the costs that a user (and potentially other economic actors) must incur in order to be able to use the innovation.

6. See Sanchez 1999, p. 95; Ulrich 1995, pp. 426–428.

7. Adding a new interface to a component without changing existing interfaces or adding a new component to an existing system that uses the services of existing components in order to provide its own services constitutes an autonomous change, although it adds visible information to the architecture. Because a new interface is not yet used by other components, other components cannot have come to depend on it. Therefore, the addition of a new interface does not affect existing components in the rest of the system.

8. In practice, a system architect may still be involved for other reasons. If system architects want to encourage third-party innovators to produce new or improved components that are complementary to the platform component offered by the system architect, they often provide tools and advice that make it easier to innovate. For example, Intel, Microsoft, and Apple all provide information on new interface specifications of their products (and, sometimes, prototype implementations of the platform component) to third-party developers, hold forums and conferences for

third-party developers, and provide development tools that make it easier to build complementary components. See Gawer and Cusumano 2002, pp. 55–67 (describing Intel’s efforts) and 150–151 (describing Microsoft’s efforts), and Evans, Hagiu, and Schmalensee 2006, pp. 98–101 (describing Microsoft’s and Apple’s efforts).

9. Adding a new interface to an existing component does not change the architecture’s existing visible information. Since it does not trigger adaptation costs, it constitutes an autonomous innovation. See note 7 above.

10. This does not mean that the new screen technology will never find its way into the iMac. Though the improvement in a single technology may not justify a complete redesign, at some stage the sum of the benefits associated with improvements in different technologies may exceed the costs of a complete redesign.

11. On options theory, see Hull 2002; Luenberger 1997; Merton 1992. For a short overview of the main concepts, see Sullivan et al. 1999, pp. 224–225.

12. See Dixit and Pindyck 1994; Trigeorgis 1996; Amram and Kulatilaka 1999.

13. See Sullivan et al. 1999, pp. 224–225.

14. See Amram and Kulatilaka 1999, chapter 2; Sullivan et al. 1999, p. 224.

15. See Amram and Kulatilaka 1999, pp. 14–20, 79–80; Sullivan et al. 1999, p. 225.

16. The seminal articles in this area are Black and Scholes 1973 and Merton 1973b.

17. Real options were first identified by Myers (1977). For an introduction to the theory of real options, see Dixit and Pindyck 1994; Trigeorgis 1996; Amram and Kulatilaka 1999. For a short overview of the history of the field, see Sanchez 1991, pp. 17–19; Sullivan et al. 1999, pp. 223, 226.

18. These conditions have been derived from the discussions in chapter 2 of Amram and Kulatilaka 1999 and in chapter 1 of Sanchez 1991.

19. This is based on the assumption that a firm that seeks to maximize its economic value will seek to maximize its net present value, and therefore choose the set of real options that maximizes the net present value of the firm. See Sanchez 1991, pp. 25–26, 71–72.

20. See Amram and Kulatilaka 1999, chapters 4 and 5.

21. See *ibid.*, pp. 28, 81–82.

22. The following analysis is based on Baldwin and Clark 2000 and Sanchez 1991.

23. Baldwin and Clark 2000, pp. 234–235, 252–255.

24. Enkel, Perez-Freije, and Gassmann 2005, p. 426; Souder, Sherman, and Davies-Cooper 1998, pp. 521–522.

25. See Baldwin and Clark 2000, pp. 255–257.
26. This follows from the assumption that innovators will not pursue projects whose expected benefits are lower than their expected costs.
27. See Sanchez 1991, pp. 25–26; Baldwin and Clark 2000, pp. 250–251.
28. Baldwin and Clark 2000, pp. 259–260.
29. *Ibid.*, pp. 255–257.
30. See Baldwin and Clark 2000, pp. 236–237; Garud and Kumaraswamy 1995, pp. 96–98; Ulrich 1995, pp. 426–432.
31. Depending on the system, it may also be possible to mix and match different components so that a variety of systems can be built from a set of components that adhere to the same architecture. In addition, a modular architecture can evolve at the architectural level through gradual changes to its visible information. Although these changes are more costly than isolated changes at the module level, they probably will still be less costly than the redesign of an integrated architecture, as all components unaffected by the changes can be reused. See Garud and Kumaraswamy 1995; Sanchez 1995; Ulrich 1995; Baldwin and Clark 2000.
32. Thus, an innovator must have an existing version of the module or must be able to fall back on an existing textbook solution to the problem before she can realize the option value associated with a modular architecture. This is because without access to such an existing solution the innovator cannot cut her losses by continuing to use the existing solution if the experiment fails. See Baldwin and Clark 2000, p. 252, note 5.
33. Baldwin and Clark 2000, pp. 257–260; Sullivan et al. 2001, p. 102.
34. Merton 1973a.
35. Baldwin and Clark 2000, pp. 259–260. Baldwin and Clark have developed a model that quantifies the difference between the option value of a modular architecture and the option value of an equivalent integrated architecture for modular architectures consisting of modules of equal size and of asymmetric size; see Baldwin and Clark 2000 (p. 259 for the symmetric case and p. 261 for the asymmetric case). Their model does not include the costs of creating the modular architecture. My analysis in the text considers only the option value of modular and integrated architectures. In comparing the value of an integrated and a modular architecture, an individual company will also consider what portion of the option value it expects to be able to capture for itself. This portion may be higher under an integrated architecture, since in an integrated architecture the system architect controls subsequent innovation and may therefore be better able to appropriate the gains from subsequent innovation than in a modular architecture, in which control over module-level innovation may move to the designers of individual modules. This is discussed further in chapter 5 below.

36. For example, the success or failure of a particular technical approach to improving screen technologies probably is not related to the success or failure of approaches that would increase the storage capacity of hard disks.

37. Baldwin and Clark 2000, pp. 285–286.

38. See *ibid.*, chapters 10, 12, and 13; Sanchez 1991, part III.

39. Baldwin and Clark 2000, chapter 12.

40. See *ibid.*, p. 312.

41. *Ibid.*, p. 313.

42. It also depends on the diversity of the approaches. See note 55 below.

43. This is because the incremental benefit from an additional approach decreases with the number of approaches. See the discussion later in this subsection.

44. How many approaches will be pursued also depends on the market structures of the design and product markets, and on innovators' and investors' expectations with respect to competitors' actions (such as the level of entry), the nature of the subsequent competition, or the total rewards that may be expected. See the discussion later in this subsection.

45. The approach described in the text can also be used to assess the effect of differences in modules within a certain modular architecture on the distribution of innovation among the modules of that architecture. See Baldwin and Clark 2000, chapter 11.

46. The following analysis is based on Baldwin and Clark 2000, Sanchez 1991, and Krengel 1991.

47. In the model described in the text, the costs of comparing the result of the experiment with the existing version of the module are part of that module's core costs of innovation. See Baldwin and Clark 2000, p. 286. The model assumes that the relative quality of the innovation can be determined using module-level tests. See next note.

48. This statement embodies certain assumptions about the resolution of uncertainty. This modeling assumes that the uncertainty can be resolved by testing the module using module-level tests. In other words, the modeling assumes, first, that module-level tests are available that make it possible to evaluate the quality of a module based on the module without having to integrate the module into the larger system. This requirement may not be met for all systems. Often module-level tests are not available until designers have gained some experience with a system. On the importance of module-level tests, see Baldwin and Clark 2000, pp. 271–280. Second, the modeling assumes that the uncertainty can be resolved using module-level tests. This is a reasonable assumption for technical uncertainty

if the module-level tests can determine which outcome is the best. If the uncertainty results from uncertainty about consumers' preferences, it may not be possible to resolve the uncertainty using module-level tests in all cases. Though it may be possible to resolve market uncertainty through the use of focus groups, sometimes market uncertainty can be resolved only by putting the product on the market.

49. A random variable is a function that associates a unique numerical value with every outcome of an experiment. Random variables can be described by probability distributions that provide information about the probability with which different values of the random variable may occur (for discrete random variables) or about the probability that the value may fall within a certain interval (for continuous random variables).

50. Formally, the expected value is the sum of all potential values of the random variable, weighted by their associated probabilities.

51. This formulation reflects the loss-limiting characteristic of the option. Although the value of the best of k experiments can be better or worse than the existing module version, the calculation of the option value considers only the potential outcomes of the best of k experiments that are better than the existing module (technically, the option value equals the expected value of the positive value portion of the probability distribution of the random variable describing the best of the k experiments). For outcomes where the best of k experiments is worse than the existing module, the innovator continues to use the existing module, and its payoff is zero. See box 4.2.

52. Baldwin and Clark 2000, pp. 252–255.

53. In a normal distribution, about 68% of the outcomes fall within one standard deviation σ from the mean; in other words, 68% of the values lie between $\mu - \sigma$ and $\mu + \sigma$. About 5% are more than two standard deviations away from the mean; that is, 5% of the values are smaller than $\mu - (2 \times \sigma)$ or larger than $\mu + (2 \times \sigma)$.

54. The variance of a random variable is a non-negative number that gives an idea of how widely spread the values of the random variable are likely to be. The larger the variance, the further individual values of the random variable tend to be from the mean, on average. The smaller the variance, the closer individual values tend to be to the mean, on average. In other words, if the variance is large, it is more likely (relative to a probability distribution with the same mean and smaller variance) that the result of an experiment will be far away from the mean. In practice, measuring uncertainty may be quite a challenge. On how one type of uncertainty (market uncertainty) may be measured, see Gaynor 2003, pp. 86–91.

55. In reality, approaches may not be completely independent. Whereas approaches that are essentially similar, with only slight variation, might provide a more limited

set of results, a more varied set of approaches might provide a greater variety of results and therefore an increased likelihood of discovering a module that is better than the existing one. Owing to the reduction in the expected value of additional experiments, correlation across experiments would reduce the number of experiments that innovators would pursue. In the model, such correlation could be modeled by reducing σ_i , the parameter that models uncertainty. See Baldwin and Clark 2000, p. 357. In addition, over several generations of experiments on a module, innovators may gain experience with the module, which may also reduce uncertainty. For a formal model of this effect of learning, see Gaynor 2001, pp. 141–155.

56. For a formal definition of $Q(k)$, see box 4.2. See Baldwin and Clark 2000, p. 264, note 17.

57. $Q_i(k)$ is monotonically increasing and concave in k . See Baldwin and Clark 2002, p. 7, note 6.

58. This terminology is taken from pp. 75 and 225 of Baldwin and Clark 2000.

59. For a detailed analysis of the effect of different market structures and of differing expectations, see Baldwin and Clark 2000, chapter 14. For an analysis of decision rules under different market structures, see Baldwin and Clark 2000, pp. 410–411.

60. In reality, firms probably would not know exactly how many competitors had already started an experiment.

61. Baldwin and Clark (2000, pp. 287–288, 293–299) use a variant of the net option value formula described in the text to calculate differences in net option value for different modules in two hypothetical architectures. The results of the simulation support the conclusions reached in the text.

62. The visibility of a module increases with the number of other modules that directly or indirectly depend on this module.

63. See also Gaynor 2003, pp. 266–267.

64. At this point, an actor that is a monopolist in the design and product market would stop mounting additional approaches. Actors in markets with free entry into the design market and winner-take-all competition in the product market would start additional approaches as long as the net option value is greater than zero.

65. The loss-limiting characteristic of options applies only if the innovator already has an existing version of the component that it can continue to use if the result of the innovation is worse. Otherwise, the option value of innovating on the module will be realized only in the second generation of the module. See Baldwin and Clark 2000, p. 252, note 5; Baldwin and Clark 2002, p. 7.

66. For example, in simulation based on a variant of the model described in the text, Baldwin and Clark (2000, pp. 287–288, 293–299) found that modules that were visible to 50% of the system did not justify any innovation on their visible information.

67. See Baldwin and Clark 2000, pp. 289–293.

68. Throughout this book, the costs of innovation are the costs of developing and producing the first complete version of the system that contains the new or improved good or service. In networking, innovating may consist of a number of stages, so the costs of innovation may be borne by different actors. Innovating on networking protocols or technologies consists of, first, specifying the protocol's visible information such as its service and peer interfaces; conceptually, this step corresponds to the design of the architecture of the protocol. For protocols that undergo standardization, this step is usually performed within the standardization organization. Subsequent stages are detailed design, which is followed by implementation (if the protocol is implemented in software) or production (if the protocol is implemented in hardware) of the first version of the protocol. These steps are usually performed by actors other than the standardization organization.

69. Conceptually, the costs of deployment are not costs of innovation, but costs of adoption. They can be incurred only after the first version of the innovation has been produced. The costs of deployment are different from the costs of using the innovation. The costs of deployment are the one-time costs that must be incurred before the innovation can be used. See Thaler and Aboba 2008, p. 6 (noting the difference between deployment and use of a protocol).

70. The economic system in which the network is used consists of the actors who use and operate the network, the relationships among them and the governance structures through which they interact.

71. I use the term “commercial Internet” to denote the economic system for the operation and use of the Internet in the US and elsewhere as it has existed since the mid 1990s. In this system, the Internet backbones (i.e., the bigger networks used to interconnect smaller networks attached to the Internet) and the access networks that users use to connect to the Internet are owned, operated, and controlled primarily by private actors who operate networks for a profit. Throughout its early history, much of the US portion of the Internet was owned, operated, and controlled by government agencies, including the Advanced Research Projects Agency and, later, the National Science Foundation. In the late 1980s, the NSF backbone became the main backbone of the Internet. The NSF's Acceptable Use Policy explicitly prohibited the use of the backbone for commercial uses. From 1990 to 1995, the Internet gradually transitioned to the commercial mode of operation that we know today. For a description of this process, see Abbate 1999, pp. 195–200; Kesan and Shah 2001, pp. 99–167.

72. See Isenberg 1998, p. 29; Isenberg 1997; Moors 2002, pp. 1217–1218.

73. See the subsection of chapter 3 titled “The End-to-End Arguments and the Division of Functionality. . . : The Broad Version” above.

74. Although Skype uses UDP as a default transport-layer protocol, it falls back on TCP when a user's network-address translator or firewall effectively blocks UDP

(Handley 2006, pp. 124–125; Kurose and Ross 2008, p. 609). As I will discuss in more detail in my concluding chapter, network-address translators and firewalls deviate from the broad version of the end-to-end arguments. When these devices force Skype to fall back on a transport-layer protocol with reliability controls, they effectively re-create the situation that the split between TCP and IP—or, more generally, the broad version of the end-to-end arguments—was designed to prevent.

75. Innovations that require only changes to end hosts can be tested on the operational Internet (or at least on a subset). If changes to computers in the core of the network are needed, an innovator cannot test his innovation on the operational network and must test its innovation on a simulated network or on a private testbed instead. See Computer Science and Telecommunications Board and National Research Council 1994, pp. 103–104. See also Peterson and Davie 2007, pp. 496–497.

76. The problems for innovations in the core of the network that require a coordinated deployment are discussed in more detail in the present chapter's subsection on "Internet Architecture and Innovation at the Internet and Transport Layer."

77. See Blumenthal and Clark 2001, p. 72.

78. Today applications often have a more distributed structure, with more than two protocol implementations cooperating to provide the functionality of the protocol.

79. In the Internet, the application programming interfaces through which applications access the Internet are publicly available for free or at low cost. This is an example of a non-architectural factor that interacts with the broad version of the end-to-end arguments to create a beneficial environment for application innovation in the Internet. In order to test the application, the innovator will have to find someone who is willing and able to install the application on a second end host. But having access to one end host with access to the Internet will probably enable the innovator to find such a person.

80. See von Hippel 2002, pp. 21–22.

81. Computer Science and Telecommunications Board and National Research Council 1996, pp. 77–78.

82. These ongoing costs associated with making an application available to others do not belong to the costs of innovation. Like the costs of production and distribution, they are discussed here because they determine the investment required for exploiting the innovation. The costs described in the text are the minimum costs associated with the exploitation of an innovation that define the minimum requirements for an innovator who writes an application that other people may use. Beyond that, an innovator may want to market the application or offer customer support.

83. Roth 2004. For a high-level description of Skype's architecture, see Kurose and Ross 2008, pp. 157–159.

84. Kraus 2005a; Buckman 2005; Helft 2006. In addition, search-engine marketing enables small companies to reach potential audiences at low cost.

85. Amazon offers a number of storage and computing services that are based on Amazon's own infrastructure to other firms. Amazon passes on some of the benefits of its own economies of scale to these companies (Reiss 2008).

86. See Isenberg 1998, p. 29.

87. The effect of differences in the costs and mode of application deployment is discussed in this chapter's subsection on "Internet Architecture and Innovation at the Internet and Transport Layer" and in chapter 7's section on "Relative Costs of Application Deployment."

88. Innovations that do not trigger changes to the network would still be possible.

89. This is because applications are placed in the highest layer of the protocol stack, and lower-layer protocols are not allowed to depend on higher layers.

90. In contrast, the high costs of producing and distributing physical products make it much more difficult to bring the product to market in order to test consumers' reaction to it.

91. See Computer Science and Telecommunications Board and National Research Council 1994, pp. 102–103.

92. The desire not to change the Internet Protocol's service interface to higher layers (see next section) effectively constrains the type of innovations that will occur at the layers below the Internet layer to those that make sense given the Internet Protocol's existing service interface. Most likely, innovators at the lower layers will not put a lot of effort into changes that could not be exposed to higher layers given the Internet Protocol's existing service interface.

93. On the use of the Internet layer as a technology-independent portability layer, see Clark 1997, pp. 133–135; Kavassalis and Lehr 2000, pp. 209–211; Computer Science and Telecommunications Board and National Research Council 2001, pp. 126–130. Another detailed discussion of the benefits of such an approach can be found on pp. 47–55 of Computer Science and Telecommunications Board and National Research Council 1994.

94. While wireless networks do not require any changes to higher-layer protocols at the application layer and the transport layer (i.e., higher-layer protocols work over wireless links attached to the Internet), the technical characteristics of wireless links challenge some of the assumptions on which higher-layer protocols are based. For a short description of the problem and possible solutions, see Kurose and Ross 2008, pp. 575–578.

95. See Comer 2000, chapter 3; Peterson and Davie 2007, pp. 234–235.
96. The economic system in which the network is used consists of the actors who use and operate the network, the relationships among them and the governance structures through which they interact.
97. It is interesting to note the difference between this design and the design of the Internet Protocol. As IP's service interface requires the higher-layer protocol to provide the IP address of the receiving end host, a change in the format of the IP address automatically affects the service interface. As a result, a change in the IP address requires changes to all higher-layer protocols that make use of the IP address, which entail huge costs of system adaptation.
98. Changing the implementation of an architectural component in a modular architecture without changing its visible information does not require any changes in the rest of the system. Consequently, changing the implementation of a protocol without changing its interfaces affects neither higher-layer protocols that use its services nor implementations of the same protocol at different computers with which it interoperates. Thus, it is possible to change the implementation of a protocol at one computer only; innovations of this type can be deployed incrementally. See Peterson and Davie 2007, p. 23.
99. See the examples discussed by Peterson and Davie (2007, p. 441) and Handley (2006, p. 120). An example is a series of refinements of the mechanism that TCP uses to decide whether to retransmit a segment. These refinements, however, did not change the format of the TCP header and could therefore be incorporated into implementations individually. See Peterson and Davie 2007, p. 435.
100. Fielding and Taylor (2002, pp. 118, 119) discuss this requirement in the context of the design of the World Wide Web.
101. For a discussion of mechanisms used by the designers of HTTP to support the extensibility of the protocol, see Fielding and Taylor 2002, pp. 138–140. For a critical voice on the mechanisms for extensibility provided by HTTP, see Mogul 2004. Clark et al. (2003, p. 248, note 1) note the implementation and validation overhead and interoperability problems that may be associated with the extended use of options.
102. See Peterson and Davie 2007, p. 441.
103. Because it is not realistic to assume that all network providers connected to the global Internet would switch to a new Internet-layer protocol at the same time, networking researchers have developed various mechanisms that would support a more incremental transition to new Internet-layer protocols. See, e.g., the overview of mechanisms that support the transition to IPv6 in Kurose and Ross 2008 (pp. 363–366) or the discussion of overlays as transitioning mechanisms in Peterson and Davie 2007 (pp. 693–696).

104. On multicast, see Diot et al. 2000; Sharma, Perry, and Malpani 2003. On Quality of Service, see Armitage 2003; Bell 2003; Burgstahler et al. 2003.

105. Anderson et al. 2005a, p. 34. For example, the invention of network-address translators that allow a number of end hosts to connect to the Internet using the same globally unique IP address has reduced network providers' and users' incentives to deploy IPv6. Network-address translators deviate from the broad version of the end-to-end arguments by implementing a considerable amount of application-specific functionality in the network. As will be discussed in my concluding chapter, this impedes the development of new applications considerably. Still, individual economic actors find it easier to install a network-address translator to deal with the shortage of IPv4 addresses (and impose a negative externality on application-level innovation) than to migrate their networks to IPv6, although all actors would be collectively better off as a result of the transition.

106. Anderson et al. 2005a, p. 34; Ratnasamy, Shenker, and McCanne 2005, p. 313; Handley 2006, pp. 121–122.

107. See generally Floyd 2002, p. 17.

108. See Diot et al. 2000; Sharma, Perry, and Malpani 2003; Bell 2003; Burgstahler et al. 2003.

109. Burness et al. 2005, pp. 2975–2977; Thaler and Aboba 2008, pp. 7–8.

110. Floyd 1998, pp. 2, 4; Burness et al. 2005, pp. 2974–2976; Thaler and Aboba 2008, p. 8.

111. Anderson et al. 2005a; Ratnasamy, Shenker, and McCanne 2005.

112. Anderson et al. 2005a,b; Turner and Taylor 2005; Roscoe 2006; Feamster, Gao, and Jennifer 2007. Anderson et al. (2005a) describe both approaches but do not take a position on which one is better. An architecture such as the one described in the text that enables several network architectures to coexist at the same time will form the core of a large-scale experimental networking testbed facility called Global Environment for Network Innovations (GENI). The GENI testbed will enable networking researchers to experiment with and test new network architectures in a large-scale testbed with realistic traffic. GENI is funded by the National Science Foundation in the US. See Anderson et al. 2005a,b; Clark, Shenker, and Falk 2007.

113. See North 2005, pp. 3–10, 73–91.

Chapter 5

1. The chapter does not explore how the existing organizational structures in which an architecture and its components are designed may affect the structure or evolution of the architecture. Three interesting studies of this question are Henderson and Clark 1990, MacCormack, Rusnak, and Baldwin 2008, and Fixson and Park 2008.

2. In addition, there may be some additional teams who are responsible for the integration at the system level. For an empirical example (the development of an aircraft engine), see Sosa, Eppinger, and Rowles 2002, p. 11.
3. Thompson 1967.
4. On product design, see McCord and Eppinger 1993; Sosa, Eppinger, and Rowles 2000, p. 3; Henderson and Clark 1990, p. 15. On software design, see Bass, Clements, and Kazman 2003, pp. 29, 167–168; van Vliet 2000, p. 92; Kruchten 1995, p. 45.
5. For an empirical example (the development of an aircraft engine), see Sosa, Eppinger, and Rowles 2002, p. 11.
6. Thompson 1967, p. 57. For the same prescription in the context of product development, see von Hippel 1990.
7. Thompson 1967, p. 58
8. *Ibid.*, pp. 54–56.
9. *Ibid.*, pp. 59–61.
10. See von Hippel 1990, p. 409.
11. See van den Bulte and Moenaert 1998; Tushman 1977; Sosa et al. 2002; Sosa, Eppinger, and Rowles 2002. For pointers to additional empirical studies that report similar results, see Sosa et al. 2002; Sosa, Eppinger, and Rowles 2002.
12. See Eppinger 1991, p. 284; Steward 1981; Baldwin and Clark 2000, pp. 43–47; Sanchez 1995, p. 145.
13. See Besanko, Dranove, and Shanley 2000, p. 109.
14. See Argyres 1996; Poppo and Zenger 1998; Combs and Ketchen 1999; Williamson 1999; Tsang 2000. This approach is also used by management strategy textbooks. See, e.g., Besanko, Dranove, and Shanley 2000, chapters 3–5.
15. See Poppo and Zenger 1998, p. 854; Besanko, Dranove, and Shanley 2000, pp. 139, 170.
16. For an overview of potential benefits of using the market, see Besanko, Dranove, and Shanley 2000, pp. 115–127.
17. See Williamson 1999, pp. 1102–1103.
18. See Williamson 1985, chapter 6; Williamson 1991, pp. 275–276, 279.
19. “As pointed out in the resource-based view of the firm, heterogeneity in such resources leads to differentiated capabilities among firms, putting them at a competitive advantage in some activities and at a disadvantage in others. Such differential returns from activities encourage firms to specialize in those activities in which their

rate of return is highest.” (Schilling and Steensma 2001, pp. 1153–1154, references omitted). See also Jacobides and Hitt 2005, p. 466.

20. See Besanko, Dranove, and Shanley 2000, pp. 115–122.

21. On maintaining capabilities, see Sobrero and Roberts 2002. On other strategic considerations, see Baldwin and Clark 2006a and chapter 1 of Boudreau 2006.

22. For empirical studies examining the organizational separation of detailed design and production, see Monteverde 1995 and Ulrich and Ellison 2005.

23. See Thompson 1967, pp. 56, 64.

24. For a similar observation, see Gulati and Singh 1998 and Jacobides 2002.

25. Gulati and Singh (1998) use a framework similar to the one described in the text to explain the choice of different levels of hierarchical control in alliances between firms. Though coordination costs have long been used to explain organizational structures within the firm, they have only recently been identified as a distinct cost category influencing firms’ governance choices with respect to vertical boundaries. (See box 5.1.)

26. Chesbrough and Teece (1996, p. 67) write: “Tacit knowledge is knowledge that is implicitly grasped or used but has not been fully articulated, such as the know-how of a master craftsman or the ingrained perspectives of a specific company or work unit. Because such knowledge is deeply embedded in individuals or companies, it tends to diffuse slowly and only with effort and the transfer of people. Established companies can protect the tacit knowledge they hold, sharing only codified information. They can be quite strategic about what they disclose and when they disclose it.” See also von Hippel 1994, pp. 430–431.

27. Sosa et al. (2002, p. 48) write: “Media richness theory . . . ranks communication media according to their capacity to process ambiguous information. [According to this theory], face-to-face is a richer medium than telephone, and telephone is a richer medium than e-mail.”

28. See Sosa et al. 2002, p. 48.

29. See Chesbrough and Teece 1996, p. 67; von Hippel 1994, pp. 429–432.

30. For empirical studies confirming the beneficial effect of co-location, see van den Bulte and Moenaert 1998 and Morelli, Eppinger, and Gulati 1995.

31. The emergence of firm-specific language and routines and their positive effects on the generation of new knowledge and its efficient dissemination have been well developed theoretically by proponents of the knowledge-based theory of the firm (Demsetz 1988; Monteverde 1995; Kogut and Zander 1996). This advantage of internal organization can be reduced or eliminated by the formation of common communication codes across firms (as in Silicon Valley and other regional networks).

Mechanisms that establish “common ground” (knowledge that is shared, and known to be shared, among partners in an outsourcing relationship) may also make it easier for firms to coordinate interdependent tasks across firm boundaries (Srikanth and Puranam 2007, 2008).

32. See Jacobides 2002, pp. 1–12.

33. See *ibid.*, pp. 30–33.

34. See Gulati and Singh 1998, pp. 785–786.

35. See Williamson 1991, pp. 274–276; Besanko, Dranove, and Shanley 2000, pp. 161–162.

36. See Jacobides 2002, pp. 21–22.

37. See Novak and Stern 2003, p. 9; Besanko, Dranove, and Shanley 2000, pp. 132–134.

38. Thompson 1967, pp. 54–56.

39. Thompson’s analysis does not cover complete independence. Instead, Thompson (1967, p. 54) analyzes a similar type of interdependency called *pooled interdependency*.

40. Thompson 1967, p. 54.

41. *Ibid.*, pp. 54–55.

42. This type of coordination has been characterized as coordination by “standardization of work outputs.” See van Vliet 2000, p. 89 (referring to a classification by Mintzberg).

43. See Thompson 1967, p. 56; van Vliet 2000, p. 89.

44. See Monteverde 1995, pp. 1628–1629.

45. See Sanchez 1995, pp. 145–146.

46. For a similar observation in the context of product development, see Sanchez 1995, pp. 145–146.

47. See Sobrero and Roberts 2001; Sobrero and Roberts 2002; Takeishi 2001; Sako 2004; Srikanth and Puranam 2007, 2008. Brusoni (2005, p. 1901) notes the emergence of projects as “a temporary administrative framework within which some form of hierarchical coordination replaces the market.”

48. On the trade-off between efficiency and learning in R&D cooperation, see Sobrero and Roberts 2001.

49. See Dyer 1996, 1997; Takeishi 2001.

50. See Monteverde 1995.

51. Jacobides 2001; Jacobides 2002; Jacobides 2005.
52. Gulati and Singh 1998.
53. See the section “Integrated Design” in chapter 2 above.
54. See Sanchez 1995, pp. 145–146; Christensen, Verlinden, and Westerman 2002, pp. 958–960. For some examples, see Dyer 1996, 1997; Takeishi 2001; Sako 2004; Srikanth and Puranam 2007, 2008.
55. See Sanchez 1999, p. 95.
56. Baldwin and Clark 2000, pp. 64–72.
57. See Baldwin and Clark 2000, pp. 70–72, 372–375. Baldwin and Clark’s (2000, pp. 61–62) definition of “block-hierarchical” corresponds to my definition of sequential interdependence. See the subsection of this chapter titled “Linking Task Interdependency to Governance Choices.”
58. Ulrich 1995, pp. 435; Sanchez 1995, pp. 146–147; Baldwin and Clark 2000, p. 352; Christensen, Verlinden, and Westerman 2002, pp. 958–959.
59. Baldwin and Clark 2006a; Boudreau 2006, chapter 1.
60. Baldwin and Clark 2000, pp. 373–375.
61. Baldwin and Clark 2000, pp. 76–77. But see Brusoni and Prencipe 2001 (describing a more central role of system integrators).
62. See Sosa, Eppinger, and Rowles 2004; Sosa, Gargiulo, and Rowles 2007; Staudenmayer, Tripsas, and Tucci 2005.
63. Baldwin and Clark 2000, pp. 76–77, 86.
64. See Chesbrough and Teece 1996, p. 67.
65. As a module’s internal design is hidden from the system’s architects as well as from the designers of other modules, neither visible design parameters at the architectural level nor hidden design parameters in other modules depend on that information. As a result, changes to a module’s hidden information cannot require changes in the rest of the system as long as the module’s visible information is unaffected. To be able to interoperate with the rest of the system, the new design must comply with the relevant interface specifications. In this case, the innovation can occur completely independent of the rest of the system. As a result, internal organization and arm’s-length relationships are equally feasible organizational choices.
66. See Baldwin and Clark 2000, pp. 267–269.
67. *Ibid.*, pp. 267–269.
68. See Garud and Kumaraswamy 1995, pp. 100–101.

69. Williamson 1985, pp. 20–22. The origins of transaction-cost theory can be traced back to Ronald Coase (1937, 1960). The transaction-cost approach was operationalized by Oliver Williamson (1975, 1985). For a short overview of the history and content of Williamson's contributions to transaction-cost theory, see Masten 1999. For overviews of empirical studies testing the predictions of transaction-cost theory, see Rindfleisch and Heide 1997 and Shelanski and Klein 1999.

70. According to the traditional transaction-cost analysis, this factor shifts firms' preferences toward market-based organization if uncertainty, asset specificity, and measurement problems are below a certain threshold. See Williamson 1985, pp. 90–95.

71. For a comparison of the behavioral assumptions of transaction-cost economics with those of neoclassical economics, see Williamson 1985, pp. 44–52.

72. Simon (1961, pp. xxiv), cited by Williamson (1985, p. 45).

73. Williamson 1985, pp. 45–46.

74. *Ibid.*, pp. 47–49.

75. With respect to the following, see Milgrom and Roberts 1992, pp. 126–131; Besanko, Dranove, and Shanley 2000, pp. 144–147.

76. For an econometric study of the relative importance of transaction costs and heterogeneous capabilities, see Jacobides and Hitt 2005.

77. On the effect of uncertainty, see Williamson 1985, pp. 56–63.

78. On the effect of asset specificity, see, e.g., Williamson 1985, pp. 52–63; Besanko, Dranove, and Shanley 2000, pp. 148–160; Milgrom and Roberts 1992, pp. 134–139.

79. See Williamson 1985, p. 78; Williamson 1991, p. 280.

80. Williamson 1975, p. 29.

81. *Ibid.*, pp. 25, 37–39.

82. *Ibid.*, p. 29.

83. *Ibid.*, pp. 29–30, 35.

84. See Jacobides 2002, pp. 21–22.

85. See Alchian and Demsetz 1972, Jensen and Meckling 1976, and Barzel 1982. On moral hazard, principal-agent theory, and performance incentives in general, see chapter 6 of Milgrom and Roberts 1992.

86. See Alchian and Demsetz 1972, p. 780.

87. Holmstrom and Milgrom 1994.

88. Williamson 1975, p. 29.

89. For an overview of alternative governance structures in general, see Milgrom and Roberts 1992, pp. 561–569; Besanko, Dranove, and Shanley 2000, pp. 182–194.
90. Dyer 1997, p. 537.
91. See *ibid.*, pp. 547–549.
92. Dyer (*ibid.*, pp. 547–549) calls these mechanisms, which reduce transaction costs over a long period of time, “self-enforcing safeguards.”
93. For an alternative explanation focusing on the difficulties associated with defining, counting, and compensating for each transaction at different points in the architecture, see Baldwin 2008.
94. See Dyer 1996, p. 275; Besanko, Dranove, and Shanley 2000, p. 163.
95. Besanko, Dranove, and Shanley (2000, p. 151) write: “Human asset specificity refers to cases in which a worker, or group of workers, has acquired skills, know-how, and information that are more valuable inside a particular relationship than outside it.”
96. See Novak and Stern 2003, pp. 5–6, 9.
97. See also Baldwin 2008.
98. As we saw above, completely new or relatively young architectures may suffer from undetected cross-component dependencies, which increase uncertainty.
99. On module-level tests, see Baldwin and Clark 2000, pp. 77, 134–135, 272, 273, 277.
100. Garud and Kumaraswamy 1995, p. 103.
101. For a similar analysis, see Gawer 2000, pp. 64–65.
102. For in-depth analyses of the conditions under which an innovator may prefer selling its innovation in the “market for ideas” to entering the product market, see Gans and Stern 2000; Gans, Hsu, and Stern 2002; Gans and Stern 2003.
103. See Schilling and Steensma 2001; Jacobides and Hitt 2005; Sobrero and Roberts 2002; Baldwin and Clark 2006a; Boudreau 2006.
104. See also Sako 2002, pp. 14–15.
105. See Langlois and Robertson 1992; Baldwin and Clark 2000, chapter 14; Christensen, Verlinden, and Westerman 2002, p. 964.
106. Baldwin and Clark 2000, p. 351.
107. The word ‘complementor’, coined by Brandenburger and Nalebuff (1998, p. 16), denotes an economic actor that provides complements.
108. See Garud and Kumaraswamy 1995, p. 102.

109. See Baldwin and Clark 2000, p. 77.
110. See Christensen, Verlinden, and Westerman 2002, pp. 963–965.
111. Samuelson and Scotchmer 2002.
112. For some examples, see Gawer and Cusumano 2002 and Woodard 2008.
113. Langlois and Robertson 1992, pp. 302–306.
114. Christensen, Verlinden, and Westerman 2002, pp. 966–969.
115. Baldwin and Clark 2000.
116. Jacobides 2001, 2002, 2005.
117. Christensen, Verlinden, and Westerman 2002, pp. 971–972.
118. Fine 1998; Christensen, Verlinden, and Westerman 2002; Fixson and Park 2008.
119. For an overview and a discussion of existing explanations, see Fixson and Park 2008.
120. Baldwin and Clark 2000, pp. 105–106, 406, 414–417. For a formal theory of the “Silicon Valley model” and of the relationship between modularity and venture-capital financing, see Aoki 1999 and chapter 14 of Aoki 2001. For more on venture capital, see chapter 8 below.
121. See Gawer and Cusumano 2002, pp. 44–45.
122. See Baldwin and Clark 2000, pp. 267–269.
123. See *ibid.*, chapter 7.
124. For an in-depth analysis of this problem and a detailed study of the mechanisms Intel uses to encourage the adoption of new specifications by independent developers of complementary products, see Gawer 2000. See also Gawer and Cusumano 2002 and Gawer and Henderson 2007.
125. For an analysis of externalities in platform systems, see chapter 9 below.
126. Gawer and Henderson 2007. For a model illustrating these problems, see Farrell and Katz 2000.
127. See Shapiro and Varian 1999, pp. 237–259.
128. Gawer and Cusumano 2002, pp. 18–28. For an analysis of the strategies used by Intel, Microsoft, Cisco, and other platform leaders to overcome these problems, see Gawer and Cusumano 2002. For an analysis of the problems associated with standard setting and standard evolution when firms compete under conditions of compatibility, see Shapiro and Varian 1999, chapter 8.

129. See Garud and Kumaraswamy 1995, pp. 100–101. If a standards body or a consortium coordinates architectural changes, that organization's rules determine who can participate in any efforts to change the architecture. For an overview of the dimensions along which standard setting organizations may differ, see West 2007.
130. The factors discussed in the text are not the only factors to be considered as part of the decision. For a detailed analysis of the strategic choices that a system architect in a modular architecture faces, see Gawer and Cusumano 2002. See also Boudreau 2006.
131. See Novak and Stern 2003, pp. 8–9.
132. See Fixson and Park 2008.
133. See Baldwin and Clark 2001, p. 8.
134. For an overview of standards and the Internet, see Weiser 2001.
135. The IETF's standard-setting process is described in Bradner 1996.
136. See Comer 2000, pp. 216, 413.
137. See Computer Science and Telecommunications Board and National Research Council 2001, pp. 124–125.
138. See Baldwin and Clark 2001, p. 8.
139. See Kavassalis and Lehr 2000; Computer Science and Telecommunications Board and National Research Council 1996, pp. 83–85; Rao 1999.
140. Also see chapter 4 above.
141. On the strategy pursued by Microsoft, see Gawer and Cusumano 2002, pp. 131–162.
142. On the strategy pursued by Cisco, see *ibid.*, pp. 162–187.
143. Thus, the architecture of the Internet also enables modularity in use. On modularity in use, see box 2.1 above and p. 78 of Baldwin and Clark 2000.
144. See the section “Costs of Change in the Original Architecture of the Internet” in chapter 4 above.
145. See the subsection “The Broad Version of the End-to-End Arguments and Innovation at the Application Layer” in chapter 4 above.
146. See Isenberg 1998, p. 29; von Hippel 2007; Benkler 2002; Gaynor 2003, p. 40. In the Internet, the application programming interfaces through which applications access the Internet are publicly available for free or at low cost. This is an example of a non-architectural factor that interacts with the broad version of the end-to-end arguments to create a beneficial environment for application innovation in the Internet.

147. Lakhani and Wolf 2005, pp. 12–16. See also Benkler 2002. For analyses of the processes and mechanisms open-source projects use to coordinate their work, see Feller and Fitzgerald 2000; Yamauchi et al. 2000; Mockus, Fielding, and Herbsleb 2002; Scacchi 2002; Ulhøi 2004. For a short overview of literature on governance, organization, and innovation processes in open-source projects, see von Krogh and von Hippel 2006, pp. 979–980.

148. This type of innovation, in which users innovate to meet their own needs, is discussed in more detail in chapter 8 below.

149. Livingston 2007, pp. 127–130; Rapp 2006; Yahoo 2005. Since the traffic was slowing Stanford's network, Yan and Phyllo moved their servers to Netscape in early 1995 (Livingston 2007, p. 134).

150. Partridge 2008a.

151. Cohen 2002, pp. 3–4, 20–23, 29–30.

152. Cassidy 2006. Several people have claimed that Zuckerberg stole their idea. A dispute with three other Harvard students who had asked Zuckerberg to help them program a similar site in November 2003 was settled in court in June 2008. See Cassidy 2006; Markoff 2007; O'Brien 2007; Levenson 2008.

153. See also Kraus 2005a; Heinemeier Hansson 2005; Helft 2006; Buckman 2005.

154. Angel investors are individuals who invest capital in start-ups. The amount of money invested by angel investors is usually considerably smaller than that invested by venture capitalists; they are usually considered to be more flexible and less constraining than venture capitalists (Shah 2006; van Osnabrugge and Robinson 2000).

155. Spector 2000, pp. 62, 68, 70, 84–93.

156. Livingston 2007, pp. 260–261.

157. *Ibid.*, pp. 112–113.

158. Joel Spolsky follows the same approach with his company FogCreek Software (Spolsky 2003; Livingston 2007, pp. 112–113).

159. On eBay, see Cohen 2002, pp. 60–65. On Google, see chapter 8 below. On ICQ, see Brooker 1999; Weisman 2005. On Blogger, see Kahney 2001; Livingston 2007, pp. 117–120.

160. Christensen and Raynor 2003, p. 256; Gompers and Lerner 2001a, p. 136.

161. Spolsky 2003; Helft 2006. See also Lerner 2002; Scherer 2001.

162. Gompers and Lerner 2006, pp. 14–17 (on 2000); Miller, 2008a,b (on 2008).

163. Kraus 2005a.

164. On ICQ, see Brooker 1999; Weisman 2005. On Google, see Battelle 2005, pp. 85–87; Vise and Malseed 2005, pp. 45–49.

165. Jason Fried and David Heinemeier Hansson of 37signals and Joel Spolsky of Fog Creek Software are particularly vocal advocates of the latter approach (Spolsky 2003; 37signals 2006, chapter 2; Heinemeier Hansson 2008; Fried 2008b). (N.B.: In the references, 37signals is alphabetized as if spelled out: Thirty-Seven Signals.) Other founders who (at times) have decided against using venture capital include Evan Williams (Blogger/Twitter), Stewart Butterfield, and Caterina Fake (both Flickr). See Chafkin 2008, Helft 2006, and p. 261 of Livingston 2007. The benefits associated with venture-capital funding are discussed in chapter 8 below.

166. For example, in the fall of 2003, when the venture-capital firms Benchmark Capital and Kleiner Perkins Caufield and Byers invested \$13 million in Friendster, the first online social networking site, the site (which had been launched in March 2003) already had 3 million users and was widely viewed as “the next big thing.” But the company was not successful. See Rivlin 2006 and Hutheesing 2004.

167. On eBay, see Cohen 2002, pp. 55–59.

168. Spector 2000, pp. 96–103.

169. Battelle 2005, pp. 85–87, 89–91; Vise and Malseed 2005, pp. 45–49, 60–69; Google 1999.

170. Apart from Bloglines and Del.icio.us, which had been completely self-funded before these events, all these companies had taken some angel investments earlier in their history.

171. See the subsection of chapter 4 above titled “The Broad Version of the End-to-End Arguments and Innovation at the Application Layer.”

172. On the relevance of embeddedness within organizations, cultures, or network connections for innovation, see Saxenian 1994; Castilla et al. 2000; Saxenian 2002; Powell et al. 2002; Powell and Grodal 2005; Porter, Bunker Whittington, and Powell 2005.

173. The problems associated with this type of systemic innovation are discussed in the subsection of this chapter titled “Implications of the Link between Architecture and Feasible Governance Choices: Effect on Ability to Innovate.”

Chapter 6

1. Whether a reduction in independent developers’ incentives to innovate will be offset by the increase in network providers’ incentives to innovate will be discussed in chapter 8.

2. Though some technology was available that enabled network providers to detect and control some applications, this technology could not detect Internet-telephony

applications yet. It took a while to develop technology that was able to detect and block Internet-telephony applications. See Cherry 2005. Today, detecting Skype is still difficult. See Bonfiglio et al. 2007.

3. See chapter 2 above.

4. The IP protocol header contains a “protocol” field that indicates to which higher-layer protocol at the destination host the data should be delivered. In other words, the “protocol” field indicates the higher-layer protocol that invoked the services of IP at the originating host. Thus, if an application does not use a transport protocol but invokes IP directly, the IP header will carry information about the application. This field, however, is used solely for de-multiplexing at the destination host, and does not influence the treatment of datagrams within the network. The numbering of protocols used in the protocol field is standardized across the entire Internet. The relevant numbers are contained in an online database located at www.iana.org. See Tanenbaum 2003, p. 435.

5. The original specification of IPv4 provided an application with the possibility of specifying the desired type of service in a “service type” field. For example, the application could ask for low delay, high throughput, or high reliability. Routers could then use this information when choosing among various paths to a destination. Even if the routers made use of this information, the Internet protocol did not guarantee to provide any particular type of service owing to its best-effort service model. Thus, in an Internet running IPv4, routers may treat datagrams differently. The treatment depends on the content of the service type field, not on the type or particular instance of application. In practice, routers often ignore the type of service field. In the 1990s, the meaning of the field was redefined to accommodate differentiated services. See Comer 2000, pp. 99–101; Tanenbaum 2003, p. 434.

6. On deep packet inspection in general, see Anderson 2007. For a specific example, see Cisco Systems 2005. For an extended description of the techniques used to perform discrimination, see Peha 2007, pp. 646–650.

7. Technologies now being developed, including the IP Multimedia Subsystem and Next Generation Networks, are intended to change this. On these technologies, see the section in chapter 9 titled “Network Provider’s Private Interests in Network Architectures.”

8. The definition follows Carlton and Perloff 2005, p. 351.

9. As was noted in chapter 5, ‘complementor’ denotes an economic actor that provides complements. See Brandenburger and Nalebuff 1998, p. 16.

10. In the context of the framework developed in chapter 1, this is an example how an actor’s existing or expected economic relationships with other actors (here, the market structure in the market for Internet services) and other constraints (here, regulations) may influence an actor’s reaction to an architectural constraint.

11. See, e.g., Internet Non-Discrimination Act of 2006 (2006), §4(a); Internet Freedom and Nondiscrimination Act of 2006 (2006), §28(a); Internet Freedom Preservation Act (2007), §12(a)(1). Academic proponents of network neutrality include Lessig (2001), Wu (2003a), Wu and Lessig (2003), Weiser (2003), van Schewick (2007), Frischmann and van Schewick (2007), and Herman (2006). Academic opponents of network neutrality include Yoo (2005), Sidak (2006), Hahn and Wallsten (2006), and Faulhaber (2007). For a short introduction to the debate, see van Schewick 2009.

12. This is a common argument used by opponents of regulation. See, e.g., Speta 2000a, pp. 76–88; Speta 2000b, pp. 995–1007. In this case, regulation may still serve an educational function and protect customers and providers of independent content, portals, and applications from discriminatory or exclusionary conduct by “incompetent incumbents” (Farrell and Weiser 2003, p. 114) that fail to recognize that discrimination is not in their best economic interest. (See, e.g., Farrell and Weiser 2003, pp. 114–117; Wu 2003a, pp. 154–156.) But in comparison with a threat of discrimination due to a real incentive to discriminate, this constitutes a weaker basis for regulatory intervention.

13. See, e.g., Yoo 2004, p. 67; Hahn and Wallsten 2006, p. 6; Baumol et al. 2007, p. 3; Wu 2006a, p. 7.

14. For an example of this view, see Yoo 2004, p. 67: “On the other hand, regulators can adopt a more humble posture about their ability to distinguish anticompetitive from pro-competitive behavior and attempt to resolve the problem by promoting entry by alternative broadband platforms. Once a sufficient number of alternative last-mile providers exists, the danger of anticompetitive effects disappears, as any attempt to use an exclusivity arrangement to harm competition will simply induce consumers to obtain their services from another last-mile provider.” Also see Baumol et al. 2007, p. 3; Hahn and Wallsten 2006, p. 6.

15. An example of this line of reasoning can be found in the FCC memorandum and opinion in the AOL TimeWarner merger proceeding: “We believe that if unaffiliated ISPs receive non-discriminatory access to TimeWarner cable systems . . . the merged firm’s incentives and ability to withhold unaffiliated content from its subscribers will be substantially mitigated.” (Federal Communications Commission 2001, pp. 6594–6595, paragraph 107). See also Federal Communications Commission 2001, p. 6596, paragraph 112; Lemley and Lessig 1999 (reprinted as Lemley and Lessig 2000).

16. See Federal Communications Commission 2001, p. 6578, paragraph 74: “The relevant geographic markets for residential high-speed Internet access services are local. That is, a consumer’s choices are limited to those companies that offer high-speed Internet access services in his or her area, and the only way to obtain different choices is to move. While high-speed ISPs other than cable operators may offer service over different local areas (e.g., DSL or wireless) or may offer service over

much wider areas, even nationally (e.g., satellite), a consumer's choices are dictated by what is offered in his or her locality." See also Hausman, Sidak, and Singer 2001, p. 135: "From a consumer's perspective, the relevant geographic market is local because one can purchase broadband Internet access only from a local residence. Stated another way, a hypothetical monopoly supplier of broadband Internet access in a given geographic market could exercise market power without controlling the provision of broadband access in neighboring geographic markets."

17. Thus, the analysis assumes that the network provider may be vertically integrated into the provision of at least some applications. Vertical integration, however, is not the only case to which the analysis applies. A similar analysis applies to other forms of close vertical relationships between the network provider and a provider of complementary products such as partial integration, partial equity investments, long-term contracts, or other forms of close affiliation.

18. This clearly is an unrealistic assumption, as antitrust law usually includes some rules that restrict monopolization. Still, participants in the network-neutrality debate do not agree on whether the existing laws in antitrust and communications law cover all relevant threats of discrimination or not. For a flavor of the debate, see, e.g., Shelanski 2007; Yoo 2007; Hemphill 2008; Weiser 2008; van Schewick 2009. On the European situation, see Cave and Crocioni 2007 and Chirico, van der Haar, and Larouche 2007.

19. See the description of the architecture of the Internet in chapter 3.

20. The "open access" debate focuses on the question whether the owners of broadband networks should be required to allow independent ISPs to provide Internet access services over their networks. For a flavor of the debate, see Lemley and Lessig 1999, reprinted as Lemley and Lessig 2000; Bar et al. 2000; Speta 2000a,b; Yoo 2002.

21. See, e.g., Bork 1993, pp. 372–375; Posner 2001, pp. 198–199. In general, two goods are complements, if a decrease in the price of one increases the demand for the other (Varian 1999, p. 112). If the two goods are used in variable proportions, the monopolist may have an incentive to monopolize the complementary market, as this creates greater flexibility in its relative pricing of both components. Through appropriate pricing, the monopolist may be able to extract more surplus from consumers. If it needs a monopoly over both products to price discriminate in this fashion, monopolizing the second market will increase its profits. See, e.g., Ordovery, Sykes, and Willig 1985.

22. Farrell and Weiser 2003, p. 89.

23. See, e.g., Whinston 1990, pp. 840, 850–852; Farrell and Katz 2000; Farrell and Weiser 2003, p. 103.

24. See, e.g., Whinston 1990, pp. 850–852, 855.

25. This case is widely recognized as an exception to the “one monopoly rent” argument. See Bork 1993, p. 376; Ordovery, Sykes, and Willig 1985, pp. 121–127. In the context of the “open access” debate, this case has been highlighted by Farrell and Weiser (2003, pp. 105–107) and by Speta (2000b, pp. 997–1000). This theory has been used to explain why AT&T had an incentive to monopolize the market for telecommunications equipment in the past (Farrell and Weiser 2003, pp. 105–107; Speta 2000b, pp. 997–1000) and why the Bell operating companies may have incentives to engage in non-price discrimination against their long-distance rivals today, if they are allowed to participate in the long-distance market. For discussions of this question in the context of the American telecommunications market, see Sibley and Weisman 1998; Weisman 1998; Schwartz 2000; Mandy 2000; Weisman and Williams 2001.

26. On the factors that may lead a price-regulated upstream monopolist with a downstream subsidiary to engage in non-price discrimination against its downstream rivals, see Mandy 2000.

27. The theory laid out below, developed by Whinston (1990, pp. 854–855), is widely accepted as an exception to the “one monopoly rent” argument. See, e.g., Carlton 2001, pp. 667–668; Choi and Stefanadis 2001, p. 55; Whinston 2001, p. 71; Carlton and Waldman 2002, p. 195. For a detailed application of this theory in the context of the “open access” debate, see Rubinfeld and Singer 2001. See also Farrell and Weiser 2003, p. 119.

28. Whinston’s (1990) model does not consider the possibility that the primary-good monopolist may extract the complementary-good revenue for the “stand-alone market” through access charges. Assume, for a moment, that the primary-good monopolist can set access charges. If the producer of a complementary good has fixed costs, the monopolist could exclude it from the market by increasing the access charge. In this case, the complementary producer will be willing to pay an access fee less than or equal to its profits across both markets (i.e., the “systems market” and the “stand-alone market”) to stay in the market. However, exclusion may still be more profitable than extracting profits through access charges. First, even if the primary-good monopolist could completely extract its rivals’ profits across both parts of the market, the net gains from monopolizing the complementary market may be higher than the net gains resulting from an access charge, because of the presence of economies of scale or network effects in the complementary market, and the reduction in fixed costs under exclusion. In addition, if rivals know that their profits will be fully extracted, they may have fewer incentives to increase profits. Under such a scheme, the rivals’ incentives to maximize profits may also be lower. Second, it is likely that the primary-good monopolist will be unable to perfectly implement access charges. Since the monopolist does not know its rivals’ profit levels, it is difficult to set access charges. If the access charge is based on past profits, the firm may have an incentive to suppress profits or burn money in order to keep future access charges low (the “ratchet effect”—see, e.g., Weitzman 1980;

Freixas, Guesnerie, and Tirole 1985). This may make foreclosure preferable, as there is no information asymmetry.

29. Economies of scale exist if an increase in output causes long-run average total costs to decrease. In other words, the more output is produced, the lower the cost per unit. For example, economies of scale exist if fixed costs are large relative to marginal costs. In this case, an increase in output allows the firm to spread the fixed costs of production over greater amounts of output, lowering the costs per unit of output. See Hall and Lieberman 2001, pp. 177–178.

30. In markets subject to indirect network effects, this relates to product differentiation with respect to the primary good. Consumers' desire for variety in complementary markets leads them to prefer a larger network. If the benefits of product differentiation with respect to the network or primary good are strong enough, multiple incompatible networks may be able to coexist in equilibrium. Katz and Shapiro (1994, p. 106) write: "Consumer heterogeneity and product differentiation tend to limit tipping and sustain multiple networks. If the rival systems have distinct features sought by certain consumers, two or more systems may be able to survive by catering to consumers who care more about product attributes than network size. Here, market equilibrium with multiple incompatible products reflects the social value of variety."

31. Two technologies are incompatible if they cannot be used together. Thus, the users of the two technologies form two different networks. In a market with direct network effects, two technologies are incompatible, if consumers using one technology cannot interact with consumers using the other technology in the way that gives rise to the network effects in the first place. That is, users of one communication service will be unable to communicate with users of the other communication service. Similarly, users of one word-processing software will be unable to exchange documents with users of the other word-processing software. In a market with indirect network effects, two technologies are said to be incompatible, if the complementary products developed for one technology do not work with the primary product of the other technology, and vice versa. For example, users of the Microsoft Windows operating system will be unable to run application programs written for the Linux operating system, and vice versa. Users of Microsoft's media player will be unable to listen to music coded for RealNetworks RealPlayer, and vice versa. Compare Katz and Shapiro 1985, pp. 424–425.

32. Whether a firm wishes to make its products compatible with a competitor's technology is a strategic decision that results in very different forms of competition. If products are incompatible, firms compete for the standard. If products are compatible, firms compete within the standard. In general, owing to the huge gains from winning the competition between incompatible technologies (market dominance), a firm will tend to oppose compatibility, if it is confident to be the winner. In contrast, a firm that fears to lose in a competition for the standard will prefer compatibil-

ity and competition within the standard in order to avoid the huge costs of losing the competition for the standard. For an analysis of the factors influencing these choices and an overview of suitable strategies in the different forms of competition, see Besen and Farrell 1994; Katz and Shapiro 1985; Katz and Shapiro 1994; Shapiro and Varian 1999.

33. Often competitors will not be driven completely from the market. In particular, some customers with high switching costs or a unique preference for a competitor's product will prefer to stay with that competitor in spite of the strong network effects associated with the winning technology. See, e.g., Faulhaber 2002, p. 329, note 37.

34. "Tipping" occurs," according to Faulhaber (2002, p. 316), "when a single provider reaches a critical mass of customers that are so attractive to others that competitors must inevitably shrink, in the absence of interoperation."

35. In network markets, consumer expectations about the future size of the network play a crucial role in determining the outcome of the competition. This is due to the costs of belonging to the losing network: A consumer who has chosen the losing network can either switch to the winner, which may be costly, or suffer from the lower value of a small network. To avoid this situation, the consumer will choose the network that it expects to be the winner. See, e.g., Besen and Farrell 1994, p. 118.

36. A substantial lead in installed base is not the only factor that influences the outcome of the competition. Owing to the huge benefits of belonging to the winning network, users have a strong desire to choose the technology that will ultimately prevail. Therefore, consumers' expectations of who the winner will be are at least as important. Other factors that may influence customers' expectations and that may therefore result in a competitive advantage are an established reputation, a well-known brand name, or ready visible access to capital. Thus, an unknown firm with an early lead may be overtaken by a market leader that enters second, but has a well-known brand name and good reputation. See, e.g., Katz and Shapiro 1994, p. 107.

37. For example, if the rival produces a differentiated product, the rival's presence creates additional surplus, some of which the monopolist can extract through its sales of the primary good. Thus, the monopolist's profits in the "systems market" are increased if its rival is in the market.

38. See Whinston 1990, pp. 850–852, 855.

39. Owing to the cost structure of information products, the primary-good monopolist can profit from additional sales of the complementary information good, even if it does not have a monopoly in the complementary market. Thus, the network provider may have an incentive to exclude rivals in the complementary market, even if it cannot monopolize that market. See the subsection of this chapter titled "Profitability of Discrimination without Monopolization."

40. Carlton and Waldman 2002, pp. 206–207.
41. The market for broadband Internet access is considered a distinct market from the narrowband-access market, see, e.g., Hausman, Sidak, and Singer 2001, pp. 135–157; Federal Communications Commission 2001, pp. 6574–6577, paragraphs 68–73.
42. For residential high-speed Internet access service, the relevant market is local. See note 16 above. According to FCC data, 34% of ZIP codes have one or less cable or ADSL provider who serves at least one subscriber living within the ZIP code as of June 2007 (Federal Communications Commission 2008b, table 16, line “ADSL and/or Cable Modem”). As the United States Government Accountability Office (2006) has pointed out, this measure (i.e., the number of providers reporting at least one subscriber in a certain ZIP code) overstates the level of competition to individual households.
43. Breznick 2003; AOL 2008a,b.
44. Microsoft Online Services 2009.
45. Beardsley, Doman, and Edin (2003) report statistics showing that “so far, . . . faster and better access to the Internet is the sole killer application of broadband.” Thus, the scenario described in the text may be quite common. This possibility is also highlighted by Farrell and Weiser (2003, p. 119).
46. The marginal cost of production is the incremental cost of producing an additional unit of the good. Thus, the marginal cost of production does not include the costs of product development (see, e.g., Hall and Lieberman 2001, pp. 168–169). In the case of software applications, Internet content and portals, the marginal cost of production is the cost of making an additional digital copy of the product, which is typically very low.
47. The average cost of production indicates a firm’s total cost per unit of output. In other words, it denotes the total cost associated with a particular product divided by the quantity of output produced. Thus, contrary to the marginal cost of production which does not include the cost of developing the first unit of the product, the average cost of production includes the cost of development divided by the total number of copies. See, e.g., Hall and Lieberman 2001, p. 168.
48. This cost structure (low marginal costs relative to average costs), which results in significant economies of scale, is generally viewed as a key economic characteristic of the markets for these products. On information goods in general, see Shapiro and Varian 1999, pp. 3–4; on software, see Katz and Shapiro 1999, pp. 34–36; on Internet content, portals and software, see Posner 2001, pp. 245–246; on broadband portals, see MacKie-Mason 2000, p.14; on broadband content, see Rubinfeld and Singer 2001, p. 307.
49. The existence of direct or indirect network effects is a fundamental economic characteristic of many software markets. See, e.g., Evans and Schmalensee 2002, pp. 9–10; Katz and Shapiro 1999, pp. 32–34.

50. E.g., Faulhaber 2002.

51. E.g., MacKie-Mason 2000, p. 16.

52. See, e.g., Rubinfeld and Singer 2001, pp. 310–313.

53. For a numerical example, see Rubinfeld and Singer 2001, pp. 310–313. That paper assesses the likelihood of content discrimination (i.e., blocking or degrading the quality of outside content) by a broadband network provider that is vertically integrated into the market for broadband content and portals in the context of the merger between AOL and TimeWarner.

54. This theory is new and has not been covered by the existing literature.

55. If firms in the complementary market sell access to the users of their product to advertisers or other third parties they have two potential sources of revenue from two different sides of the market: the consumer side of the market, and the third-party side of the market. Such markets are known as two-sided markets; the good in the middle, here the complementary good, is known as a platform product. In a two-sided market, one or more platforms enable interactions between two different types of users, which form the two sides of the market, and the volume of interactions between users on both sides of the market depends on the structure of fees and not only on the overall level of fees charged by the platform (Rochet and Tirole 2006, p. 645). In our case, the complementary product enables interactions between consumers of the complementary product on the one side and advertisers on the other side; an interaction occurs when a consumer of the complementary product views an ad. Platform products that enable interactions between consumers of the platform good and advertisers are a special case of two-sided markets: In most two-sided markets, the two sides of the market want to interact with each other, resulting in positive network effects between the two sides. For example, the users of a video game console (the platform product) would like to play video games written for the console, and prefer a console that is compatible with more video games. Video game developers, the other side of the market, want to sell their video games to users of the console and prefer to write video games for a console that has more users. In contrast, the benefits that end users receive from the platform product may or may not increase with the amount of advertisers associated with the platform product. For example, consumers of yellow pages have been shown to prefer yellow pages with more advertising (Rysman 2004, p. 508). Researchers often assume that television viewers prefer to watch stations with less advertising (Anderson and Coate 2005, p. 951). However, this depends on the situation—more than a quarter of viewers of the Super Bowl tune in for the ads (Markillie 2005, p. 3). The more easily consumers can avoid advertising in a particular medium, the more likely that the ads provided in that medium are utility-enhancing, because otherwise consumers would simply choose to ignore the ads (Becker and Murphy 1993, p. 962). On the Internet, advertisers clearly benefit from additional consumers using the platform, but end users may or may not benefit from an increased number of advertisers.

56. The conditions under which this pricing structure will arise are explored by the literature on two-sided markets (Armstrong 2006; Rochet and Tirole 2003; Rochet and Tirole 2006).

57. A common set of tactics forces rival producers of the complementary good to lower the quality-adjusted price of their product (e.g., Farrell and Katz 2000, pp. 414–415). This increases the consumer surplus available for extraction in the primary market. In another tactic, the monopolist imposes an access charge on rivals who wish to participate in the complementary market (*ibid.*, p. 422). Of course, the monopolist must actually have the power to do so because of intellectual-property rights or because rivals' access to the primary good requires the monopolist's cooperation.

58. This may be an unrealistic assumption. See note 28 above.

59. More formally, assume that the monopolist's (M 's) complementary product has x customers and a rival's (R 's) complementary product (before the exclusion) has y customers. Let $f_M(u)$ and $f_R(u)$ represent the advertising revenues that the monopolist and the rival can realize if they sell access to u customers of their respected version of the complementary good. Finally, assume that after exclusion z customers demand the monopolist's complementary product. If $f_M(z) > f_M(x) + f_R(y)$, the monopolist will generate more advertising revenue by excluding its rival from the complementary market and selling access to all users of the complementary good directly than by having its rivals stay in the market and extracting the advertising revenue from them, even if it is able to extract all such revenues.

60. Monopolizing the market for the complementary good directly monopolizes sales to the end-user side of the market. As access to end users is monopolized, the primary-good monopolist becomes the only supplier of access to these customers through the complementary good and the advertiser side of the market is also monopolized. Of course, advertisers may find other ways to market their offerings to users of the primary good, for example through other media, so the exclusion of rivals from the complementary market will not necessarily confer market power over advertisers on the primary-good monopolist and the primary-good monopolist may not be able to raise advertising prices. Whether market power is conferred or not will depend on the specific circumstances of the advertising market and the characteristics of the consumers of the complementary good.

61. With multiple auctions the third parties may have been able to bid in different auctions, thereby reducing competition in any given auction.

62. Ultimately, higher prices on the third-party side of the market will harm consumers, as firms will pass on at least some of the increased costs to their customers. For example, higher advertising fees will ultimately lead to higher prices for the goods that are advertised. See Rubinfeld and Singer 2001, p. 316; MacKie-Mason 2000, p. 23.

63. See, e.g., Shapiro and Varian 1999, pp. 34–35.

64. Theoretically, the monopolist could always increase the value of complementary market by sharing this information with all the firms in the complementary market (so that they could share it with their advertisers and increase their advertising fees) and then extract these additional rents. However, data protection may prevent this information being shared on a disaggregated basis or the primary-good monopolist may consider such information a valuable business secret that it does not wish to disclose.

65. Under the analysis above, foreclosure may be welfare enhancing or destroying. Generally, the reduction of fixed costs and of costs associated with access charge administration and negotiation under foreclosure will increase welfare. The increased value of advertising space from network effects and additional information provided by the monopolist will also generally increase welfare. However, the increased rents that can be extracted through increased monopoly power are likely to reduce welfare. As prices on the third-party side of the market increase, third parties are likely to demand less access to the complementary market's customers and this may reduce welfare in the standard way that a monopoly reduces welfare. As a side effect of foreclosure, welfare is likely to be reduced if the complementary goods market was differentiated. In this case, the reduction in choice is likely to reduce consumer surplus which directly reduces welfare. In addition, this may feed through to lower demand in the complementary market, also reducing producer surplus (in the notation of note 59 above, $z < x + y$). Overall, the static welfare effects of foreclosure are likely to be ambiguous. This book's argument, however, is based on the dynamic welfare effects of foreclosure, in particular the reduction in innovation in complementary markets. See the discussion in chapter 9.

66. Theoretically, the primary-good monopolist may be able to set an access charge that extracts its rivals' advertising profits across the "stand-alone market" and the "systems market." Whether access charges are an option at all, depends on the markets under analysis. Even if they are possible, exclusion may enable the monopolist to realize higher profits than access charges. First, we just saw that the total profits available in the complementary market may be higher under foreclosure than under access charges. Second, access charges may distort incentives for rivals in the complementary market. Third, the monopolist may be unable to extract the full revenue because of information asymmetry about profit levels. See note 28 above.

67. Afuah and Tucci 2001, p. 56; Shapiro and Varian 1999, pp. 162–163.

68. Google 2008, p. 39.

69. Time Warner 2008, p.1; Yahoo 2008b, p. 35.

70. See Schiesel 2002 with respect to a contract between AOL and AT&T Comcast.

71. See, e.g., Rubinfeld and Singer 2001, p. 316; MacKie-Mason 2000, p. 23.

72. Rashtchy et al. 2007, p. 117.

73. Holahan 2006.

74. By excluding competing providers of the complementary product from access to its Internet-service customers, the network provider monopolizes the customer side of the market and becomes the exclusive provider of access to its Internet-service customers through the complementary good. However, this will only translate into an ability to raise prices on the advertiser side of the market when advertisers cannot get access to equivalent end users at the current price through another medium. See note 60 above.

75. For example, comScore, a market research company, estimated Google's market share of the US search-engine market in February 2007 at 59%, Yahoo's at 22%, and Microsoft's at 10% (comScore 2008).

76. For example, SearchIgnite, a firm that offers technology that helps advertisers and advertising agencies manage paid search campaigns, reported that revenue per search for ads by all marketers in June 2007 was \$21 for Google, \$15 for Microsoft, and \$9 for Yahoo. Revenue per search for ads by large brand marketers in June 2007 was \$25 for Google, \$21 for Microsoft, and \$16 for Yahoo. (SearchIgnite and RBC Capital Markets 2007, pp. 3–4.) According to SearchIgnite (*ibid.*, p. 3), “revenue per search (RPS) takes into account how well [search] engines generate revenue from search ad impressions—specifically, factoring in search ads' cost-per-click (CPC) and click-through-rate (CTR) to measure the revenue per every 1,000 ad impressions served.”

77. Varian 2006, p. 178.

78. *Ibid.*, pp. 186–188; Lohr 2008; Holahan 2007; Rashtchy et al. 2007, pp. 169, 174.

79. Of course, for the higher number of advertisers to create larger value to users, the search engine needs to be able to identify and display those ads that are most relevant to the users.

80. In early 2007, Yahoo introduced a similar system called Panama in order to increase the revenue from search. While this system has increased the click-through rate of ads on Yahoo's search engine, Yahoo said in a presentation to investors in March 2008 that Panama had improved Yahoo's revenue per search for searches in the United States by 30% in the first nine months of 2007, but that it still earned 60–70 percent less for searches than Google (Yahoo 2008a, Exhibit 99.2, p. 14). See also Rashtchy et al. 2007, pp. 169, 174, 195–196; Holahan 2007.

81. This assumes that the users' demographic characteristics are similar across search engines. If users' demographic characteristics differed across search-engine providers and advertisers were willing to pay more for access to Google's users because they value the particular demographic characteristics of Google's users, having the users

of other search engines use Google may make Google's demographics worse and may reduce Google's revenue per search. It also assumes that a sufficient number of users that had used Microsoft's or Yahoo's search engine before the exclusion would use Google's search engine after the exclusion (instead of stopping to use search engines altogether).

82. Hansell 2008b.

83. Unless otherwise noted, the analysis in this paragraph is based on Hansell 2008a, Gapper 2008, and Evans and Noel 2007.

84. Rashtchy et al. 2007, pp. 276, 241–242. The gap between advertising revenue may decrease in the future with the introduction of behavioral targeting methods by advertising networks. These methods would enable advertising networks to track users' behavior across websites, enabling them to sell highly targeted ads. Still, the website on which the ad is ultimately placed will most likely get only a share of the advertising revenue that the advertiser pays to the ad network. For example, Tacoda, an advertising-network company using behavioral marketing that was acquired by AOL in 2007, keeps 40% of the revenue from a targeted ad; 40% go to the site on which the ad is ultimately placed, and 20% go to the site that enabled Tacoda to identify specific interests of a user (Holahan 2006). Thus, this still makes it likely that the advertising revenue that a network provider could extract from an independent website will be lower than the advertising revenue that the network provider can realize if the users visit its own website instead of the independent website (assuming the network provider's website is part of a larger conglomerate).

85. Yahoo has an ad network, partnering in a revenue-sharing model with AT&T in the United States, BT in the United Kingdom, and Rogers Cable in Canada; in addition, it sells co-branded Internet services with a number of other providers (Yahoo 2008b, p. 12). For details of such a deal, see Kaye 2008.

86. Holahan 2006.

87. Even if those rivals require consumers to register before using their product or service, they have no way to verify the information, unless they require payment; in this case, they can verify the information as part of the billing process. See Shapiro and Varian 1999, pp. 34–35; MacKie-Mason 2000, p. 11. Theoretically, the network provider could share this information with providers of complementary goods. This would enable them to charge higher advertising fees which the network provider could extract through access charges. However, data protection may prevent this information being shared on a disaggregated basis or the network provider may consider such information a valuable business secret that it does not wish to disclose.

88. For example, in the notation of note 59, if there is one complementary good rival to a vertically integrated monopolist before exclusion, the net gains in the complementary market are positive if $f_M(z) - c_M(z) > f_M(x) - c_M(x) + f_R(y) - c_R(y)$, with $c_M(u)$ representing the monopolist's fixed and variable costs of serving u

complementary good customers and $c_r(u)$ representing the rival's fixed and variable costs of serving u complementary good customers. To make exclusion profitable overall, the net gains in the complementary market must be larger than the reduction in profit in the Internet-services market.

89. If the costs were sunk (i.e., non-recoverable) instead of fixed, existing providers would stay in the market as long as the network provider would let them keep enough revenue to cover their variable costs. However, future applications and content providers would not make the start-up investment, because they would anticipate it being appropriated by the network provider. Thus, the network provider would have to let existing providers keep their sunk costs in order to motivate future applications providers to enter the market.

90. See the references cited in note 48 above. On economies of scale in search engines, see Varian 2006.

91. While the reduction in fixed costs and transaction costs under foreclosure may be welfare enhancing, exclusion may not be welfare enhancing overall. See note 65 above.

92. Thanks to Robert Pepper for highlighting this example.

93. See Nuechterlein and Weiser 2005, pp. 49–50, 53–54, 195, 204, 294.

94. The access charge is lost if the call is placed using VoIP, regardless of whether VoIP is provided by the network provider or by an independent provider. Thus, the network provider has an incentive not to have VoIP used on its network at all.

95. In theory, network providers may be able to realize the same or higher profits as under blocking by setting an access charge that all providers of Internet-telephony applications must pay. However, the costs of setting up a system for these access charges and the potential political fallout may make it preferable to simply block (or secretly slow) the Internet-telephony applications and continue to realize regulated access charges through the existing system. In addition, for reasons discussed in note 28 above, a network provider may prefer to realize profits himself over extracting them from others via access charges.

96. See Federal Communications Commission 2005b,c; Charny 2005b; McCullagh 2005; Madison River Communications 2006.

97. On international settlement charges and the US regulatory regime governing these charges, see, e.g., Federal Communications Commission 2007; Einhorn 2003; Stanley 2000.

98. Sources: Cherry 2005 (Saudi Telecom); Charny 2005a (Mexico); PriMetrica 2008 (United Arab Emirates).

99. See, e.g., Salop and Romaine 1999, pp. 625–626; Carlton and Waldman 2002, p. 195.

100. On this type of monopoly maintenance in general, see, e.g., Carlton 2001, pp. 668–671; Farrell and Weiser 2003, pp. 109–112; Salop and Romaine 1999, pp. 625–626. For specific models, see, e.g., Carlton and Waldman 2002; Choi and Stefanadis 2001.

101. See, e.g., Carlton 2001, pp. 669–670.

102. E.g., U.S. Department of Justice and Federal Trade Commission 1984, §4.212 (promulgated in 1984 and reaffirmed in 1992 and 1997) (“The relevant question is whether the need for simultaneous entry to the secondary market gives rise to a substantial incremental difficulty as compared to entry into the primary market alone. If the entry at the secondary level is easy in absolute terms, the requirement of simultaneous entry to that market is unlikely adversely to affect entry to the primary market.”).

103. For an argument along these lines, see Williamson 1979, pp. 962–964; U.S. Department of Justice and Federal Trade Commission 1984, §4.212.

104. U.S. Department of Justice and Federal Trade Commission 1984, §4.212.

105. Carlton and Gertner 2003; Choi and Stefanadis 2001.

106. For a definition of indirect network effects, see box 6.4.

107. Gilbert and Katz 2001, p. 30 (referring to operating systems and application programs). Under the label “applications barrier to entry,” this line of reasoning has featured prominently in the Microsoft case. See, e.g., *United States v. Microsoft*, 253 F.3d 34, 54–56 (D. C. Cir. 2001); Gilbert and Katz 2001, pp. 28–30.

108. In addition to offering its own content and applications to the customers of its Internet service, the monopolist may also “allow” independent producers of these products to offer their products to the customers of its Internet service, as long as they agree to offer their products exclusively to these customers. Stated differently, instead of depriving a potential entrant into the market for Internet services of a source of complementary products by driving rival content and application producers from the market, the monopolist could deprive the potential entrant of a source of complementary products by signing exclusive contracts with independent content and application producers. Whether a monopolist could profitably impose such an exclusivity provision, has been the subject of considerable debate. The Chicago school denied such a possibility, arguing that the other party to the exclusive contract would not agree to contracts that made it worse off. (See, e.g., Bork 1993, p. 309.) More recent research has shown that this argument is incomplete: it does not consider the possibility that the exclusive contract imposes harm on third parties that are not parties to the contract, while not making the contracting parties worse off. In other words, the exclusive contract gives rise to a negative externality on third parties, and because of this externality signing an exclusive contract is jointly optimal for the contracting parties. For discussions of this question with pointers to the literature, see Gilbert and Katz 2001, pp. 31–33; Whinston 2001, pp. 66–70.

109. Usually, this theory is applied to cases in which the entrant's primary good is technically unable to take advantage of the set of applications developed for the monopolist's primary good. For example, software applications make use of a specific operating system's application programming interfaces and therefore run only on this operating system. As a result, customers of the entrant's operating system are technically unable to use applications developed for the incumbent's operating system. In contrast, as long as an application complies with the specifications of the Internet protocol, it can run over any physical network that supports the Internet protocol. As a result, applications adhering to that standard can be used by anyone connected to the Internet. Thus, from a technical point of view, the applications offered by the monopolist could be used by customers of a rival network provider as well. Therefore, the entrant's inability to use the monopolist's applications and content is not due to technical differences or incompatibility between the Internet services offered by the monopolist and a potential entrant, but results from the monopolist's business decision to offer its content and applications exclusively to customers of its own Internet service.

110. The strategy described here requires that the monopolist does not offer the content, application or portal to consumers outside its network; in contrast, in the "primary good not essential" strategy, the inability to earn monopoly profits on its sales to consumers outside its network is the reason that leads the monopolist to monopolize the complementary market as well. See the subsection titled "Primary Good Not Essential."

111. The potential anti-competitive implications of such a strategy are explored by, e.g., MacKie-Mason 2000, pp. 23–25; Rubinfeld and Singer 2001, pp. 313–316.

112. Alternatively, the monopolist could reach the same result by allowing independent producers of applications, content, and portals to offer their products to the customers of its Internet service, if they agree to provide the products exclusively to its customers. See note 108 above.

113. For example, compared to PCs, mobile handsets have small screens, limited keypads, and not a lot of storage. See, e.g., Deprez, Rosengren, and Soman 2002, p. 97, note 4.

114. Finally, one may imagine a situation in which the nationwide market for Internet services consists of a collection of local monopolies who all bundle their content, portal, and applications exclusively with their Internet service. In this case, a new entrant into the market for Internet services would have to enter the market for content, portals, or applications as well.

115. See, e.g., Niewijk, Songhurst, and Todd 2003, section "Moving towards partnerships."

116. That the costs of capital may increase with the amount of entry costs that are sunk is discussed by Viscusi, Harrington, and Vernon (2005, pp. 169–170, 173).

117. Deprez, Rosengren, and Soman 2002, pp. 96–97.

118. For example, as of March 2008, the Top 19 cable and phone broadband providers account for 94% of the US market; the seven smallest of these providers had between 227,900 (Cincinnati Bell) and 911,000 (Windstream) subscribers (Leichtman Research Group 2008).

119. E.g., Speta 2000a, pp. 83–84.

120. See the subsection of this chapter titled “Primary Good Not Essential.”

121. See box 6.4.

122. An ISP could reach the same effect (i.e., reintroduce indirect network effects with respect to its own network) by using proprietary protocols inside its network, see, e.g., Computer Science and Telecommunications Board and National Research Council 2001, pp. 147–149. An alternative strategy may be the provision of quality of service only within an Internet provider’s network, see, e.g., Shapiro and Varian 1999, p. 187.

123. On a similar situation in the context of provider-specific indirect network effects due to the use of proprietary protocols inside the network, see Computer Science and Telecommunications Board and National Research Council 2001, pp. 147–149.

124. As was highlighted in note 108 above, an alternative way of deterring entry would be to sign exclusive contracts with independent producers of applications, content, and portals. Such a strategy would have the advantage that the monopolist does not have to bear losses with respect to its Internet-service fees, as its customers would have access to all existing applications, content, and portals.

125. This theory has not been used as an exception to the “one monopoly rent” argument before. It generalizes from an argument that was used by the Federal Communications Commission in the AOL/TimeWarner merger proceeding with regard to instant messaging (Federal Communications Commission 2001, pp. 6603–6629, paragraphs 128–200) and subsequently analyzed by Faulhaber (2002).

126. See the subsection of this chapter titled “Primary Good Not Essential.”

127. On dynamic or “Schumpeterian” competition, see, e.g., Evans and Schmalensee 2002; Shelanski and Sidak 2001, pp. 10–15; Carlton and Gertner 2003, pp. 45–48.

128. See e.g., Evans and Schmalensee 2002, pp. 3–13.

129. Federal Communications Commission 2001, pp. 6603–6629, paragraphs 128–200. For an in-depth analysis of the economic rationale underlying this condition, see Faulhaber 2002.

130. The fifth exception, “monopoly preservation in the primary market,” requires that rival producers of excluded complementary products are driven from the market.

131. For a similar argument in the context of a general analysis of the profitability of exclusionary conduct, see Elhauge and Geradin 2007, pp. 498–499, 548–549.

132. For an economic model demonstrating this effect (i.e., if marginal profits in the complementary market are positive, a primary-good monopolist or duopolist may have an incentive to engage in exclusionary conduct in the complementary market if this increases its number of sales in this market, even if the conduct does not monopolize the complementary market) in the context of tying, see DeGraba 1996. In DeGraba's model, oligopolists sell a differentiated good (the primary good) and a homogenous good (the complementary good) that are used in fixed proportions to produce the final good. The homogenous good can be produced at constant marginal cost by any firm incurring a certain fixed cost. The homogenous market is characterized by free-entry Cournot competition. In such a market, the zero-profit price of the good is greater than the marginal cost, since each firm has to make a positive profit on each sale to cover the fixed costs of production. As a result, the oligopolist in DeGraba's model will tie in order to increase the sales of the complementary good, which increases profits (although the tying does not monopolize the complementary market). Note that this model does not require the complementary good to be a differentiated good. In deGraba's model, tying increases welfare by reducing competition in the complementary market, which reduces the wasteful replication of fixed cost expenditures. As products in the downstream market are homogeneous, there is no scope in this model for the reduction in choice harming consumers. Were products modeled as heterogeneous in the downstream market then it is likely that the static welfare effects would be ambiguous. Moreover, deGraba does not consider the dynamic effects on innovation, which are the focus of this book. See the discussion in chapter 9 below.

133. If the price were equal to marginal costs, firms would not be able to cover their fixed costs and would earn negative profits. In the long run, firms would not operate in such a market. Thus, even if all firms earn zero profit per unit in long-run equilibrium, equilibrium prices are above marginal costs.

134. Shapiro and Varian 1999, p. 161. The importance of market share and number of units sold in knowledge-based products is also described by Afuah and Tucci (2001, pp. 52–54). For an economic model demonstrating this effect in the context of tying, see DeGraba 1996. For a discussion of DeGraba's model, see note 132 above.

135. Katz and Shapiro 1999, p. 41.

136. See the subsection of this chapter titled "Primary Good Not Essential."

137. As outlined in the subsection "Primary Good Not Essential," the conditions are as follows: The network provider has a monopoly in the primary market (i.e., the market for Internet services). The primary good is not essential (i.e., there are uses of the complementary product that do not require the primary good). This condition is met when the ISP offers its complementary product not only to its

Internet-service customers, but to customers nationwide. The complementary market is subject to economies of scale or network effects, a condition that is met in most markets for applications, content, or portals. The monopolist has a mechanism at its disposal that enables it to exclude its rivals from access to its primary-good customers. In the Internet context, technology that enables the network provider to distinguish between applications running over its network and to control their execution provides the network provider with this capability.

138. See the subsection of this chapter titled “Complementary Product Source of Outside Revenue.”

139. Even if those rivals require consumers to register before using their product or service, they have no way to verify the information, unless they require payment; in this case, they can verify the information as part of the billing process (Shapiro and Varian 1999, pp. 34–35; MacKie-Mason 2000, p. 11).

140. Access charges may not be similarly effective. (See notes 28 and 66 above.) This is particularly likely if, as discussed in the text, the amount of advertising revenue in the complementary market is larger if the monopolist realizes the advertising revenue itself rather than through access charges.

141. See the subsection of this chapter titled “Complementary Product Source of Outside Revenue.”

142. See the subsection of this chapter titled “Monopolist’s Complementary Product Source of Outside Revenue.”

143. See the subsection of this chapter titled “Monopoly Preservation in the Complementary Market.”

144. For a similar argument with respect to the profitability of monopoly preservation through exclusionary conduct in new economy markets, if the monopoly is of intellectual property, see Posner 2001, p. 254.

145. Under tying, a sale of a product is conditioned on the purchase of another (Carlton and Perloff 2005, pp. 675–677; Posner 2001, pp. 197–207). Under exclusive dealing, a firm forbids the firm it supplies to sell products of competing firms (Carlton and Perloff 2005, pp. 672–673; Posner 2001, pp. 229–232, 251–256). In a vertical merger, a firm merges with a firm belonging to another stage in the production or distribution chain (Carlton and Perloff 2005, pp. 669; Posner 2001, pp. 223–229).

146. E.g., Posner 2001, p. 195; Yoo 2002, p. 188. Similarly, some sort of market power or political power is considered to be a prerequisite for strategies that raise rivals costs; see, e.g., Carlton and Perloff 2005, p. 371.

147. For an important exception to this point, see the literature on the exercise of aftermarket power by a firm that faces competition in the foremarket. This literature focuses on the question whether primary market competition precludes

anti-competitive aftermarket actions. For an analysis of these issues with pointers to the literature, see MacKie-Mason and Metzler 1999.

148. See, e.g., Yoo 2002, pp. 249–250, 253, in the context of the “open access” debate (“I conclude that the structure of the broadband industry renders it unlikely that such combinations will pose any significant anti-competitive threat”), and Yoo 2005, p. 61, in the context of the network-neutrality debate (“This suggests that for most of the country, competition should remain sufficiently robust to ameliorate concerns of anticompetitive effects”).

149. E.g., Speta 2000b, p. 986, in the context of the “open access” debate.

150. See the references cited in note 13 above.

151. The analysis assumes that the network provider competes with at least one other network provider; thus, it is not restricted to the duopoly situation. Farrell (2006) calls this market structure “duopoly +/-.”

152. See also Farrell (2006). Farrell argues that limited competition may not necessarily remove network providers’ incentives to discriminate. For a similar argument in the context of the debate over censorship by private proxies, see Kreimer 2006, pp. 33–36. Kreimer argues that competition between ISPs may not be sufficient to discipline ISPs that disable content needlessly on the basis of arguments very similar to the ones advanced above.

153. There may be other reasons that justify these proposals, though. For example, according to Lemley and Lessig (1999, pp. 21–25) the reduction in application-level innovation by independent providers resulting from the threat of discrimination constitutes only one of three arguments in favor of open access.

154. This assumption reflects the reality in the broadband market for residential customers in the US. According to a recent study by the United States Government Accountability Office (2006, p. 18), the median number of broadband providers available to residential users is two.

155. See subsection “Primary Good Not Essential.”

156. But even if the network provider does not manage to drive complementary goods producers from the market completely, it may still have an incentive to discriminate. See subsection “Profitability of Discrimination without Monopolization.”

157. See subsection “Profitability of Discrimination without Monopolization.”

158. See subsection “Profitability of Discrimination without Monopolization.”

159. See subsection “Profitability of Discrimination without Monopolization: More Sales at Market Prices.”

160. DeGraba’s (1996) model, which demonstrates this effect in the context of tying, supports this analysis. In the model, the producer of the primary good has

an incentive to tie in order to increase the number of sales of the secondary good, although it competes with another producer in the primary market. Thus, in the model the incentive to exclude independent competitors from the secondary market is not dependent on a monopoly position in the primary market. The model is discussed in note 132 above.

161. See subsections “Complementary Product Source of Outside Revenue” and “More Outside Revenue.”

162. Federal Communications Commission 2001, p. 6594, paragraph 106; pp. 6593–6594, paragraphs 104–106.

163. See subsections “Monopolist’s Complementary Product Source of Outside Revenue” and “More Outside Revenue.”

164. See subsection “Monopoly Preservation in the Complementary Market.”

165. See the references in note 13 above.

166. Wu (2003a, p. 153) makes a similar point in his discussion of the costs of a discriminatory pricing scheme that prohibits customers of a network provider’s basic Internet service from using specific applications.

167. See, e.g., Rubinfeld and Singer 2001, p. 310.

168. See *ibid.*, p. 310.

169. See the references in note 13 above.

170. See, e.g., Rubinfeld and Singer 2001, pp. 312–313.

171. See, e.g., Rubinfeld and Singer 2001, pp. 310, 313.

172. See, e.g., Craswell 1982; Bar-Gill 2006.

173. At the end of March 2008, Comcast provided about 40% of cable broadband Internet and 20% of all broadband Internet in the US (Leichtman Research Group 2008).

174. Eckersley, von Lohmann, and Schoen 2007; Topolski 2008. Comcast stopped using this method of interfering with peer-to-peer file-sharing applications in 2008.

175. Reardon 2007; Schoen 2007a.

176. Sandvine 2004, p. 14.

177. Mennecke 2005; Hellweg 2003; Sandvine 2004, pp. 5–6. See also National Cable and Telecommunications Association 2007, p. 31 (“cable operators will not go down the path of blocking access to video or P2P services. Blocking such services would be a recipe for . . . massive dissatisfaction among consumers, which would lead to loss of customers to our competitors.”)

178. See Hausman, Sidak, and Singer 2001, p. 164; Kreimer 2006, pp. 34–35; Nuechterlein and Weiser 2005, p. 156.

179. For an in-depth overview of the economic literature on switching costs, see Farrell and Klemperer 2007. For a short overview, see Varian 1999, pp. 603–605. For a treatment of switching costs in the context of information goods, see Shapiro and Varian 1999, chapters 5 and 6.

180. E.g., Varian 1999, pp. 604–605; Hausman, Sidak, and Singer 2001, p. 164.

181. Switching costs do not protect the network provider from losing business from new customers.

182. For example, HearUsNow.org, a project of the Consumers Union, found that a number of the top broadband providers in the United States charge early-termination fees. For example, at the time of the survey in March 2007, Qwest charged a \$200 early-termination fee on a two-year contract for high-speed Internet service, Earthlink charged a \$149 early-termination fee on a one-year contract for DSL service, and AT&T (including SBC and BellSouth) charged a \$99 early-termination fee (Consumers Union 2007; Dunbar 2007).

183. The customer may switch his whole bundle to the new provider, but that creates other problems, for example by making the decision to switch more complex, or by resulting in the loss of the preferred service offering for example in television or telephony.

184. Providers have considerable influence over this cost. For example, in 2005, AOL paid \$1.25 million in fines as part of a settlement with the state of New York, because AOL's customer service representatives were incentivized to dissuade customers from switching away from America Online, "by either making the cancellation process so painful for the customers that they could not bear to continue, or by simply ignoring their requests" (Stross 2005).

185. On the use of provider-specific e-mail addresses as a way to increase switching costs in Internet services, see Shapiro and Varian 1999, pp. 109–110. In other telecommunications markets such as wireline telephony and mobile telephony, regulation often requires providers to provide number portability, i.e., to enable a customer to keep its phone number when he switches providers. The Federal Communications Commission currently has to consider a petition to require e-mail providers to forward e-mail to a new e-mail address for a limited time (Mortenson 2007).

186. Stross 2005.

187. For example, an empirical study of competition between cable television and direct broadcast satellite (DBS) multi-channel services showed that while customers generally tend to switch from cable to DBS when the quality-adjusted price of cable increases substantially, the availability of regional sports channels reduced DBS penetration, either because it raised consumers' switching costs or because it

increased product differentiation between the two types of services (Wise and Duwadi 2005, pp. 695, 699–700).

188. DW Staff 2006.

189. This is a hypothetical example. At the time of this writing, Deutsche Telekom offers customers of other ISPs the opportunity to subscribe to the soccer content for a monthly fee (Deutsche Telekom 2008).

190. That product differentiation may provide sellers with some degree of market power is well established in the literature (Carlton and Perloff 2005, pp. 203–205).

191. Ranaweera and Prabhu 2003.

192. Burnham, Frels, and Mahajan 2003.

193. Status quo bias seems to result from a number of factors. For example, contrary to rational-choice theory, consumers often take past sunk costs into account when making consumption decisions (Samuelson and Zeckhauser 1988, pp. 37–38). Choosing one option and rejecting the other also creates cognitive dissonance, which is reduced by subsequent rationalization that the chosen option is more desirable than it was *ex ante* (Brehm 1956, p. 389). Finally, people tend to regret bad outcomes that are a result of their own action more than bad outcomes that are the result of their inaction, which again leads to a bias for doing nothing (Kahneman and Tversky 1984, pp. 343–344).

194. Trial subscriptions with a low introductory price that automatically convert to a higher price, or other contracts with automatic renewal also exploit the cognitive bias that people tend to overestimate their future willingness to incur the then immediate costs of switching (or terminating the contract) in order to reap the future benefits (i.e., the savings) resulting from switching (or terminating the contract) (DellaVigna and Malmendier 2004, pp. 381–393).

195. In addition, open-access regulations (regulations that require the owners of broadband networks to allow independent ISPs to offer their services over these networks) usually do not protect against cases in which the provider of the underlying physical network infrastructure discriminates against a specific application in all traffic that is running over its network, independent of which ISP the traffic belongs to. This is because open-access regulations usually only require the network provider to treat ISPs in a non-discriminatory way, not to treat applications and content in a non-discriminatory way. A recent case in Canada exemplifies this concern. Bell Canada slowed down all peer-to-peer file-sharing traffic on its network, regardless of which ISP the traffic belonged to. In November 2008, the Canadian regulator subsequently denied an application by the Canadian Association of Internet Providers to order Bell Canada to cease and desist this practice (Anderson 2008a,b).

196. Yoo (2006) argued that singling out specific applications or classes of applications and blocking them is a socially efficient way to manage congestion. Brett

Frischmann and I have taken up this argument (Frischmann and van Schewick 2007). As we show, not only is Yoo's theory based on the false assumption that it is currently impossible to meter subscribers' Internet use; Yoo also neglects the significant social costs associated with the blocking of specific applications to manage congestion.

197. In DSL networks, end users do not share the link between the DSL modem and the Digital Subscriber Line Access Multiplexer (DSLAM), but they share the link from the DSLAM which aggregates data traffic from a number of subscribers onto a single high-speed data link to the ISP. See, e.g., Mervana and Le 2001, pp. 189–197.

198. This is obvious if the network provider has a transit agreement with the other network; in a transit agreement, the smaller network pays a price for data transport that is based on use. Under a peering agreement, the networks usually do not compensate each other for the exchange of traffic. Peering agreements are based on the assumption that both networks will send and receive an equal amount of data. If this assumption does not hold any more, for example because new applications lead to an imbalance between the amount of traffic sent and received, network providers may be required to pay under a peering agreement as well. See Weiss and Shin 2004, pp. 43–46; van der Berg 2008; MIT Communications Futures Program and Cambridge University Communications Research Network Broadband Working Group 2005, pp. 5, 7.

199. The Broadband Working Group of the MIT Communications Futures Program has characterized this as the "Broadband Incentive Problem." See MIT Communications Futures Program and Cambridge University Communications Research Network Broadband Working Group 2005.

200. Eckersley, von Lohmann, and Schoen 2007; Svensson 2007; Schoen 2007b. Comcast stopped using this method of interfering with peer-to-peer file-sharing applications in 2008.

201. See, e.g., Aggarwal et al. 2004, pp. 219–220; Karagiannis, Rodriguez, and Papagiannaki 2005, pp. 63–64, 67–68; Aggarwal, Feldmann, and Scheideler 2007, pp. 31–32.

202. Mennecke 2005; Hellweg 2003.

203. See notes 173–177 above and accompanying text.

204. In particular, the Internet Policy Statement states that "consumers are entitled to access the lawful Internet content of their choice" and "to run applications and use services of their choice, subject to the needs of law enforcement" (Federal Communications Commission 2005a, p. 14988).

205. See, e.g., van Schewick 2008; Frischmann and van Schewick 2007.

206. See, e.g., Yoo 2006. The social costs associated with such a rule are discussed in Frischmann and van Schewick 2007.

207. Free Press and Public Knowledge 2007; Free Press et al. 2007; Federal Communications Commission 2008a.

208. The account in the text is based on the following sources: CBC News 2005; Barrett 2005; OpenNet Initiative 2005; Geist 2005, pp. 5–6; Miller 2005.

209. Barrett 2005.

210. The account in the text is based on the following: Hansell 2006; Olsen 2006; Karr 2006.

211. The account in the text is based on the following: Liptak 2007a,b; Public Knowledge et al. 2007.

212. All quotations are from Jones 2007.

213. For a comprehensive overview of the state of Internet Filtering, a description of the tools and techniques used in Filtering, and detailed regional and country reports based on empirical tests, see Deibert et al. 2008.

214. Discrimination to manage bandwidth or exclude unwanted content does not depend on the network provider offering a competing application.

215. For a similar argument with respect to the profitability of monopoly preservation through exclusionary conduct in new economy markets, if the monopoly is of intellectual property, see Posner 2001, p. 254.

216. The exclusionary conduct hurts independent producers of excluded complementary products in several ways: first, they are excluded from the part of the complementary market that consists of the network provider's Internet-service customers. As a result, they are not able to make any sales in that market. In addition, owing to economies of scale and, potentially, network effects in the production of their products, the exclusion from a part of the market may put them at a competitive disadvantage in the rest of the market as well. In the worst case, they may be forced to exit the complementary market completely. If they had made at least some sales to the network provider's Internet-service customers in the absence of the exclusionary conduct, the exclusion will reduce their profits.

217. Similarly, the network provider may fail to assess the situation correctly and discriminate against or exclude an independent provider of complementary products, even if none of the conditions under which this conduct would be profitable apply. Farrell and Weiser (2003, pp. 114–117) call this problem “incompetent incumbents” and include it in their list of exceptions to their version of the “one monopoly rent” argument.

218. Miller 2007.

219. Ibid.

220. E.g., Christensen 2000. For a full discussion of these reasons, see the discussion in chapter 8 below.

221. On the ability of antitrust law to capture all instances of discrimination that proponents of network neutrality are concerned about, see also van Schewick 2009, pp. 36–37; Frischmann and van Schewick 2007, p. 414, note 119.

222. More precisely, the bandwidth-adjusted transport prices may differ between applications. The following analysis expands on an argument by MacKie-Mason, Shenker, and Varian (1995), who examine the effect of application-blind and application-aware architectures on content provision and content creation, focusing on the ability to select different transport prices for different applications. See also MacKie-Mason, Shenker, and Varian 1996.

223. More precisely, the bandwidth-adjusted transport price is the same for all applications.

224. MacKie-Mason, Shenker, and Varian 1995, p. 7.

225. The difference is similar to the difference between a uniform price and a non-uniform pricing structure. A seller with some degree of market power who can charge different prices to different consumers can capture more consumer surplus than if it had to charge the same price to all customers. See the subsection of this chapter titled “Price Discrimination.”

226. The analysis also assumes that all consumers assign the same value to a particular application.

227. See also MacKie-Mason, Shenker, and Varian 1995, pp. 19, 22.

228. Applications for which users’ willingness to pay is lower than the general transport price will not be produced in an end-to-end network. See also MacKie-Mason, Shenker, and Varian 1995, p. 7.

229. If the profit-maximizing transport price for an individual application in the discriminatory network is equal to the uniform transport price in the end-to-end network, the remaining surplus is the same in both architectures.

230. Applications for which users’ willingness to pay is lower than the general transport price, however, will not be produced in such a network. See MacKie-Mason, Shenker, and Varian 1995, p. 7.

231. It is difficult to find a precise definition that encompasses all types of price discrimination; see, e.g., Tirole 1988, pp. 133–134; Varian 1989, pp. 598–599. For general treatments of price discrimination, see Varian 1989; Tirole 1988, chapter 3; Carlton and Perloff 2005, chapters 9–10. For an introduction to price discrimination in the context of information goods, see Shapiro and Varian 1999, chapters 2 and 3.

232. See, e.g., Carlton and Perloff 2005, p. 294.

233. Price discrimination is also feasible in monopolistically competitive industries, i.e., in industries characterized by the presence of significant product differentiation, relatively free entry and zero long-run profits. See Varian 1989, pp. 641–643.

234. For treatments of second-degree price discrimination, see, e.g., Varian 1989, pp. 611–617; Tirole 1988, pp. 135, 142–152; Carlton and Perloff 2005, chapter 10. For a treatment of second-degree price discrimination in the context of information goods, see Shapiro and Varian 1999, chapter 3.

235. In choosing products and prices, the firm is subject to a self-selection constraint: the firm must design products and prices such that consumers in any group do not prefer a product intended for another group. See, e.g., Tirole 1988, pp. 134–135, 142–162.

236. This strategy is also described by Wu and Lessig (2003, pp. 4, 15) and by Wu (2003a, pp. 151–154). Those authors present evidence for the use of this strategy by broadband operators. (See next note.)

237. Wu (2003a) presents evidence from the year 2001 showing that Comcast, an operator of broadband cable networks, had implemented the discriminatory pricing policy described in the text: customers of Comcast's "basic high-speed Internet service" were not allowed to use virtual private networks. Customers who wanted to use virtual private networks were offered the higher-priced "Comcast @Home Professional product" to which this contractual restriction did not apply. See Wu 2003a, pp. 151–152, 165. After this policy was made public and used as an example of "discriminatory behavior" by broadband cable operators in the debate over network-neutrality regulation, Comcast eliminated the ban on virtual private networks from its subscriber agreements and terms of service in 2003. See Wu 2003a, p. 156; Wu and Lessig 2003, p. 4. For a survey of contractual and technical restrictions on certain classes of applications employed by the ten largest cable operators and the six largest DSL providers in the United States, see Wu 2003a.

238. In general, the welfare effects of second price discrimination are theoretically ambiguous and can only be determined empirically on the basis of detailed knowledge about consumers' preferences and market and cost structures. See, e.g., Tirole 1988, pp. 149, 158; Varian 1989, p. 617. The general analysis of the welfare effects, however, focuses on stand-alone products and considers only allocative or static efficiency. As the text shows, in the case of platform products consisting of a primary product that can be used with various complementary products, price discrimination in the primary product based on the use of specific complementary products may have negative consequences for dynamic efficiency that are not captured by the conventional analysis. For a similar observation, see Wu 2003a, pp. 152–153.

239. Increasing profits is the objective of price discrimination. See, e.g., Carlton and Perloff 2005, pp. 293–94, 338. For a similar argument in the context of restrictions

on home networking that require that home networking equipment be purchased and installed by the cable operator, see Wu and Lessig 2003, p. 8.

240. For a similar argument, see Wu and Lessig 2003, p. 15; Wu 2003a, pp. 168–169.

241. For a similar observation in the context of restrictions on home networking that require that home networking equipment be purchased and installed by the cable operator, see Wu and Lessig 2003, p. 8.

242. Stated differently, at least some of the consumers who are excluded from the market for the blocked application under the discriminatory pricing policy would buy the blocked application under the uniform price for Internet services.

243. MacKie-Mason, Shenker, and Varian (1995, pp. 19, 22) make a similar argument when analyzing the effect of the ability to set different Internet transport prices for different applications on content creators' incentives to innovate.

244. See the references cited in notes c–f to box 6.1. The debate mostly focuses on whether network providers should be allowed to charge application or content providers for better service (e.g., for Quality of Service or higher bandwidth), but some proposals would ban any type of access charges.

245. Economides and Tåg 2007; Ralph 2006; Lee and Wu 2009. Note that applications and content providers pay user fees to their ISP who connects them to the Internet. Similarly, end users pay their ISP for access to the Internet. The providers of the access networks, in turn, compensate the backbone providers connecting the two access networks according to the interconnection agreements between them. In the scenario discussed in the text, the network providers of the end users who use the application or content would charge the providers of application or content for transport over the end users' access networks. For an introduction to the inter-provider compensation mechanisms used in the Internet, see van der Berg 2008.

246. With respect to the following, compare Rochet and Tirole 2003, pp. 1017–1020; Rochet and Tirole 2006, pp. 648–650.

247. Rochet and Tirole 2006, p. 650.

248. See generally Armstrong 2006, pp. 669–670, 677–689. Wright (2002) develops the same result in the context of cellular networks.

249. Ralph 2006. For a similar observation, see Weiser 2008.

250. See generally Wright 2002. For an application of this line of reasoning to the network-neutrality context, see pp. 2–3 of Ralph 2006.

251. See generally Armstrong 2006, pp. 669–670, 677–689; Wright 2002; Wright 2004, pp. 48–49.

Chapter 7

1. On asymmetric bandwidth, see my discussion of the broad version in chapter 2. On network-address translators and firewalls, see my concluding chapter, section “End-to-End Arguments in the Current Internet.”
2. See chapter 2, subsection “Costs of the Broad Version of the End-to-End Arguments.”
3. In theory, the functionality in the second network would enable the network provider to extract fees from providers of applications or of content by threatening to exclude them unless they pay an access charge. The architecture does not, however, provide the fine-grained accounting and billing infrastructure necessary to implement sophisticated pricing schemes.
4. Moving from the fully controllable to the core-centered architecture may increase the network providers’ benefits from application-level innovation: as fewer innovators are able to participate in application-level innovation (in the extreme case, network providers are the only remaining innovators), it is more likely that network providers will have market power over a specific application, which would enable it to increase prices. As a theoretical matter, the effect of market structure on incentives to innovate is ambiguous. Even if the increase in profits from market power increases network providers’ incentives to innovate, this effect probably will not offset the significant increase in the cost of developing new applications.

Chapter 8

1. A similar problem is at the heart of the debate about the scope of patents in intellectual-property law. The debate focuses on whether an innovator whose invention has a potential for significant subsequent improvement and variegation should be granted a patent of broad scope that would enable her to control subsequent innovations in the field opened by her invention. The article that started the debate was Kitch 1977. Subsequent contributions include Klemperer 1990; Merges and Nelson 1990; Scotchmer 1991; Merges 1994; Merges and Nelson 1994; Green and Scotchmer 1995; Matutes, Regibeau, and Rockett 1996; Scotchmer 1996; Lemley 1997; and Cohen and Lemley 2001. For an overview of the debate, see Jaffe 2000, pp. 549–552.
2. For an overview of the behavioral and cognitive assumptions underlying neoclassical approaches to innovation and an overview of different models, see Merges and Nelson 1994, pp. 3–5. See also Nelson 2005, pp. 9–38 (discussing neoclassical growth models). For a detailed comparison of neoclassical and evolutionary accounts of innovation, see Lipsey, Carlaw, and Bekar 2005, chapter 2. For some newer neoclassical growth models, see Grossman and Helpman 1990; Romer 1990; Aghion and Howitt 1992. For overviews of neoclassical models of innovation, see Reinganum 1989 and Scotchmer 2004.

3. If actors in these models disagree about the probability of success, this is attributed to asymmetric information. In other words, these models assume that once the different actors share their information, their views regarding the probability of success will be the same. This assumption, called the *common prior assumption* or the *Harsanyi doctrine*, is a fundamental assumption in many neoclassical models dealing with subjective probabilities (Kreps 1990, pp. 110–111).
4. For a prototypical example of the neoclassical approach to innovation described in the text, see the economic models in Dasgupta and Maskin 1987 or those in Farrell, Gilbert, and Katz 2003.
5. Lipsey, Carlaw, and Bekar 2005, p. 30. Newer neoclassical models of patent races acknowledge that different market positions (e.g., incumbent monopolist vs. new entrant) or different cost structures may lead to different incentives to innovate. For an overview of these models, see Gilbert 1989 or Reinganum 1989.
6. See Dasgupta and Maskin 1987.
7. See Kitch 1977, pp. 276, 278–279.
8. See Nelson and Winter 1977, p. 47; Nelson and Winter 1982, pp. 389–390; North 1990, pp. 80–82; Merges and Nelson 1994, p. 6; Rosenberg 1994, pp. 87–108; Rosenberg 1996; Cohen and Malerba 2001. On the link between the end-to-end architecture of the Internet and evolutionary theories of innovation, see Wu and Lessig 2003, pp. 5–7; Wu 2003a, pp. 145–146; Wu 2004, section II.A. Researchers working on the “open innovation” paradigm have used arguments similar to the ones described in the text to highlight the value to individual firms of using not only internal, but also external sources of innovation. See, e.g., Chesbrough 2003, 2006 and Chesbrough, Vanhaverbeke, and West 2006.
9. See Rosenberg 1996, pp. 334–336; Nelson 1991, pp. 64–70; Nelson 2005, pp. 11–12. (These sources all criticize new neoclassical growth models.)
10. See Rosenberg 1996; Holbrook et al. 2000; Shane 2000; Murmann 2003; Nelson 2005; Greenstein 2007b. The type of uncertainty described in the text is also called *Knightian uncertainty*. See Knight 1971; Rosenberg 1996, p. 340; Lipsey, Carlaw, and Bekar 2005, pp. 29–32; Nelson 2005, pp. 11, 21.
11. See Lipsey, Carlaw, and Bekar 2005, pp. 29–32, 44, 72–73. Scholars of innovation often model decision making under the type of uncertainty described in the text as a heuristic search (e.g., Nelson and Winter 1982) or with models of bounded rationality (e.g., North 1990, pp. 80–82; Nelson 2005, p. 37). However, models based on rational-choice theory can reach the same result by allowing the actors that have access to the same information to have different prior probability distributions over the success of different innovations, i.e., different subjective beliefs about the probability of success (e.g., Scotchmer 2004, pp. 56, 59–61).
12. See Nelson and Winter 1982, pp. 28, 61, 186–187; Merges and Nelson 1994, pp. 5–6.

13. See Nelson and Winter 1982, pp. 186–187, 389; Merges and Nelson 1994, p. 6; MacKie-Mason, Shenker, and Varian 1996, p. 215; Cohen and Klepper 1992; Cohen and Malerba 2001.

14. See Nelson and Winter 1977, p. 47; Nelson and Winter 1982, pp. 389–390; North 1990, pp. 80–92; Merges and Nelson 1994, p. 6; Rosenberg 1996, pp. 352–353.

15. See Dosi and Nelson 1994, p. 158: “But when put this way, rational choice theory would seem applicable to contexts to which the actors can be presumed familiar, and evolutionary theoretic arguments can be understood as an attempt to deal with situations where this presumption does not seem applicable. In particular, evolutionary theory can be argued to be needed for analyses of behavior in contexts that involve significant elements of novelty, so that it cannot be presumed that good responses already have been learned, but rather that they are still to be learned. . . . This line of argument would appear to preserve for neoclassical theory the analysis of decisionmaking in situations that are relatively stable and actions repetitive.” See also Rosenberg 1996, pp. 352–353; Lipsey, Carlaw, and Bekar 2005, pp. 72–73.

16. See Boudreau, Lacetera, and Lakhani 2008.

17. See the discussion later in this subsection.

18. E.g., Kihlstrom and Laffont 1979.

19. E.g., Kirzner 1997; Shane 2000; Shane and Venkataraman 2000; Sarasvathy et al. 2003.

20. E.g., Rosenberg 1996.

21. The following account is based on pp. 83–111 of Abbate 1999.

22. Cohen 2002, pp. 3–4, 26, 31, 64, 74.

23. Livingston 2007, p. 68.

24. PayPal’s origins can be traced to two different companies: Confinity (founded in December 1998 by Max Levchin, Peter Thiel, and Luke Nosek) and X.com (founded in March 1999 by Elon Musk). The two companies merged in March 2000, focusing on e-mail payments under the name paypal. The company later changed its name to PayPal. See Livingston 2007 and Denton 2007.

25. For qualitative studies of specific innovations or industries, see Holbrook et al. 2000; Shane 2000; Murmann 2003. For theoretical analyses, see Merges and Nelson 1994; Ardichvili, Cardozo, and Ray 2003; Lipsey, Carlaw, and Bekar 2005.

26. Afuah and Tucci 2001, p. 259.

27. Bronson 1999, pp. 78–79; Livingston 2007, pp. 18–19.

28. King 2002; Honan 2008.

29. Roth 2004; Maney 2006.
30. Charny 2003.
31. Livingston (2007) provides ample evidence of such behavior.
32. Spector 2000, pp. 37, 47–48, 64.
33. Cohen 2002, p. 49.
34. See Christensen 2000, pp. 128–130.
35. Yahoo Finance 2008b. (Again, in the United States a billion is a thousand million, i.e., 10^9 .)
36. Yahoo Finance 2008a.
37. Helft 2006; Spolsky 2003; Stephenson 2005.
38. More formally, if success is not guaranteed, increasing the number of attempts may increase the likelihood of success. This fact is well known from probability theory. For example, assume that the probability that an approach will be successful is δ for all approaches. Let n be the number of independent approaches to the same problem. Then the probability that at least one approach will be successful is $1 - (1 - \delta)^n$, which increases with n , the number of approaches (MacKie-Mason, Shenker, and Varian 1995, p. 20). Similarly, the more independent approaches, the higher the expected quality of the winner. This expectation too is familiar from probability theory. (See chapter 4 above.) This effect has been confirmed by empirical studies (Cohen and Malerba 2001; Leiponen and Helfat 2005). Although earlier research in economics predicts that an increase in the number of agents may reduce the level of effort incurred by the individual agent, the increase in the quality of the best result resulting from the parallel search effect seems to outweigh the effect of individual reductions in effort at least for complex problems. For a theoretical model, see Terwiesch and Xu 2008. For empirical confirmation of this result, see Boudreau, Lacetera, and Lakhani 2008.
39. Spillovers are uncompensated benefits that one person's activity provides to another (Frischmann and Lemley 2006, p. 102). That innovators need to engage in economic experiments to reduce the business uncertainty regarding what consumers want and what business models may be appropriate has been emphasized by Rosenberg (1994), by Stern (2005), and by Greenstein (2007a,b). Greenstein, in particular, highlights how actors learn not only from their own economic experiments, but also from competitors' successes and failures.
40. The ability to monitor experiments by others may differ depending on the type of experiment. For example, innovators may be able to keep technical experiments performed "in house" secret, but consumers' reaction to a product's feature or price or the success of a firm's business model usually is visible to other market participants, at least to some degree.

41. See the qualitative studies by Holbrook et al. (2000) and Shane (2000).
42. Spolsky 2000; Stephenson 2005; Kraus 2005b.
43. Cohen 2002, pp. 44, 59, 72.
44. Spector 2000, pp. 1–70. Amazon.com was not the first online bookstore. Computer Literacy Bookstores, Inc., a small independent bookstore in Silicon Valley that specialized in books for computer engineers and scientists, had set up a website and started selling books via e-mail in 1991 (Spector 2000, pp. 21–22). Book Stacks Unlimited, an online bookstore that had been founded by Charles Stack, a lawyer who had become a software engineer, had started selling books online through a bulletin board service in 1992 and launched an online bookstore in October 1994 (Spector 2000, pp. 29; Klein 1999; Rayport and Louie 2001, p. 5).
45. Barnes & Noble became the exclusive bookseller on AOL's marketplace in March 1997 and opened an online store open for the general public in May 1997 (Quistgaard 1997a,b; Knecht 1996). In an interview, Stephen Riggio, CEO of Barnes & Noble, claimed that his company had followed the evolution of Internet bookstores from the beginning (Schrage 1997).
46. Knecht 1996; Rayport and Louie 1998.
47. The account of Microsoft's reaction to the Internet is based on Rebello, Cortese, and Hof 1996a,b.
48. Berners-Lee and Fischetti 2000, pp. 28–30, 49.
49. Battelle 2005, pp. 82–84; 102; Vise and Malseed 2005, pp. 40–43, 85–86; Livingston 2007, pp. 132–133.
50. In addition, a rich literature in economics analyzes how incentives to innovate may differ between an incumbent monopolist and a new entrant. For an overview, see Reinganum 1989.
51. These assumptions are at the core of the resource-based theory of the firm and the dynamic capabilities framework. On the resource-based theory of the firm, see Wernerfelt 1984; Barney 1991; Peteraf 1993. On the dynamic capabilities framework, see Teece, Pisano, and Shuen 1997.
52. See Christensen and Rosenbloom 1995.
53. This argument has been advanced by K. B. Clark (1985) and is supported by empirical evidence (Tushman and Anderson 1986, 1990; K. B. Clark 1988).
54. For a detailed exposition of this argument and an application to the semiconductor photolithographic alignment equipment industry, see Henderson and Clark 1990. For other industry studies supporting this view, see the pointers to the literature on pp. 236–237 of Christensen and Rosenbloom 1995.

55. This theory has been developed by Christensen. For a detailed exposition of the theory, see Christensen 2000. For a short overview, see Christensen and Rosenbloom 1995. The theory is based on substantial evidence. In particular, evidence from a wide variety of industries supports the theory that new entrants have an advantage over established firms in perceiving and exploiting the value of disruptive technologies. This theory has been successfully applied to explain why new entrants overtook market leadership from established firms in the hard disk drive industry, the mechanical excavator industry, the steel industry, the retail industry, the printer industry, the personal digital assistant industry, the electric motor control industry, the motorcycle industry, the logic circuitry industry, the computer industry, the accounting software industry, and the insulin industry. See Christensen 2000. For alternative theories explaining when and why established firms may find it difficult to react to new developments, see Kaplan and Henderson 2005; Barnett and Pontikes 2008.

56. Hedlund 2005.

57. See Christensen and Rosenbloom 1995, pp. 238–242; Christensen 2000, p. 98.

58. See Christensen 2000, pp. xxii–xxiii.

59. See *ibid.*, pp. 44–45.

60. See *ibid.*, pp. 101–104.

61. See *ibid.*, pp. 79–84.

62. See *ibid.*, pp. 128–130.

63. *Ibid.*, pp. 82, 104, 155–156.

64. See *ibid.*, pp. xxi–xxii, 143–159.

65. See *ibid.*, pp. 44–45. Christensen (*ibid.*, p. xv) writes: “Most new technologies foster improve product performance. I call these sustaining technologies. Some sustaining technologies can be discontinuous or radical in character, while others are of an incremental nature. What all sustaining technologies have in common is that they improve the performance of established products, along the dimensions of performance that mainstream customers in major markets have historically valued.”

66. On disruptive technologies, see *ibid.*, p. xv.

67. See *ibid.*, pp. 121, 128–130.

68. See *ibid.*, pp. 143–159.

69. In addition, incumbent providers may have been more interested in protecting their existing dominant positions and highly profitable margins against competition by lower margin competitors. See the discussion in chapter 6.

70. Christensen 2000, p. 55.
71. *Ibid.*, pp. 70–71, 101–117.
72. *Ibid.*, p. 132.
73. *Ibid.*, pp. 128–130.
74. This definition is taken from p. 676 of Kortum and Lerner 2000.
75. For an in-depth treatment of theoretical and empirical research on venture capital, see Gompers and Lerner 2004. For a more practical treatment, see Gompers and Lerner 2001a. For a short overview with pointers to the literature, see Gompers and Lerner 2001b.
76. See Christensen 2000, pp. 82–84, 104; Scherer 2001, p. 20.
77. See Gompers and Lerner 2004, p. 161; Gompers and Lerner 2001a, pp. 53–58.
78. In recent years, successful funds have demanded higher proportions. For an overview of the compensation structure used in limited partnership venture funds, the most prominent organizational structure of venture-capital funds, see Gompers and Lerner 2004, chapter 5; Gompers and Lerner 2001a, pp. 100–102.
79. See Aoki 2001, pp. 368–369.
80. For venture capitalists, this incentive structure may even lead to excessive risk-taking. For some of the mechanisms used to mitigate this threat, see Gompers and Lerner 2001a, p. 106.
81. Scherer 2001, pp. 15–19, 20. Aoki (2001, chapter 14) arrives at a similar conclusion on the basis of a game-theoretic model.
82. For a theoretical deduction of this insight from agency theory and an empirical study supporting it, see Gompers and Lerner 2004, chapter 8.
83. *Ibid.*, pp. 164, 177–183.
84. For a historical breakdown of venture capital of investment by industry and stage of development, see Gompers and Lerner 2001a, pp. 72–73, table 4-5. See also Gompers and Lerner's analysis of a random sample of 794 firms that received venture-capital financing between January 1961 and July 1992 (2004, pp. 177–183).
85. See Gompers and Lerner 2002, p. 26.
86. National Venture Capital Association 2008, p. 27, figure 3.16.
87. *Ibid.*, p. 30, figure 3.22.
88. For an overview of other sources of financing, see Gompers and Lerner 2001a, pp. 9–11.

89. See *ibid.*, p. 62.
90. See *ibid.*, pp. 21–23.
91. Moukheiber 1996; Netscape 1995a–d.
92. In an empirical study of 170 young high-technology firms in Silicon Valley, Hellmann and Puri (2002) find that companies that obtain venture capital are more likely or faster to professionalize along these dimensions.
93. Battelle 2005, pp. 134–137; Vise and Malseed 2005, pp. 103–109.
94. Heilemann 2005; Vise and Malseed 2005; Reingold 2008.
95. See Gompers and Lerner 2004, p. 355.
96. On Netscape, see Gompers and Lerner 2001a, p. 22. On eBay, see Afuah and Tucci 2001, pp. 286–287; Cohen 2002, pp. 73–77.
97. Hellmann and Puri (2000) make this finding using a hand-selected data set of 173 start-up companies in Silicon Valley.
98. Gompers and Lerner 2001a, pp. 62–64.
99. Gompers and Lerner 2004, chapter 20; Gompers and Lerner 2001a, pp. 64–66.
100. On Adobe, see Scherer 2001, p. 11.
101. On eBay, see Afuah and Tucci 2001, pp. 286–287.
102. According to Gompers and Lerner (2001a, pp. 5, 70), these companies were funded by venture capitalists. Gompers and Lerner do not discuss Adobe and eBay.
103. Gompers and Lerner 2001a, p. 70.
104. Lerner 2000, p. 204.
105. See Kortum and Lerner 2000. The article focuses on manufacturing industries, since survey evidence suggests that the reliance on patenting as a means of appropriating new technological discoveries as opposed to, e.g., trade secrecy or first-mover advantages is much higher in these industries. Patenting is thus likely to be a better indicator of the rate of technological innovation in the manufacturing sector (Kortum and Lerner 2000, p. 678). For an accessible discussion of the findings of this study, see Gompers and Lerner 2001a, pp. 74–77.
106. Gompers and Lerner (2001a, p. 77) summarize the results of Kortum and Lerner (2000) this way: “Given the continued rise in venture funding since 1992, and assuming that the potency of venture funding remained consistent with the measured impact of venture capital and patenting rates, the calibrations imply that, by 1999, venture capital investment accounted for about 18% of US innovative activity.”

107. The following account is based on Gompers and Lerner 2001a and Gompers and Lerner 2000.

108. On eBay, see Cohen 2002, p. 74. On Skype, see Rangar 2005. On Google, see the subsection of the present chapter titled “The Value of Innovation by New Entrants.”

109. The seminal work in this area is von Hippel 1988.

110. See von Hippel 1988, pp. 3–4; von Hippel 2005, p. 3. Depending on its functional relationship to the innovation, an innovator can be a manufacturer with respect to some innovations and a user with respect to others. For example, if Google improves its search-engine algorithm, we would categorize it as a manufacturer, as it profits from the search engine by making it available to users. In contrast, if Google develops software to better manage its server farms, we would consider it a user.

111. Livingston 2007, pp. 233–234.

112. Grimmer 2006; Livingston 2007; Park 2008.

113. Spicer 1998.

114. See Luethje, Herstatt, and von Hippel 2005, pp. 962–964.

115. See von Hippel and von Krogh 2003, p. 214; von Hippel 1998, p. 630; Luethje, Herstatt, and von Hippel 2005, p. 953. In a Web-based survey that was administered to 684 software developers in 287 free or open-source software projects, 58.7% reported use of the output they produce for work or non work-related needs as one of their three most important motivations to innovate. See Lakhani and Wolf 2005. For an empirical study showing higher satisfaction of users of the open-source Apache Web server software who customized their software compared to users who did not customize, see Franke and von Hippel 2003, pp. 1209–1210.

116. For a short overview of the theoretical literature on intrinsic and extrinsic motivation, see Lakhani and Wolf 2005, pp. 4–5. In a Web-based survey that was administered to 684 software developers in 287 free or open-source software projects, 44.9% ranked “project code is intellectually stimulating to write” as one of the top three motivations. 41.8% were motivated by the improvement in programming skills resulting from their involvement in the project. See Lakhani and Wolf 2005.

117. See chapter 4 above.

118. For a more detailed analysis of the opportunity costs incurred by programmers working on an open-source software development project, see Lerner and Tirole 2002, p. 213.

119. Being able to use the resulting software often seems to have been the initial motivation. In addition, improving the application often involved research, which

was fun. During the time of the ARPANET, people who developed successful applications were highly regarded, so the desire to improve their reputation may have motivated these researchers as well. See Partridge 2008a.

120. Berners-Lee and Fischetti 2000, pp. 16–19. His idea for the World Wide Web emerged from Enquire, a simpler program that Berners-Lee had written during an earlier employment as a contractor at CERN to help him keep track of relationships among people, equipment, and programs. Foreshadowing the websites and links of the World Wide Web, Enquire made it possible to store pieces of information and to establish links between them. See Berners-Lee and Fischetti 2000, pp. 10–12.

121. In July 2008, Apache's market share was 49%, Microsoft's was 36%, and Google's was 10% (Netcraft 2008).

122. Moody 2001, pp. 130–139; chromatic and Sheppard 2008. According to the PHP Group (2008), "PHP/FI was created by Rasmus Lerdorf in 1995, initially as a simple set of Perl scripts for tracking accesses to his online resume," and Lerdorf "named this set of scripts 'Personal Home Page Tools.'" Yukihiro Matsumoto, developer of Ruby, recalled: "I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language." (Stewart 2001)

123. For a short introduction to the history and characteristics of open-source software, see von Hippel and von Krogh 2003, pp. 209–212; Lerner and Tirole 2002, pp. 200–207. For views on the open-source movement from some important participants, see DiBona, Ockman, and Stone 1999; Raymond 1999.

124. For biographies of the members of the core developer group of Ruby on Rails, see Ruby on Rails 2008.

125. The average level of respondents' agreement with that statement was 4.7 on a scale of 5. See von Hippel 2007, p. 297.

126. Lakhani and Wolf 2005.

127. Owing to these characteristics of open-source software, a particular open-source software program and open-source tools for software design and testing constitute a complete tool kit for user innovation for users of this program in the sense described by von Hippel (2001). This observation is made by Franke and von Hippel (2003, p. 1204) with respect to the Apache Web server software.

128. For unfamiliar users, the costs of understanding and modifying an existing code base may be significant. For such users, it may be more cost-effective to use sub-optimal software or to hire someone else to learn and modify the code for them.

129. In making this make-or-buy decision, the user faces a trade-off that is well known from the literature on make-or-buy decisions with respect to software (e.g.,

Messerschmitt 2000, pp. 286–278). Buying the product from the manufacturer is usually less costly, since the development costs are spread over a number of sales; however, the manufacturer's product is standardized to meet the needs of the average user and may leave some of the user's needs unserved. For example, SAP R/3 (a standardized software for business applications with extensive options for customization) is said to meet about 80% of most users' needs (Bancroft, Seip, and Sprengel 1998, p. 141). In contrast, when making the software itself, the user bears the complete development costs and the development risk, but gets software that exactly meets its needs. In this situation, the user will often prefer to buy from the manufacturer. This is not an option, however, if there is no standardized product available in the market or the existing products are not satisfactory. As will be set out below, such a situation may well occur for applications supporting specific needs of an individual user or a small group of users. In this situation, the ability of the user to develop the innovation herself becomes highly relevant. For an analysis of a user's decision whether to innovate herself or to buy the product from a manufacturer of custom products, see von Hippel 2005, chapter 4.

130. See Luethje, Herstatt, and von Hippel 2005, pp. 963–964.

131. For a short overview of common approaches to this problem and pointers to the literature, see e.g., Jeppesen 2005, pp. 349–351.

132. The term “sticky information” was coined by von Hippel (1994). For an overview of the reasons for information stickiness, see von Hippel 1994, pp. 430–432; von Hippel 1998, pp. 630–631.

133. See von Hippel 1998, pp. 630–631; von Hippel 2001, pp. 248–249.

134. See von Hippel 1998, p. 630; Franke and von Hippel 2003, pp. 1199–1202.

135. Of course, there may be small markets whose consumers have a very high willingness to pay.

136. See Franke and von Hippel 2003, pp. 1199–1202. Benkler (2000, p. 576) describes the same phenomenon in the context of professional content provision.

137. Franke and von Hippel 2003, pp. 1199, 1201.

138. Based on surveys of literature on market segmentation studies, Franke and von Hippel (2003, pp. 1199, 1201–1202) find that current practice in market segmentation studies may leave about half of customers' needs within a segment unaddressed. Intuitively, one might expect the problem to be less profound in software, where the costs of coding additional functionality may not be very high, which may make it easier to respond to individual customers' needs. Adding additional functionality increases the complexity of the product, which considerably increases the amount of testing and support. Because most of the cost of software comes from testing, debugging, and customer maintenance (according to a study by Cusumano (1991, p. 65), testing, debugging, and maintenance account for 82% of the cost of

software), software vendors have the same incentive to limit the number of additional features and focus on the need of the average consumer as the manufacturer innovators of physical products.

139. According to von Hippel (1988, p. 107), lead users are defined by two characteristics: “1. Lead users face needs that will be general in a marketplace, but they face them months or years before the bulk of that marketplace encounters them, and 2. Lead users are positioned to benefit significantly by obtaining a solution to those needs.” On lead users in general, see *ibid.*, chapter 8.

140. See the subsection of this chapter titled “The Value of Innovation by New Entrants.” See also von Hippel 2007, p. 302.

141. See Franke and von Hippel 2003, pp. 1200–1201. In two surveys of market segmentation studies described in the literature, the authors found that the current market segmentation techniques leave unaddressed, on average, 54% (survey 1) and 46% (survey 2) of the total variation of customer needs within a segment.

142. For a detailed analysis of the welfare effects of user innovation in the context of the economic literature on product diversity, innovation, and welfare, see Henkel and von Hippel 2005, pp. 76–81. For a complementary view on the importance of user involvement in innovation, see Bar and Munk Riis 2000.

143. Grimmer 2006; Livingston 2007; Park 2008.

144. The literature on free revealing by users usually defines free revealing as follows: “When we say that an innovator ‘freely reveals’ proprietary information, we mean that all existing and potential intellectual property rights to that information are voluntarily given up by that innovator and all interested parties are given access to it—the information becomes a public good. Thus, we define the free revealing of information by a possessor as the granting of access to all interested agents without imposition of any direct payment.” (Harhoff, Henkel, and von Hippel 2003, pp. 1753–1754) In the case of the Internet, this definition is not fully accurate—software is usually released under a license that describes the restrictions associated with the software.

145. After Bloglines continued to grow, Fletcher decided to focus on it full-time. In February 2005, Bloglines was acquired by Ask Jeeves (Livingston 2007, pp. 233–236). That user innovators end up commercializing their innovation once it becomes successful is not uncommon on the Internet. Other examples of this phenomenon are the blogging software Movable Type (*ibid.*, pp. 405–417) or the online book-marking service *del.icio.us* (*ibid.*, pp. 223–232).

146. The source code is usually not made available without any restrictions, but released under a license which specifies what exactly programmers are allowed to do with the code. For an overview of different licenses and an empirical study regarding their use, see Lerner and Tirole 2005b.

147. Berners-Lee and Fischetti 2000, pp. 32, 50–53, 80.
148. Grimmer 2006.
149. SourceForge.net 2008.
150. See von Hippel and von Krogh 2003, pp. 212–213.
151. See von Hippel 2007, p. 306.
152. For a more detailed analysis of the positive welfare effects of free revealing by users, see Henkel and von Hippel 2005, pp. 81–82.
153. Park 2008; Slater 2008.
154. Netcraft 2008. The Netcraft survey includes parked domains (i.e., Internet domains that point to a server, but that are not actively used as websites) and may therefore overestimate the number of websites (Varghese 2003).
155. Net Applications 2008.
156. This includes commercial versions of Sendmail that are distributed by the company Sendmail, Inc. (Sendmail 2008).
157. Lerner and Tirole 2002, p. 210.
158. Lerner and Tirole 2005a, pp. 102–107.
159. In the context of the patent scope debate, this argument is made by Kitch (1977, pp. 277–278).
160. In the core-centered architecture, the number of concurrent application projects is limited for reasons independent of the usual reasons for the limits of internal organization. See box 8.8.
161. For an overview of the factors that may impede or prevent a contractual solution, see, for example, Lemley 1997, pp. 1048–1067, in the context of the patent scope debate. For an analysis of the conditions under which innovators may prefer selling their innovation to an incumbent in the “market for ideas” to competing with the incumbent in the product market, see Gans and Stern 2000; Gans, Hsu, and Stern 2002; Gans and Stern 2003.
162. To see this, assume that the probability that one decision maker will not approve the innovation is β , with $0 < \beta < 1$, and that the number of decision makers is n . In order for the project to be realized, all decision makers must approve the project. In this case, the probability that at least one decision maker will not approve the project, and, consequently, the project will not be realized, is $1 - (1 - \beta)^n$, which increases with n , the number of innovators.
163. Livingston 2007, pp. 111–125.
164. See Christensen and Raynor 2003, chapters 7 and 8; Amram and Kulatilaka 1999; Chesbrough 2003, pp. 12–13.

165. See Nelson and Winter 1982, pp. 389–390; Merges and Nelson 1994, p. 6; Rosenberg 1996, p. 353.

166. MacKie-Mason, Shenker, and Varian 1995, p. 21.

167. See, e.g., the empirical studies on the distribution of gains for new drug chemical entities and investments in high-technology start-ups reported by Scherer (2001) or the empirical study on the returns to investments in the software industry by Lerner (2002). The empirical study on the return to invention by independent inventors by Åstebro (2003) arrives at similar findings.

168. For a similar argument in the context of peer production, see Benkler 2002, p. 423.

169. Rosenberg 1994; Greenstein 2007a,b.

170. Network providers will exclude such an application, if the reduction in profit in the Internet-service market due to the reduction in application variety (which is also dependent on factors such as switching costs and the network providers' ability to use discrimination instead of exclusion) is offset by the increase in profit in the application market.

171. Thus, although Apple is not acting as a network provider here, it is acting as a central gatekeeper that controls which applications users can deploy.

172. Arthur 2008; Hadar 2008; Manjoo 2008; Wilson 2008.

173. On PayPal, see the discussion early in this chapter. On Flickr, see box 8.3.

174. See note 4 to chapter 7.

Chapter 9

1. The analysis in this paragraph and in the next 11 paragraphs draws on pp. 383–386 of van Schewick 2007.

2. See Reinganum 1989; Tirole 1988, pp. 399–400; Katz 2002, pp. 329–331.

3. For an overview of this literature, see Reinganum 1989. For a particular example of such a model, see Dasgupta and Maskin 1987.

4. See Tirole 1988, pp. 399–400.

5. See Mansfield et al. 1977; Jones and Williams 1998.

6. Lichtman 2000, p. 615.

7. This observation is made, in different contexts, by Bresnahan and Trajtenberg (1995) and Lichtman (2000).

8. See Farrell and Katz 2000, p. 414 and appendix; Bresnahan and Trajtenberg 1995, p. 94.

9. For some important refinements to this statement, see Farrell and Katz 2000. As Farrell and Katz demonstrate, integration between two firms that each are the sole supplier of a component that is complementary with the other does not necessarily increase the incentives to invest in socially valuable research and development. (See *ibid.*, appendix.) In addition, they show that integration between a monopoly supplier of one component with one of several suppliers of a complementary component may inefficiently lower independent suppliers' incentives to innovate.

10. On platform products, see Bresnahan and Greenstein 2001 and Lichtman 2000.

11. See the explanation in this paragraph and the next five. For an alternative explanation of how the Internet contributes to economic growth, see Crawford 2007, part II.

12. Jones (2002, p. 229) attributes this percentage of growth to "a rise in the stock of ideas produced by researchers throughout the G-5 countries." See also Helpman's (2004, pp. 47–48) description of Jones's results.

13. On general-purpose technologies, see Bresnahan and Trajtenberg 1995, Bresnahan and Greenstein 2001, and Helpman 1998.

14. See Bresnahan and Trajtenberg 1995 and Helpman and Trajtenberg 1998b.

15. See Bresnahan and Trajtenberg 1995, pp. 86–88; Bresnahan and Greenstein 2001, p. 96.

16. Bresnahan and Greenstein 2001, p. 97.

17. For empirical evidence suggesting that the US productivity acceleration observed in the mid to late 1990s is linked to information technology, see Brynjolfsson 1996, Oliner and Sichel 2000, Jorgenson 2001, and Jorgenson, Ho, and Stiroh 2008. For a more critical statement, see Gordon 2000. For an overview of the debate, see Litan and Rivlin 2001, pp. 12–17.

18. See David 1990. For similar explanations, see Brynjolfsson and Hitt 2000, Bresnahan and Greenstein 2001, Brynjolfsson and Hitt 2003, and Basu and Fernald 2007. Owing to the time and cost needed for co-invention to take place, delay between the time of investment in a general-purpose technology and the time when its effect on economic growth can be measured may be a general feature of general-purpose-technology-based growth. See Helpman and Trajtenberg 1998a,b.

19. See Harris 1998.

20. Current ways of measuring economic growth may underestimate the economic value created by the Internet. As Brynjolfsson and Saunders (2009) point out, traditional measures of output (which in turn are used for gross domestic product and productivity accounting) do not adequately capture the value created by information technology and the Internet for two reasons: First, the gross domestic product

generally does not include economic activity that takes place outside the market, or market transactions in used goods and services. As a result, applications or content that are made available online for free (e.g., Wikipedia or search engines) do not count as output for the gross domestic product. The same applies for applications or services that facilitate transactions in used goods (e.g., eBay or Amazon's market for used books). (See *ibid.*, pp. 21–25, 91–93, 120–121.) Second, traditional measures of input and output do not adequately capture the value created by “new products, quality improvements, increased product variety, [and] improved timeliness” (*ibid.*, p. 122). Research that attempts to quantify the value of consumer surplus from specific products or online markets indicates that this value is huge. However, it does not appear in the gross domestic product. For example, studies have shown that the value to consumers of greater variety and choice in the online book market is about a billion dollars; that the estimated total value of consumer surplus from transactions on eBay was about \$7 billion in 2003; or that the overall increase in consumer surplus from Amazon's used book markets was about \$67 million per year. (See *ibid.*, pp. 25–38, 109–115, 120–122.)

21. For a similar observation, see Litan and Rivlin 2001, pp. 104–107.

22. See also Frischmann 2005. Frischmann's work on infrastructure has fundamentally shaped my thinking on this issue. According to Frischmann, infrastructure resources are general-purpose resources that generate value primarily as inputs “into a wide range of goods and services, including private goods, public goods and non-market goods.” For the full definition, see *ibid.*, p. 956.

23. My analysis focuses on the possibilities afforded by these applications. It makes a claim about the Internet's potential which may or may not be realized. It is not meant as a description of current reality. In reality, differences in class, socioeconomic status, race, literacy or geographic location may result in a much more hierarchical and less open environment than the one described in the text (see Barney 2000; Hindman 2008).

24. In Internet policy debates, some scholars criticize other researchers' focus on application innovation as too narrowly focused on economic issues. The criticism seems to neglect that applications are necessary to enable users to create value from the Internet in all areas of society. Thus, a focus on application innovation does not automatically imply a focus on economic effects only.

25. Benkler 2006, chapter 10; Boase et al. 2006; Griffith and Fox 2007; Lenhart 2009.

26. Benkler 2006, pp. 129–355. The list in the text cites from a similar list by Benkler (*ibid.*, p. 2).

27. This story is more complicated than it appears in the text. As the “long tail” literature has shown, lower costs of producing and distributing information goods, the ability to offer a much larger number of products online than is possible in

conventional stores, and lower search costs have made it possible to create and offer consumers access to much greater product variety online than was possible before the Internet. Thus, applications such as search engines or product marketplaces may create value not just by enabling people to find what they want, but also by enabling more product variety (Anderson 2006; Brynjolfsson, Hu, and Smith 2006; Brynjolfsson, Hu, and Simester 2006). Empirical research suggests that the value of increased product variety online may be even larger than the value resulting from lower prices online, see Brynjolfsson, Hu, and Smith 2003.

28. For an attempt to describe the externalities associated with speech more generally, see Frischmann 2008, pp. 310–321.

29. Mercurio 2002.

30. For descriptions of these cases and others, see Drezner and Farrell 2008, pp. 2–6.

31. Farrell and Drezner 2008, pp. 23–27.

32. Mutz 2006.

33. According to Frischmann (2005), Internet users create these externalities when they engage in the creation of public and non-market goods.

34. For a similar description of the role of technology, see Benkler 2006, pp. 16–18; Lipsey, Carlaw, and Bekar 2005, pp. 16–19.

35. See Balkin 2004; Balkin 2008; Benkler 2006; Fisher 1988; Fisher 2001; Lessig 2001; Lessig 2004; Lessig 2008a.

36. On how to best get users to internalize at least some of the negative externalities created by their uses, and on the effect such measures may have on the positive externalities created by users, see Frischmann and van Schewick 2007, part II.

37. The cost of adopting new uses of the general-purpose technology contributes to the costs of co-invention.

38. Benkler 2006; Balkin 2004, 2008; Fisher 1988, 2001.

39. It may be possible to work around this limitation if the user can upload his information product to a server that makes the work available to others. However, even uploading long text, photos or video once is more difficult and time-consuming with small upstream bandwidth than with symmetric bandwidth.

40. For a detailed exposition of this argument, see Frischmann 2005. For a shorter summary, see Frischmann and van Schewick 2007.

41. Strictly speaking, applying the broad version only ensures user choice in a deployment context in which users control the end hosts. See note 118 to chapter 2 and accompanying text.

42. Others point out that if the denial of service attack is intended to overwhelm a specific end host by sending more traffic than this host can process, or by overloading the link to the end host, stopping the attack requires stopping the traffic before it reaches the end host, which may require stopping it in the network.

43. Zittrain 2008.

44. See the section titled “Effect of Differences in Control over Deployment” in chapter 8 above. See also Gillett et al. 2001 and Zittrain 2008, chapter 5 and pp. 162–168.

45. Zittrain 2008, pp. 163–167. See also Zittrain 2009.

46. Zittrain (2008, pp. 164–165) ascribes this requirement to “end-to-end theory,” criticizing those who “cling to a categorical end-to-end approach.” Although Zittrain uses a slightly different terminology (“end-to-end theory”), he seems to be referring to the (broad version of the) end-to-end arguments (*ibid.*, pp. 164–165 and note 24). In this case, his description of the broad version is overly broad. For an interpretation similar to mine, see Reed 2009b. Ultimately, the results of Zittrain’s approach (he suggests replacing “strict loyalty to end-to-end neutrality” with “a new generativity principle, a rule that asks that any modifications to the Internet’s design or to the behavior of ISPs be made where they will do the least harm to generative possibilities” —2008, p. 165) may be similar to the results of the trade-off I suggest below.

47. See also Reed 2009b.

48. See also Reed 2009b.

49. Whether distributed denial of service can be identified and stopped only in the network is subject to debate. For example, Stavrou et al. (2005) propose an approach that, according to them, would work completely on end hosts. Gligor (2005) describes denial of service attacks that can only be mitigated at the end hosts. For an overview of solutions that require support from the network, see Xiaowei, Whetherall, and Anderson 2005, p. 242.

50. Blumenthal and Clark 2001, p. 80 (citing personal communication with Jerome Saltzer).

51. Reed, Saltzer, and Clark 1998, p. 69. See also Saltzer, Reed, and Clark 1984, pp. 282–283; Bradner 2006, p. 77. The latter two sources discuss what I call the narrow version of the end-to-end arguments.

52. One may wonder whether implementing such a function in the network may be justified by the broad version itself. If implementing the function solely at the end points does not provide “effective” security in practice, one could argue that the function cannot be completely and correctly implemented at the end hosts alone. Under such an interpretation, “completely and correctly implemented” would refer not only to technical correctness (the traditional interpretation), but would also refer to the effectiveness of the solution in practice. Broadening the

interpretation of “completely and correctly implemented” in this way, however, replaces a relatively clear criterion (the technical correctness of the solution) with a criterion that is open to debate. Such an interpretation may have unwanted side effects by broadening the range of functions that could be implemented in the network even in cases that are unrelated to security, and may create unnecessary debates about “effectiveness in practice.” In contrast, the solution advanced in the text (allowing deviations from the broad version case by case on the basis of a trade-off) clearly shows that this decision requires a trade-off, and invites debate about the relevant question how this trade-off should be resolved.

53. Generally, and for one such attempt to reconcile the change from a network connecting a small number of users that trust each other to a global network connecting millions of users whose trustworthiness is often unclear with the end-to-end arguments, see Clark et al. 2003, pp. 6, 39–61, 71–74.

54. See my concluding chapter. See also Handley 2006 and Rosenberg 2008.

55. See Benkler 2006, pp. 129–355. The list in the text cites from a similar list by Benkler (*ibid.*, p. 2).

56. This argument assumes that the lack of optimization does not make it impossible for some applications to use the network. This is exactly what the broad version tries to prevent by keeping the network general. Some scholars criticize the Internet Protocol for offering only “best effort” service and not guaranteeing bandwidth, jitter, or delay. As was discussed in box 4.3., this may create problems for applications that strictly require guaranteed bandwidth, jitter, or delay. To the extent that there is a problem (in the current Internet, real-time applications have been able to function without these guarantees), it does not result from a lack of optimization, though. Instead, it results from the fact that the service provided by the Internet layer may not be general enough, because there is a class of applications it cannot support. As we saw in chapter 3, the broad version of the end-to-end arguments does not generally prevent the provision of Quality of Service. Thus, insofar as there is a problem, it results from the way the broad version was applied; it is not a problem of the broad version of the end-to-end arguments as such. Finally, technologies that would enable the Internet to offer Quality of Service have been standardized by the Internet Engineering Task Force. As we saw in chapter 4, these technologies have been implemented by the vendors of routers and of operating systems for end hosts, but they have not been widely deployed in the operational Internet. Thus, the fact that the current operational Internet mostly does not offer Quality of Service is attributable to a lack of deployment, not to a lack of available Internet technology.

57. For a detailed analysis of the positive externalities of broadband infrastructure, see Atkinson 2007.

58. Wu 2006b.

59. Anderson 2007.
60. O’Connell 2005; Maier 2008; Oates 2006.
61. On IMS, see Camarillo and Garcia-Martin 2008. On session control and charging in IMS, see Cuevas et al. 2006 and Camarillo and Garcia-Martin 2008. On IMS deployments, see Taylor and Hettick 2008 (on Verizon), Buckley 2007 (on AT&T), and Crane 2007 (on BT).
62. Cisco Systems 2006, p. 8.
63. Braden et al. 2000, pp. 6–9.
64. On asymmetric bandwidth, see the discussion of the broad version in chapter 2 above.
65. On network-address translators and firewalls, see my concluding chapter.
66. On discriminatory bandwidth management, see chapter 6 above.
67. The definition of intertemporal bias in the text mirrors Read’s (2004, p. 425) definition of “intertemporal choice.”
68. See Loewenstein and Prelec 1992; Frederick, Loewenstein, and O’Donoghue 2002; Read 2004.
69. E.g., Read 2004, p. 431.
70. For an overview of the potential explanations, see Lavery 1996 and Marginson and McAulay 2007.
71. See Cuthbertson, Hayes, and Nietzsche 1997 and Bushee 2001.

Conclusion

1. Saltzer, Reed, and Clark 1981. The 1981 paper was a conference paper. A revised version of the paper was published in the *ACM Transactions on Computer Systems* as Saltzer, Reed, and Clark 1984. The later version is commonly cited as the paper that first identified and described the end-to-end arguments.
2. See Reed, Saltzer, and Clark 1998.
3. For a similar summary of the narrow version, see p. 387 of Peterson and Davie 2007.
4. On this division (end hosts implement all layers, while IP routers typically implement only lower layers, up to and including the Internet layer), see Kurose and Ross 2008, pp. 51–52. In practice, routers may implement higher layers to terminate routing protocols such as BGP or management protocols.
5. The forces behind the increasing pressure on the end-to-end design of the architecture of the Internet have been described by Blumenthal and Clark (2001). Most

subsequent literature summarizes Blumenthal's and Clark's analysis (see, for example, Kempf and Austein 2004, section 3; Yoo 2004, pp. 34–37). The problems resulting from the increasing number of architectural solutions that deviate from the original design principles of the Internet have also been covered by the IETF in a series of Requests for Comments. See, e.g., Kaat 2000; Carpenter 2000; Carpenter and Brim 2002; Floyd 2002; Kempf and Austein 2004. See also Saltzer 1999; Reed 2000; Moors 2002; Clark et al. 2002.

6. See Clark et al. 2002, p. 354.

7. As has been indicated above, the fact that parts of the application run on end hosts that are topologically in the core of the network and under the administrative ownership of third parties is irrelevant: the broad version only focuses on the functional relationship between users and providers of communication services. See chapter 3 above.

8. See Sterbenz and Touch 2001, pp. 351–352.

9. The moves toward Web-based applications and cloud computing are examples of this trend. For an application of “end-to-end” reasoning to these issues, see Clark and Blumenthal 2007. See also Clark et al. 2003, pp. 47–61; Zittrain 2008, p. 167.

10. Zittrain 2008, pp. 167–168.

11. Clark and Blumenthal (2007) take a similar position.

12. Such devices are called *middleboxes*. See Carpenter and Brim 2002.

13. See Kruse 1999, pp. 1–2; Blumenthal and Clark 2001, pp. 83–84.

14. On NATs, see Srisuresh and Holdrege 1999 and Srisuresh and Egevang 2001.

15. See Kruse 1999, section 2; Tanenbaum 2003, p. 448.

16. See Blumenthal and Clark 2001, p. 83.

17. Apart from the negative effect on innovation, the proliferation of middleboxes creates a number of other problems: for example, such devices may affect end-to-end data integrity, may prevent the use of end-to-end IPSec, or may require the maintenance of state in the device, violating the principle of fate sharing and introducing single points of failure. See Kaat 2000; Carpenter 2000; Hain 2000; Carpenter and Brim 2002. IPSec (Internet Protocol Security) is a suite of protocols, standardized by the IETF, that provides various security services (such as encryption or authentication) at the Internet layer. In end-to-end IPSec, IPSec is used between the endpoints of an IP connection.

18. On the problems for applications caused by NATs, see Hain 2000, Senie 2002, Holdrege and Srisuresh 2001, and Dutcher 2001. On the problems for applications caused by firewalls, see Freed 2000 and Cheswick, Bellovin, and Rubin 2003.

19. In a way, NATs and firewalls that create the necessity to coordinate an innovation with a large number of economic actors that produce or operate devices in the core of the network convert the Internet to an anticommons. In an anticommons, “multiple owners are each endowed with the right to exclude others from a scarce resource, and no one has an effective privilege of use. When too many owners hold such rights of exclusion, the resource is prone to underuse—a tragedy of the anticommons.” (Heller 1998, abstract) The concept of an anticommons has been developed by Heller (*ibid.*) and mirrors the concept of common property. On the tragedy of the commons, see Hardin 1968. On commons and anticommons, see Buchanan and Yoon 2000.

20. See Handley 2006 and Rosenberg 2008.

21. To work around the problem, an application that wants to use a new transport-layer protocol can include an implementation of the transport-layer protocol as part of the application. As transport-layer protocols work best when they are implemented as part of the operating system, this strategy often comes with a performance penalty.

22. For more detailed discussions, see Gillett et al. 2001 and Zittrain 2008. See also note 118 to chapter 2 and accompanying text.

23. Benkler 2006, pp. 129–355. The list in my text cites from a similar list from p. 2 of the same work.

24. See Camarillo and Garcia-Martin 2008, pp. 6–8.

25. For an overview of the obstacles of developing applications in the traditional mobile Internet, see Wu 2007.

26. For a detailed analysis of Google’s motivations and for a history of Android, see Roth 2008.

27. Google 2007.

28. See Whitt and Faber 2008. Google had also asked for a fourth condition, which would have required a licensee of 700-MHz spectrum to allow third parties (such as Internet service providers) to interconnect at any technically feasible point in the wireless network of that licensee.

References

- Abbate, J. 1999. *Inventing the Internet*. MIT Press.
- Afuah, A., and Tucci, C. L. 2001. *Internet Business Models and Strategies: Text and Cases*. McGraw-Hill/Irwin.
- Aggarwal, V., Bender, S., Feldmann, A., and Wichmann, A. 2004. Methodology for Estimating Network Distances of Gnutella Neighbors. Paper presented at Workshop GI-16, INFORMATIK 2004.
- Aggarwal, V., Feldmann, A., and Scheideler, C. 2007. Can ISPs and P2P Users Cooperate for Improved Performance? *Computer Communication Review* 37(3): 31–40.
- Aghion, P., and Howitt, P. 1992. A Model of Growth through Creative Destruction. *Econometrica* 60(2): 323–351.
- Alchian, A. A., and Demsetz, H. 1972. Production, Information Costs, and Economic Organization. *American Economic Review* 62(5): 777–795.
- Alexander, C. 1978. *The Timeless Way of Building*. Oxford University Press.
- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press.
- Allen, F., and Gale, D. 2001. *Comparing Financial Systems*. MIT Press.
- Almeroth, K. C. 2000. The Evolution of Multicast: From the MBone to Interdomain Multicast to Internet2 Deployment. *IEEE Network* 14(1): 10–20.
- American Registry for Internet Numbers. 2007. ARIN Board Advises Internet Community on Migration to IPv6. <https://www.arin.net/announcements/2007/20070521.html>.
- Amram, M., and Kulatilaka, N. 1999. *Real Options: Managing Strategic Investment in an Uncertain World*. Harvard Business School Press.
- Anderson, C. 2006. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion.

- Anderson, N. 2007. Deep Packet Inspection Meets 'Net Neutrality, CALEA. *Ars Technica*, July 25. <http://arstechnica.com/articles/culture/Deep-packet-inspection-meets-net-neutrality.ars>.
- Anderson, N. 2008a. Canadian ISPs Furious about Bell Canada's Traffic Throttling. *Ars Technica*, March 25. <http://arstechnica.com/news.ars/post/20080325-canadian-isps-furious-about-bell-canadas-traffic-throttling.html>.
- Anderson, N. 2008b. Canadian Regulators Allow P2P Throttling. *Ars Technica*, November 20. <http://arstechnica.com/news.ars/post/20081120-canadian-regulators-allow-p2p-throttling.html>.
- Anderson, S. P., and Coate, S. 2005. Market Provision of Broadcasting: A Welfare Analysis. *Review of Economic Studies* 72(4): 947–972.
- Anderson, T., Peterson, L., Shenker, S., and Turner, J. 2005a. Overcoming the Internet Impasse through Virtualization. *IEEE Computer* 38(4): 34–41.
- Anderson, T., Peterson, L., Shenker, S., and Turner, J. 2005b. *Report of NSF Workshop on Overcoming Barriers to Disruptive Innovation in Networking*.
- Aoki, M. 1999. Information and Governance in the Silicon Valley Model. Working paper, Stanford University.
- Aoki, M. 2001. *Towards a Comparative Institutional Analysis*. MIT Press.
- AOL. 2008a. AOL, Select the Plan That's Right for You!—AOL Dial-up Advantage. http://free.aol.com/thenewaol/plan_choice.adp.
- AOL. 2008b. AOL, Select the Plan That's Right for You!—AOL High-Speed Essentials. http://free.aol.com/thenewaol/plan_choice.adp.
- Apache Software Foundation. 2008a. About the Apache HTTP Server Project. http://httpd.apache.org/ABOUT_APACHE.html.
- Apache Software Foundation. 2008b. Apache History—Timeline. <http://www.apache.org/history/timeline.html>.
- Ardichvili, A., Cardozo, R., and Ray, S. 2003. A Theory of Entrepreneurial Opportunity Identification and Development. *Journal of Business Venturing* 18(1): 105–123.
- Argyres, N. S. 1996. Evidence on the Role of Firm Capabilities in Vertical Integration Decisions. *Strategic Management Journal* 17(2): 129–150.
- Armitage, G. J. 2003. Revisiting IP QoS: Why Do We Care, What Have We Learned? ACM SIGCOMM 2003 RIPQOS Workshop Report. *Computer Communication Review* 33(5): 81–88.
- Armstrong, M. 2006. Competition in Two-Sided Markets. *Rand Journal of Economics* 37(3): 668–691.

Arrow, K. J. 1962. Economic Welfare and the Allocation of Resources for Invention. *The Rate and Direction of Inventive Activity: Economic and Social Factors. A Conference of the Universities–National Bureau Committee for Economic Research and the Committee on Economic Growth of the Social Science Research Council.*

Arrow, K. J. 1974. *The Limits of Organization.* Norton.

Arthur, C. 2008. Things Are Rosy in Apple's Walled Garden—For Some. *The Guardian*, September 25.

Asia Pacific Network Information Centre. 2007. JPNIC Releases Statement on IPv4 Consumption. June 26. <http://www.europe-ipv6.net/index.php?page=news/newsroom&id=3043>

Åstebro, T. 2003. The Return to Independent Invention: Evidence of Unrealistic Optimism, Risk Seeking or Skewness Loving? *Economic Journal* 113 (January): 226–239.

Atkinson, R. D. 2007. Framing a National Broadband Policy. *CommLaw Conspectus* 16(1): 145–177.

Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X. 2002. Overview and Principles of Internet Traffic Engineering. Request for Comments 3272. IETF.

Bachmann, F., Bass, L., and Klein, M. 2002. Illuminating the Fundamental Contributors to Software Architecture Quality. Technical Report CMU/SEI-2002-TR-025, ESC-TR-2002-025, Carnegie Mellon Software Engineering Institute.

Baldwin, C. Y. 2008. Where Do Transactions Come From? Modularity, Transactions, and the Boundaries of Firms. *Industrial and Corporate Change* 17(1): 155–195.

Baldwin, C. Y., and Clark, K. B. 1997. Managing in an Age of Modularity. *Harvard Business Review* 75(5): 84–93.

Baldwin, C. Y., and Clark, K. B. 2000. *Design Rules: The Power of Modularity*, volume 1. MIT Press.

Baldwin, C. Y., and Clark, K. B. 2001. Modularity after the Crash. Working Paper 01-075, Harvard Business School.

Baldwin, C. Y., and Clark, K. B. 2002. The Technology of Design and Its Problems. Working Paper 02-076, Harvard Business School.

Baldwin, C. Y., and Clark, K. B. 2006a. Architectural Innovation and Dynamic Competition: The Smaller “Footprint” Strategy. Working Paper 07-014, Harvard Business School .

Baldwin, C. Y., and Clark, K. B. 2006b. Between “Knowledge” and “The Economy”: Notes on the Scientific Study of Designs. In B. Kahin and D. Foray, eds., *Advancing Knowledge and the Knowledge Economy.* MIT Press.

- Balkin, J. M. 2004. Digital Speech and Democratic Culture: A Theory of Freedom of Expression for the Information Society. *New York University Law Review* 79(1): 1–58.
- Balkin, J. M. 2008. Media Access: A Question of Design. *George Washington Law Review*, 76(4): 101–118.
- Bancroft, N. H., Seip, H., and Sprengel, A. 1998. *Implementing SAP R/3: How to Introduce a Large System into a Large Organization*, second edition. Manning.
- Bar, F., Cohen, S., Cowhey, P., DeLong, B., Kleeman, M., and Zysman, J. 2000. Access and Innovation Policy for the Third-Generation Internet. *Telecommunications Policy* 24(6–7): 489–518.
- Bar, F., and Riis, A. M. 2000. Tapping User-Driven Innovation: A New Rationale for Universal Service. *Information Society* 16(2): 99–108.
- Bar-Gill, O. 2006. Bundling and Consumer Misperception. *University of Chicago Law Review* 73 (winter): 33–61.
- Barlow, J. P. 1992. The Great Work. *Communications of the ACM* 35(1): 25–28.
- Barnett, W. P., and Pontikes, E. G. 2008. The Red Queen, Success Bias, and Organizational Inertia. *Management Science* 54(7): 1237–1251.
- Barney, D. 2000. *Prometheus Wired: The Hope for Democracy in the Age of Network Technology*. University of Chicago Press.
- Barney, J. 1991. Firm Resources and Sustained Competitive Advantage. *Journal of Management* 17(1): 99–120.
- Barrett, T. 2005. To Censor Pro-Union Web Site, Telus Blocked 766 Others. *The Tyee*, August 4. <http://thetyee.ca/News/2005/08/04/TelusCensor/>.
- Barzel, Y. 1982. Measurement Cost and the Organization of Markets. *Journal of Law and Economics* 25(1): 27–48.
- Bass, L., Clements, P., and Kazman, R. 1998. *Software Architecture in Practice*. Addison-Wesley Longman.
- Bass, L., Clements, P., and Kazman, R. 2003. *Software Architecture in Practice*, second edition. Addison-Wesley.
- Bass, L., Klein, M., and Bachmann, F. 2000. Quality Attribute Design Primitives. Technical Note CMU/SEI-2000-TN-017, Carnegie Mellon Software Engineering Institute.
- Basu, S., and Fernald, J. 2007. Information and Communications Technology as a General-Purpose Technology: Evidence from US Industry Data. *German Economic Review* 8(2): 146–173.
- Battelle, J. 2005. *The Search*. Penguin.

- Baumol, W. J., Cave, M., Cramton, P., Hahn, R., Hazlett, T. W., Joskow, P. L., et al. 2007. Economists' Statement on Network Neutrality Policy. Working Paper RP07-08, AEI-Brookings Joint Center .
- Beardsley, S., Doman, A., and Edin, P. 2003. Making Sense of Broadband. *McKinsey Quarterly* 2003(2): 78–87.
- Becker, G. S., and Murphy, K. M. 1993. A Simple Theory of Advertising as a Good or Bad. *Quarterly Journal of Economics* 108(4): 941–964.
- Beije, P. 1998. *Technological Change in the Modern Economy: Basic Topics and New Developments*. Elgar.
- Bell, G. 2003. Failure to Thrive: QoS and the Culture of Operational Networking. *ACM SIGCOMM Workshop on Revisiting IP QoS: What Have We Learned, Why Do We Care?*
- Bengtsson, O., Lassing, N., Bosch, J., and van Vliet, H. 2000. *Analyzing Software Architectures for Modifiability*. Research Report 1/00, Department of Software Engineering and Computer Science, University of Karlskrona.
- Benkler, Y. 2000. From Consumers to Users: Shifting the Deeper Structures of Regulation Towards Sustainable Commons and User Access. *Federal Communications Law Journal* 52(3): 561–579.
- Benkler, Y. 2002. Coase's Penguin, or, Linux and the Nature of the Firm. *Yale Law Journal* 112(3): 369–446.
- Benkler, Y. 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press.
- Berners-Lee, T., and Fischetti, M. 2000. *Weaving the Web: The Past, Present and Future of the World Wide Web by Its Inventor*. Orion Business.
- Besanko, D., Dranove, D., and Shanley, M. 2000. *Economics of Strategy*, second edition. Wiley.
- Besen, S. M., and Farrell, J. 1994. Choosing How to Compete: Strategies and Tactics in Standardization. *Journal of Economic Perspectives* 8(2): 117–131.
- Black, F., and Scholes, M. 1973. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy* 81(3): 637–654.
- Blackshaw, R. E., and Cunningham, I. M. 1980. Evolution of Open Systems Interconnection. *Fifth International Conference on Computer Communication*.
- Blumenthal, M. S., and Clark, D. D. 2001. Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World. *ACM Transactions on Internet Technology* 1(1): 70–109.

- Boase, J., Horrigan, J. B., Wellman, B., and Rainie, L. 2006. The Strength of Internet Ties. Report, Pew Internet and American Life Project.
- Bochmann, G. V., and Goyer, P. 1977. *Datagrams as a Public Packet-Switched Data Transmission Service*. Report for the Department of Communications of Canada, University of Montreal. Internet Engineering Note 17.
- Boehm, B. W., and Sullivan, K. J. 2000. Software Economics: A Roadmap. *Conference on the Future of Software Engineering. International Conference on Software Engineering*.
- Boggs, D. R., Shoch, J. F., Taft, E. A., and Metcalfe, R. M. 1980. Pup: An Internetwork Architecture. *IEEE Transactions on Communications* 28(4): 612–624.
- Bolt Beranek and Newman Inc. 1974. *Network Design Issues*. Report 2918.
- Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., and Tofanelli, P. 2007. Revealing Skype Traffic: When Randomness Plays with You. *Computer Communication Review* 37(4): 37–48.
- Border, J., Kojo, M., Griner, J., Montenegro, G., and Shelby, Z. 2001. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. Request for Comments 3135. IETF.
- Bork, R. H. 1993. *The Antitrust Paradox: A Policy at War with Itself*. Free Press.
- Bosch, J. 2000. *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. Addison-Wesley.
- Bosch, J., and Bengtsson, P. 2001. Assessing Optimal Software Architecture Maintainability. *Fifth European Conference on Software Maintenance and Reengineering*.
- Boudreau, K. J. 2006. How Open Should an Open System Be? PhD dissertation, Massachusetts Institute of Technology.
- Boudreau, K. J., Lacetera, N., and Lakhani, K. R. 2008. Parallel Search, Incentives and Problem Type: Revisiting the Competition and Innovation Link. Working Paper 09-41, Harvard Business School .
- Bowman, L. M. 2002. Google Unveils New Pay-for-Play Plan. *CNET News*, March 28. http://news.cnet.com/Google-unveils-new-pay-for-play-plan/2100-1023_3-841348.html.
- Boyle, J. 1997. Foucault in Cyberspace: Surveillance, Sovereignty, and Hardwired Censors. *University of Cincinnati Law Review* 66(1): 177–203.
- Braden, B., Faber, T., and Handley, M. 2003. From Protocol Stack to Protocol Heap—Role-Based Architecture. *Computer Communication Review* 33(1): 17–22.
- Braden, R., Clark, D., Shenker, S., and Wroclawski, J. 2000. Developing a Next-Generation Internet Architecture. Introductory Paper, NewArch Project.

- Bradner, S. 1996. The Internet Standards Process—Revision 3. Request for Comments 2026. IETF.
- Bradner, S. 2006. The End of End-to-End Security? *IEEE Security and Privacy* 4(2): 76–79.
- Brandenburger, A. M., and Nalebuff, B. J. 1998. *Co-Opetition*. Doubleday.
- Brehm, J. 1956. Postdecision Changes in Desirability of Alternatives. *Journal of Abnormal and Social Psychology* 52(3): 384–389.
- Bresnahan, T. F., and Greenstein, S. 2001. The Economic Contribution of Information Technology: Towards Comparative and User Studies. *Journal of Evolutionary Economics* 11(1): 95–118.
- Bresnahan, T. F., and Trajtenberg, M. 1995. General-Purpose Technologies: Engines of Growth. *Journal of Econometrics* 65(1): 83–108.
- Breznick, A. 2003. AOL Shifts Broadband Strategy. *Cable Datacom News*, January 1. <http://www.cabledatacomnews.com/jan03/jan03-3.html>.
- Briand, L. C., Lounis, H., and Wüst, J. 1999. Using Coupling Measurement for Impact Analysis in Object-Oriented Systems. *IEEE International Conference on Software Maintenance*.
- Briand, L. C., Wüst, J., Daly, J. W., and Victor Porter, D. 2000. Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems. *Journal of Systems and Software* 51(3): 245–273.
- Briand, L., Devanbu, P., and Melo, W. 1997. An Investigation into Coupling Measures for C++. *19th International Conference on Software Engineering*.
- Bronson, P. 1999. *The Nudist on the Late Shift*. Random House.
- Brooker, K. 1999. No Sales? No Profits? No Problem. *Fortune* 139(3): 163–169.
- Brusoni, S. 2005. The Limits to Specialization: Problem Solving and Coordination in ‘Modular Networks’. *Organization Studies* 26(12): 1885–1907.
- Brusoni, S., and Prencipe, A. 2001. Unpacking the Black Box of Modularity: Technologies, Products and Organizations. *Industrial and Corporate Change* 10(1): 179–205.
- Brynjolfsson, E. 1996. The Contribution of Information Technology to Consumer Welfare. *Information Systems Research* 7(3): 281–300.
- Brynjolfsson, E., and Hitt, L. M. 2000. Beyond Computation: Information Technology, Organizational Transformation and Business Performance. *Journal of Economic Perspectives* 14(4): 23–48.
- Brynjolfsson, E., and Hitt, L. M. 2003. Computer Productivity: Firm-Level Evidence. *Review of Economics and Statistics* 85(4): 793–808.

- Brynjolfsson, E., Hu, Y. J., and Smith, M. D. 2003. Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers. *Management Science* 49(11): 1580–1596.
- Brynjolfsson, E., Hu, Y. J., and Smith, M. D. 2006. From Niches to Riches: Anatomy of the Long Tail. *MIT Sloan Management Review* 47(4): 67–71.
- Brynjolfsson, E., Hu, Y. J., and Simester, D. 2006. Goodbye Pareto Principle, Hello Long Tail: The Effect of Search Costs on the Concentration of Product Sales. Working paper, MIT Center for Digital Business.
- Brynjolfsson, E., and Saunders, A. 2009. *Wired for Innovation: How Information Technology Is Reshaping the Economy*. MIT Press.
- Buchanan, J. M., and Yoon, Y. J. 2000. Symmetric Tragedies: Commons and Anticommons. *Journal of Law and Economics* 43(1): 1–13.
- Buckley, S. 2007. AT&T's Converged IMS Vision. *Telecommunications Online*. October 5. http://www.telecommagazine.com/article.asp?HH_ID=AR_3566.
- Buckman, R. 2005. Many Internet Start-ups Are Telling Venture Capitalists: "We Don't Need You." *Wall Street Journal*, October 31.
- Burgstahler, L., Dolzer, K., Hauser, C., Jähnert, J., Junghans, S., Macián, C., et al. 2003. Beyond Technology: the Missing Pieces for QoS Success. *ACM SIGCOMM Workshop on Revisiting IP QoS: What Have We Learned, Why Do We Care?*
- Burk, D. L., and Cohen, J. E. 2001. Fair Use Infrastructure for Rights Management Systems. *Harvard Journal of Law and Technology* 15(1): 41–92.
- Burness, A.-L., Eardley, P., Akhtar, N., Callejo, M. A., and Colas, J. A. 2005. Making Migration Easy: A Key Requirement for Systems Beyond 3G. *IEEE 61st Vehicular Technology Conference*.
- Burnham, T., Frels, J., and Mahajan, V. 2003. Consumer Switching Costs: A Typology, Antecedents, and Consequences. *Journal of the Academy of Marketing Science* 31(2): 109–126.
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. 1996. *Pattern-Oriented Software Architecture. A System of Patterns*, volume 1. Wiley.
- Bushee, B. J. 2001. Do Institutional Investors Prefer Near-Term Earnings over Long-Run Value? *Contemporary Accounting Research* 18(2): 207–246.
- Business Wire, 1999. EarthLink Signs GoTo.com as Its Default Search Provider: Two-Year, \$10 Million Premiere Partnership Puts GoTo.com on EarthLink's Top Web Properties. *Business Wire*, October 14. http://findarticles.com/p/articles/mi_m0EIN/is_1999_Oct_14/ai_56280327.
- Calomiris, C. W. 2000. *U.S. Bank Deregulation in Historical Perspective*. Cambridge University Press.

- Camarillo, G., and Garcia-Martin, M. A. 2008. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*, third edition. Wiley.
- Camerer, C. F., Loewenstein, G., and Rabin, M., eds. 2004. *Advances in Behavioral Economics*. Princeton University Press.
- Carlton, D. W. 2001. A General Analysis of Exclusionary Conduct and Refusal to Deal: Why Aspen and Kodak Are Misguided. *Antitrust Law Journal* 68(3): 659–683.
- Carlton, D. W., and Gertner, R. H. 2003. Intellectual Property, Antitrust and Strategic Behavior. In A. Jaffe, J. Lerner, and S. Stern, eds., *Innovation Policy and the Economy*, volume 3. MIT Press.
- Carlton, D. W., and Perloff, J. M. 2005. *Modern Industrial Organization*, fourth edition. Pearson.
- Carlton, D. W., and Waldman, M. 2000. The Strategic Use of Tying to Preserve and Create Market Power in Evolving Industries. Working paper, University of Chicago.
- Carlton, D. W., and Waldman, M. 2002. The Strategic Use of Tying to Preserve and Create Market Power in Evolving Industries. *Rand Journal of Economics* 33(2): 194–220.
- Carney, W. J. 1997. Large Bank Stockholders in Germany: Saviors or Substitutes? *Journal of Applied Corporate Finance* 9(4): 74–80.
- Carpenter, B. 1996. Architectural Principles of the Internet. Request for Comments 1958. IETF.
- Carpenter, B. 2000. Internet Transparency. Request for Comments 2775. IETF.
- Carpenter, B., and Brim, S. 2002. Middleboxes: Taxonomy and Issues. Request for Comments 3234. IETF.
- Cassidy, J. 2006. Me Media. *The New Yorker*, May 15. http://www.newyorker.com/archive/2006/05/15/060515fa_fact_cassidy.
- Castilla, E. J., Hwang, H., Granovetter, E., and Granovetter, M. 2000. Social Networks in Silicon Valley. In C.-M. Lee, W. Miller, M. Hancock, and H. Rowen, eds., *The Silicon Valley Edge: A Habitat for Innovation and Entrepreneurship*. Stanford University Press.
- Cave, M., and Crocioni, P. 2007. Does Europe Need Network Neutrality Rules? *International Journal of Communication* 1, 669–679.
- CBC News. 2005. Telus Cuts Subscriber Access to Pro-Union Website. *CBC News*, June 9. <http://www.cbc.ca/story/canada/national/2005/07/24/telus-sites050724.html>.
- Cerf, V. G. 1977. TCP (Version 2) Specification. Internet Engineering Note 5.
- Cerf, V. G. 1978a. A Proposal for TCP Version 3.1 Header Format. Internet Engineering Note 27.

- Cerf, V. G. 1978b. A Proposed New Internet Header Format. Internet Engineering Note 26.
- Cerf, V. G. 1980a. Final Report of the Stanford University TCP Project. Internet Engineering Note 151.
- Cerf, V. G. 1980b. Protocols for Interconnected Packet Networks. *Computer Communication Review* 10(4): 10–11.
- Cerf, V. G., and Kahn, R. E. 1974. A Protocol for Packet Network Intercommunication. *Transactions on Communications* 22(5): 637–648.
- Cerf, V. G., and Kirstein, P. T. 1978. Issues in Packet-Network Interconnection. *Proceedings of the IEEE* 66(11): 1386–1408.
- Cerf, V. G., and Postel, J. B. 1978. Specification of Internetwork Transmission Control Program. TCP. Version 3. Internet Engineering Note 21.
- Chafkin, M. 2008. Anything Could Happen. *Inc.*, March 1. <http://www.inc.com/magazine/20080301/anything-could-happen.html>.
- Charny, B. 2003. Why VoIP Is Music to Kazaa's Ear. *CNET News*, September 11. http://www.news.com/Why-VoIP-is-music-to-Kazaas-ear/2008-1082_3-5074558.html.
- Charny, B. 2005a. Mexico Telephone Operator under VoIP Fire. *ZDNet News*, April 25. http://news.zdnet.com/2100-1035_22-5681542.html.
- Charny, B. 2005b. Vonage Says Broadband Provider Blocks Its Calls. *CNET News*, February 14. http://news.cnet.com/Vonage-says-broadband-provider-blocks-its-calls/2100-7352_3-5576234.html.
- Cherry, S. 2005. The VOIP Backlash. *IEEE Spectrum*, 42(10): 61–63.
- Chesbrough, H. W. 2003. *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business School Press.
- Chesbrough, H. W. 2006. *Open Business Models: How to Thrive in the New Innovation Landscape*. Harvard Business School Press.
- Chesbrough, H. W., and Teece, D. J. 1996. When Is Virtual Virtuous? Organizing for Innovation. *Harvard Business Review* 74(1): 65–73.
- Chesbrough, H., Vanhaverbeke, W., and West, J., eds. 2006. *Open Innovation: Researching a New Paradigm*. Oxford University Press.
- Cheswick, W. R., Bellovin, S. M., and Rubin, A. D. 2003. *Firewalls and Internet Security: Repelling the Wily Hacker*, second edition. Addison-Wesley.
- Chiappa, J. N. 2002. Will the Real "End-End Principle" Please Stand Up? http://mercury.lcs.mit.edu/~jnc/tech/end_end.html
- Chirico, F., van der Haar, I., and Larouche, P. 2007. Network Neutrality in the EU. TILEC Discussion Paper DP 2007-030, Tilburg University.

- Choi, J. P., and Stefanadis, C. 2001. Tying, Investment, and the Dynamic Leverage Theory. *Rand Journal of Economics* 32(1): 52–71.
- Christensen, C. M. 2000. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harper Business.
- Christensen, C. M., and Raynor, M. E. 2003. *The Innovator's Solution: Creating and Sustaining Successful Growth*. Harvard Business School Press.
- Christensen, C. M., and Rosenbloom, R. S. 1995. Explaining the Attacker's Advantage: Technological Paradigms, Organizational Dynamics, and the Value Network. *Research Policy* 24(2): 233–257.
- Christensen, C. M., Verlinden, M., and Westerman, G. 2002. Disruption, Disintegration and the Dissipation of Differentiability. *Industrial and Corporate Change* 11(5): 955–993.
- chromatic and Sheppard, D. 2008. A Beginner's Introduction to Perl 5.10. April 23. <http://www.perl.com/pub/a/2008/04/23/a-beginners-introduction-to-perl-510.html>.
- Cisco Systems. 2005. Network-Based Application Recognition and Distributed Network-Based Application Recognition. Documentation. <http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t8/dtnbarad.pdf>.
- Cisco Systems. 2006. *Supporting the IP Multimedia Subsystem for Mobile, Wireline, and Cable Providers*. White paper.
- Clark, D. D. 1985. The Structuring of Systems Using Upcalls. *Tenth ACM Symposium on Operating Systems Principles*.
- Clark, D. D. 1988. The Design Philosophy of the DARPA Internet Protocols. *Computer Communication Review*, 18(4): 106–114.
- Clark, D. D. 1997. Interoperation, Open Interfaces, and Protocol Architecture. In *White Papers: The Unpredictable Certainty: Information Infrastructure through 2000*. National Academy Press.
- Clark, D. D., and Blumenthal, M. S. 2007. The End-to-End Argument and Application Design: The Role of Trust. Paper presented at 35th Research Conference on Communication, Information and Internet Policy.
- Clark, D. D., Pograd, K. T., and Reed, D. P. 1978. An Introduction to Local Area Networks. *Proceedings of the IEEE* 66(11): 1497–1517.
- Clark, D. D., Shenker, S., and Falk, A. 2007. GENI Research Plan (Version 4.5). GENI Research Coordination Working Group and GENI Planning Group. GDD 06-28

- Clark, D. D., Sollins, K., Wroclawski, J., and Faber, T. 2003. Addressing Reality: An Architectural Response to Real-World Demands on the Evolving Internet. *ACM SIGCOMM Workshop on Revisiting IP QoS: Why Do We Care, What Have We Learned?*
- Clark, D. D., Sollins, K., Wroclawski, J., Katabi, D., Kulik, J., Yang, X., et al. 2003. *NewArch: Future Generation Internet Architecture*. Final Technical Report, NewArch Project.
- Clark, D. D., and Tennenhouse, D. L. 1990. Architectural Considerations for a New Generation of Protocols. *ACM Symposium on Communications Architectures and Protocols*.
- Clark, D. D., Wroclawski, J., Sollins, K. R., and Braden, R. 2002. Tussle in Cyberspace: Defining Tomorrow's Internet. *Computer Communication Review* 32(4): 347–356.
- Clark, D. D., Wroclawski, J., Sollins, K. R., and Braden, R. 2005. Tussle in Cyberspace: Defining Tomorrow's Internet. *IEEE/ACM Transactions on Networking* 13(3): 462–475.
- Clark, K. B. 1985. The Interaction of Design Hierarchies and Market Concepts in Technological Evolution. *Research Policy* 14(5): 235–251.
- Clark, K. B. 1988. Managing Technology in International Competition: The Case of Product Development in Response to Foreign Entry. In A. Spence and H. Hazard, eds., *International Competitiveness*. Ballinger.
- Coase, R. 1937. The Nature of the Firm. *Economica* 4(16): 386–405.
- Coase, R. H. 1960. The Problem of Social Cost. *Journal of Law and Economics* 3(1): 1–44.
- Cohen, A. 2002. *The Perfect Store: Inside eBay*. Little, Brown.
- Cohen, D. 1979. On Interconnection of Computer Networks. In K. Beauchamp, ed., *Interlinking of Computer Networks: Proceedings of the NATO Advanced Study Institute held at Bonas, France, August 28–September 8, 1978*. Reidel.
- Cohen, J. E., and Lemley, M. A. 2001. Patent Scope and Innovation in the Software Industry. *California Law Review* 89(1): 1–57.
- Cohen, W. M., and Klepper, S. 1992. The Tradeoff between Firm Size and Diversity in the Pursuit of Technological Progress. *Small Business Economics* 4(1): 1–14.
- Cohen, W. M., and Levinthal, D. A. 1990. Absorptive Capacity: A New Perspective on Learning and Innovation. *Administrative Science Quarterly* 35(1): 128–152.
- Cohen, W. M., and Malerba, F. 2001. Is the Tendency to Variation a Chief Cause of Progress? *Industrial and Corporate Change* 10(3): 587–608.
- Combs, J. G., and Ketchen, D. J. 1999. Explaining Interfirm Cooperation and Performance: Toward a Reconciliation of Predictions from the Resource-Based View and Organizational Economics. *Strategic Management Journal* 20(9): 867–888.

Comer, D. E. 2000. *Internetworking with TCP/IP: Principles, Protocols, and Architectures*, fourth edition. Prentice-Hall.

Computer Science and Telecommunications Board and National Research Council. 1994. *Realizing the Information Future: The Internet and Beyond*. National Academy Press.

Computer Science and Telecommunications Board and National Research Council. 1996. *The Unpredictable Certainty: Information Infrastructure through 2000*. National Academy Press.

Computer Science and Telecommunications Board and National Research Council. 2001. *The Internet's Coming of Age*. National Academy Press.

comScore. 2008. comScore Releases February 2008 U.S. Search Engine Rankings. Press release, March 19. <http://www.comscore.com/press/release.asp?press=2119>.

Consumers Union. 2007. The Next Big Thing in Broadband: Early Termination Penalties. HearUsNow.org Blog. April 9. http://www.consumersunion.org/blogs/hun/2007/04/the_next_big_thing_in_broadban.html.

Cooter, R. D. 1998. Expressive Law and Economics. *Journal of Legal Studies* 27(S2): 585–608.

Coulson, G., Blair, G. S., Davies, N., Robin, P., and Fitzpatrick, T. 1999. Supporting Mobile Multimedia Applications through Adaptive Middleware. *IEEE Journal on Selected Areas in Communications* 17(9): 1651–1659.

Crane, P. 2007. A New Service Infrastructure Architecture. *BT Technology Journal* 23(1): 15–27.

Craswell, R. 1982. Tying Requirements in Competitive Markets: The Consumer Protection Issues. *Boston University Law Review* 62(3): 661–700.

Crawford, S. C. 2007. The Internet and the Project of Communications Law. *UCLA Law Review* 55(2): 359–407.

Crocker, D. undated. Email History: How Email Was Invented. <http://livinginternet.com/e/ei.htm>.

Cuevas, A., Moreno, J., Vidales, P., and Einsiedler, H. 2006. The IMS Service Platform: A Solution for Next Generation Network Operators to Be More Than Bit Pipes. *IEEE Communications Magazine*, 44(8): 75–81.

Cusumano, M. A. 1991. *Japan's Software Factories: A Challenge to U.S. Management*. Oxford University Press.

Cuthbertson, K., Hayes, S., and Nitzsche, D. 1997. The Behaviour of UK Stock Prices and Returns: Is the Market Efficient? *Economic Journal* 107(443): 986–1008.

- Dasgupta, P., and Maskin, E. 1987. The Simple Economics of Research Portfolios. *Economic Journal* 97(387): 581–595.
- David, P. A. 1990. The Dynamo and the Computer: An Historical Perspective on the Modern Productivity Paradox. *American Economic Review* 80(2): 355–361.
- David, P. A. 2001. The Evolving Accidental Information Super-highway. *Oxford Review of Economic Policy* 17(2): 159–187.
- DeGraba, P. 1996. Why Lever into a Zero-Profit Industry: Tying, Foreclosure, and Exclusion. *Journal of Economics and Management Strategy* 5(3): 433–447.
- Deibert, R., Palfrey, J., Rohozinski, R., and Zittrain, J. L., eds. 2008. *Access Denied: The Practice and Policy of Global Internet Filtering*. MIT Press.
- DellaVigna, S., and Malmendier, U. 2004. Contract Design and Self-Control Theory and Evidence. *Quarterly Journal of Economics* 119(2): 353–402.
- Demsetz, H. 1988. The Theory of the Firm Revisited. *Journal of Law Economics and Organization* 4(1): 141–162.
- Denton, N. 2007. An Alternate History According to Elon Musk. *Valleywag*, January 19. <http://valleywag.com/tech/paypal/an-alternate-history-according-to-elon-musk-230076.php>.
- Deprez, F., Rosengren, J., and Soman, V. 2002. Portals for All Platforms. *McKinsey Quarterly* 2002(1): 93–101.
- Deutsche Telekom. 2008. Allgemeine Geschäftsbedingungen und Preise: Bundesliga Insider.
- Deutschman, A. 1997. Imposter Boy. *Gentlemen's Quarterly*. January. <http://web.archive.org/web/20030212202753/http://www.chrispy.net/marca/gqarticle.html>.
- Diamond, J. M. 1997. *Guns, Germs, and Steel: The Fates of Human Societies*. Norton.
- DiBona, C., Ockman, S., and Stone, M., eds. 1999. *OpenSources: Voices from the Open Source Revolution*. O'Reilly.
- DiMaggio. 1998. The New Institutionalism: Avenues of Collaboration. *Journal of Institutional and Theoretical Economics* 154(4): 696–715.
- DiMaggio, P. J., and Powell, W. W. 1983. The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields. *American Sociological Review* 48(2): 147–160.
- Diot, C., Levine, B. N., Lyles, B., Kassem, H., and Balensiefen, D. 2000. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network* 14(1): 78–88.
- Dixit, A. K., and Pindyck, R. S. 1994. *Investment under Uncertainty*. Princeton University Press.

- Dosi, G., and Nelson, R. R. 1994. An Introduction to Evolutionary Theories in Economics. *Journal of Evolutionary Economics* 4(3): 153–172.
- Drezner, D. W., and Farrell, H. 2008. Blogs, Politics and Power: A Special Issue of Public Choice. *Public Choice* 134(1–2): 1–13.
- Dunbar, J. 2007. Pulling Plug on Net Service Not Easy. *USA Today*, April 9. http://www.usatoday.com/tech/techinvestor/industry/2007-04-09-net-fees_N.htm?POE=TECISVA.
- Dutcher, B. 2001. *The NAT Handbook: Implementing and Managing Network Address Translation*. Wiley.
- DW Staff. 2006. Bundesliga Agrees on Massive Sponsorship Deal with T-Com. *Deutsche Welle*, June 8. <http://www.dw-world.de/dw/article/0,2144,2046790,00.html>.
- Dyer, J. H. 1996. Specialized Supplier Networks as a Source of Competitive Advantage: Evidence from the Auto Industry. *Strategic Management Journal* 17(4): 271–291.
- Dyer, J. H. 1997. Effective Interfirm Collaboration: How Firms Minimize Transaction Costs and Maximize Transaction Value. *Strategic Management Journal* 18(7): 535–556.
- Easterbrook, F. H. 1997. International Corporate Differences: Markets or Law? *Journal of Applied Corporate Finance* 9(4): 23–29.
- Eckersley, P., von Lohmann, F., and Schoen, S. 2007. *Packet Forgery by ISPs: A Report on the Comcast Affair*. Electronic Frontier Foundation. November 28. http://www.eff.org/files/eff_comcast_report2.pdf
- Economides, N., and Tåg., J. 2007. Net Neutrality on the Internet: A Two-sided Market Analysis. Working Paper 07-45, NET Institute.
- Edwards, J. S. S., and Fischer, K. 1994. *Banks, Finance and Investments in Germany*. Cambridge University Press.
- Eggert, L., and Eddy, W. M. 2006. Towards More Expressive Transport-Layer Interfaces. *First ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*.
- Einhorn, M. A. 2003. US Settlement Reform: An Historical Review. In G. Madden, ed., *Traditional Telecommunications Networks*, volume 1. Elgar.
- Elhauge, E., and Geradin, D. 2007. *Global Antitrust Law and Economics*. Foundation Press.
- Ellickson, R. C. 1991. *Order Without Law: How Neighbours Settle Disputes*. Harvard University Press.

- Enkel, E., Perez-Freije, J., and Gassmann, O. 2005. Minimizing Market Risks through Customer Integration in New Product Development: Learning from Bad Practice. *Creativity and Innovation Management* 14(4): 425–437.
- Eppinger, S. D. 1991. Model-Based Approaches to Managing Concurrent Engineering. *Journal of Engineering Design* 2(4): 283–290.
- Erdogmus, H., Boehm, B. W., Harrison, W., Reifer, D. J., and Sullivan, K. J. 2002. Software Engineering Economics: Background, Current Practices, and Future Directions. *24rd International Conference on Software Engineering*.
- Eubanks, M. 2006. Multicast Business Models (PowerPoint Presentation). Multicast Workshop, SANOG 7.
- Evans, D. S., Hagiu, A., and Schmalensee, R. 2006. *Invisible Engines: How Software Platforms Drive Innovation and Transform Industries*. MIT Press.
- Evans, D. S., and Noel, M. D. 2007. Defining Markets That Involve Multi-Sided Platform Businesses: An Empirical Framework with an Application to Google's Purchase of DoubleClick. Working Paper 07-18, Reg-Markets Center.
- Evans, D. S., and Schmalensee, R. L. 2002. Some Economic Aspects of Antitrust Analysis in Dynamically Competitive Industries. In A. Jaffe, J. Lerner, and S. Stern, eds., *Innovation Policy and the Economy*, volume 2. MIT Press.
- Fake, C. 2007. The History of Flickr. *IT Conversations*, February 12. <http://itc.conversationsnetwork.org/shows/detail1755.html>.
- Farrell, H., and Drezner, D. W. 2008. The Power and Politics of Blogs. *Public Choice* 134(1–2): 15–30.
- Farrell, J. 2006. Open Access Arguments: Why Confidence Is Misplaced. In T. Lenard and R. May, eds., *Net Neutrality or Net Neutering: Should Broadband Internet Services Be Regulated?* Springer.
- Farrell, J., Gilbert, R. J., and Katz, M. L. 2002. Market Structure, Organizational Structure, and R&D Diversity. In R. Arnott, B. Greenwald, R. Kanbur, and B. Nalebuff, eds., *Economics for an Imperfect World: Essays in Honor of Joseph E. Stiglitz*. MIT Press.
- Farrell, J., and Katz, M. L. 2000. Innovation, Rent Extraction, and Integration in Systems Markets. *Journal of Industrial Economics* 48(4): 413–432.
- Farrell, J., and Klemperer, P. 2007. Coordination and Lock-in: Competition and Switching Costs and Network Effects. In M. Armstrong and R. Porter, eds., *Handbook of Industrial Organization*, volume 3. Elsevier.
- Farrell, J., and Weiser, P. J. 2003. Modularity, Vertical Integration, and Open Access Policies: Towards a Convergence of Antitrust and Regulation in the Internet Age. *Harvard Journal of Law and Technology* 17(1): 85–134.

- Faulhaber, G. 2002. Network Effects and Merger Analysis: Instant Messaging and the AOL-Time Warner Case. *Telecommunications Policy* 26(5-6): 311-333.
- Faulhaber, G. R. 2007. Network Neutrality: The Debate Evolves. *International Journal of Communication* 1: 680-700.
- Feamster, N., Gao, L., and Rexford, J. 2007. How to Lease the Internet in Your Spare Time. *Computer Communication Review* 37(1): 61-64.
- Federal Communications Commission. 2001. Applications for Consent to the Transfer of Control of Licenses and Section 214 Authorizations by Time Warner Inc. and America Online, Inc., Transferors, to AOL Time Warner Inc., Transferee. Memorandum Opinion and Order. FCC 01-12. *Federal Communications Commission Record* 16: 6547-6612.
- Federal Communications Commission. 2005a. Appropriate Framework for Broadband Access to the Internet over Wireline Facilities. Policy Statement. *Federal Communications Commission Record* 20: 14986-14988.
- Federal Communications Commission. 2005b. Madison River Communications, LLC and Affiliated Companies. Consent Decree. *Federal Communications Commission Record* 20: 4296-4299.
- Federal Communications Commission. 2005c. Madison River Communications, LLC and Affiliated Companies. Order. *Federal Communications Commission Record* 20: 4295-4295.
- Federal Communications Commission. 2007. International Settlements Policy and U.S.-International Accounting Rates. *International Bureau*, February 20. <http://www.fcc.gov/ib/pd/pf/account.html>.
- Federal Communications Commission. 2008a. Formal Complaint of Free Press and Public Knowledge Against Comcast for Secretly Degrading Peer-to-Peer Applications. Memorandum Opinion and Order. FCC 08-183.
- Federal Communications Commission. 2008b. *High-Speed Services for Internet Access: Status as of June 30, 2007*. Report.
- Federal Trade Commission. 2008. Commission Announces Revised Jurisdictional Thresholds for Section 8 and Section 7A of the Clayton Act. January 18. <http://www.ftc.gov/opa/2008/01/clayton.shtm>.
- Feller, J., and Fitzgerald, B. 2000. A Framework Analysis of the Open Source Software Development Paradigm. *Twenty First International Conference on Information Systems*.
- Felten, E. W. 2003. Layers. *Freedom to Tinker*, June 18 (updated September 11, 2003). <http://www.freedom-to-tinker.com/blog/felten/layers>.
- Ferguson, C. H., and Morris, C. R. 1993. *Computer Wars: How the West Can Win in a Post-IBM World*. Times Books.

- Festa, P. 2002. GoTo Gains Amid Web Pains. *CNET News*, January 2. http://news.cnet.com/GoTo-gains-amid-Web-pains/2100-1023_3-257328.html.
- Fielding, R. T., and Taylor, R. N. 2002. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology* 2(2): 115–150.
- Fine, C. H. 1998. *Clockspeed: Winning Industry Control in the Age of Temporary Advantage*. Perseus Books.
- Fisher, W. W., III. 1988. Reconstructing the Fair Use Doctrine. *Harvard Law Review* 101(8): 1659–1795.
- Fisher, W. W., III. 2001. Theories of Intellectual Property. In S. Munzer, ed., *New Essays in the Legal and Political Theory of Property*. Cambridge University Press.
- Fitzgerald, M. 2006. How We Did It: Stewart Butterfield and Caterina Fake, Co-founders, Flickr. *Inc.*, December. <http://www.inc.com/magazine/20061201/hidi-butterfield-fake.html>.
- Fixson, S. K. 2001. Three Perspectives on Modularity—A Literature Review of a Product Concept for Assembled Hardware Products. Working Paper EDS-WP-2001-06, Engineering Systems Division, MIT.
- Fixson, S. K., and Park, J.-K. 2008. The Power of Integrality: Linkages between Product Architecture, Innovation, and Industry Structure. *Research Policy* 37(8): 1296–1316.
- Fixson, S., and Sako, M. 2001. Modularity in Product Architecture: Will the Auto Industry Follow the Computer Industry? Paper presented at Annual Meeting of International Motor Vehicle Program.
- Fligstein, N., and Freeland, R. 1995. Theoretical and Comparative Perspectives on Corporate Organization. *Annual Review of Sociology* 21: 21–43.
- Floyd, S. 1998. Internet Research: Comments on Formulating the Problem. Unpublished.
- Floyd, S. 2002. General Architectural and Policy Considerations. Request for Comments 3426. IETF.
- Franke, N., and Shah, S. 2003. How Communities Support Innovative Activities: An Exploration of Assistance and Sharing among End-Users. *Research Policy* 32(1): 157–178.
- Franke, N., and von Hippel, E. 2003. Satisfying Heterogeneous User Needs Via Innovation Toolkits: The Case of Apache Security Software. *Research Policy* 32(7): 1199–1215.
- Franks, J., and Mayer, C. 1997. Corporate Ownership and Control in the U.K., Germany, and France. *Journal of Applied Corporate Finance* 9(4): 30–45.

Frederick, S., Loewenstein, G., and O'Donoghue, T. 2002. Time Discounting and Time Preference: A Critical Review. *Journal of Economic Literature* 40(2): 351–401.

Free Press, Public Knowledge, Media Access Project, Consumer Federation of America, Consumers Union, Information Society Project at Yale Law School, et al. 2007. Petition for Declaratory Ruling to Federal Communications Commission. WC Dkt. No. 07–52. November 1.

Free Press and Public Knowledge. 2007. Formal Complaint to Federal Communications Commission. November 1.

Freed, N. 2000. Behavior of and Requirements for Internet Firewalls. Request for Comments 2979. IETF.

Freeman, C., and Louca, F. 2001. *As Time Goes By: From the Industrial Revolutions to the Information Revolution*. Oxford University Press.

Freixas, X., Guesnerie, R., and Tirole, J. 1985. Planning under Incomplete Information and the Ratchet Effect. *Review of Economic Studies* 52(2): 173–191.

Fried, J. 2006. Bezos Expeditions Invests in 37signals. *Signal vs. Noise*, July 20. http://www.37signals.com/svn/archives2/bezos_expeditions_invests_in_37signals.php.

Fried, J. 2008a. Ask 37signals: Does “Getting Real” Work in This Economy? *Signal vs. Noise*, November 14. <http://www.37signals.com/svn/posts/1398-ask-37signals-does-getting-real-work-in-this-economy>.

Fried, J. 2008b. Start a Business, Not a Startup. *Signal vs. Noise*, April 25. <http://www.37signals.com/svn/posts/997-start-a-business-not-a-startup>.

Frischmann, B. M. 2005. An Economic Theory of Infrastructure and Commons Management. *Minnesota Law Review* 89 (April): 917–1030.

Frischmann, B. M. 2008. Speech, Spillovers, and the First Amendment. *University of Chicago Legal Forum*, 2008: 301–333.

Frischmann, B. M., and Lemley, M. A. 2006. Spillovers. *Columbia Law Review* 100(2): 101–143.

Frischmann, B. M., and van Schewick, B. 2007. Network Neutrality and the Economics of an Information Superhighway: A Reply to Professor Yoo. *Jurimetrics Journal* 47 (summer): 383–428.

Furubotn, E. G., and Richter, R. 2005. *Institutions and Economic Theory: The Contribution of the New Institutional Economics*, second edition. University of Michigan Press.

Gabbay, N. 2006. Flickr Case Study: Still about Tech for Exit? August 27. <http://www.startup-review.com/blog/flickr-case-study-still-about-tech-for-exit.php>.

Galant, G. 2005. Venture Voice Show #7—Evan Williams of Odeo. *Venture Voice*, July 26. http://www.venturevoice.com/2005/07/vv_show_7_-_eva.html.

- Ganjam, A., and Zhang, H. 2005. Internet Multicast Video Delivery. *Proceedings of the IEEE* 93(1): 159–170.
- Gans, J. S., Hsu, D. H., and Stern, S. 2002. When Does Start-Up Innovation Spur the Gale of Creative Destruction? *Rand Journal of Economics* 33(4): 571–586.
- Gans, J. S., and Stern, S. 2000. Incumbency and R&D Incentives: Licensing the Gale of Creative Destruction. *Journal of Economics and Management Strategy* 9(4): 485–511.
- Gans, J. S., and Stern, S. 2003. The Product Market and the Market for “Ideas”: Commercialization Strategies for Technology Entrepreneurs. *Research Policy* 32(2): 333–350.
- Gapper, J. 2008. Haze May Obscure Microsoft’s Path. *FT.com*, February 6. http://www.ft.com/cms/s/0/27a2794e-d4d9-11dc-9af1-0000779fd2ac.html?nclick_check=1.
- Garber, L. 1999. Steve Deering: Multicast Waits for Its Killer App. *IT Professional* 1(2): 78–80.
- Garlick, L. L., Rom, R., and Postel, J. B. 1977. Issues in Reliable Host-to-Host Protocols. Internet Engineering Note 12.
- Garrett, J. J. 2005. An Interview with Flickr’s Eric Costello. *adaptive path*, August 4. <http://adaptivepath.com/ideas/essays/archives/000519.php>.
- Garud, R., and Kumaraswamy, A. 1993. Changing Competitive Dynamics in Network Industries: An Exploration of Sun Microsystems Open Systems Strategy. *Strategic Management Journal* 14(5): 351–369.
- Garud, R., and Kumaraswamy, A. 1995. Technological and Organizational Designs for Realizing Economies of Substitution. *Strategic Management Journal* 16 (special issue, summer): 93–109.
- Gawer, A. 2000. The Organization of Platform Leadership: An Empirical Investigation of Intel’s Management Processes Aimed at Fostering Complementary Innovation by Third Parties. PhD dissertation, Massachusetts Institute of Technology.
- Gawer, A., and Cusumano, M. A. 2002. *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Harvard Business School Press.
- Gawer, A., and Henderson, R. 2007. Platform Owner Entry and Innovation in Complementary Markets: Evidence from Intel. *Journal of Economics and Management Strategy* 16(1): 1–34.
- Gaynor, M. 2001. The Effect of Market Uncertainty on the Management Structure for Network-Based Services. PhD dissertation, Harvard University.
- Gaynor, M. 2003. *Network Services Investment Guide: Maximizing ROI in Uncertain Times*. Wiley.

Gaynor, M., Bradner, S., Iansiti, M., and Kung, H. T. 2000. The Real Options Approach to Network-Based Service Architecture. <http://people.bu.edu/mgaynor/papers/IEEE-special.pdf>.

Geist, M. 2005. *Telecommunications Policy Review Submission*. University of Ottawa Faculty of Law.

Gilbert, R. J. 1989. Mobility Barriers and the Value of Incumbency. In R. Schmalensee and R. Willig, eds., *Handbook of Industrial Organization*, volume 1. Elsevier.

Gilbert, R., and Katz, M. L. 2001. An Economist's Guide to US v. Microsoft. *Journal of Economic Perspectives* 15(2): 25–44.

Gillespie, T. 2006. Engineering a Principle: 'End-to-End' in the Design of the Internet. *Social Studies of Science* 36(3): 427–457.

Gillett, S. E., Lehr, W. H., Wroclawski, J. T., and Clark, D. D. 2001. Do Appliances Threaten Internet Innovation? *IEEE Communications Magazine* 39(10): 46–51.

Gligor, V. D. 2005. Guaranteeing Access in Spite of Distributed Service-Flooding Attacks. In B. Christianson et al., eds., *Security Protocols*. Springer.

Gompers, P. A., and Lerner, J. 2000. The Determinants of Corporate Venture Capital Success: Organizational Structure, Incentives, and Complementarities. In R. Morck, ed., *Concentrated Corporate Ownership*. University of Chicago Press.

Gompers, P. A., and Lerner, J. 2001a. *The Money of Invention: How Venture Capital Creates New Wealth*. Harvard Business School Press.

Gompers, P. A., and Lerner, J. 2001b. The Venture Capital Revolution. *Journal of Economic Perspectives* 15(2): 145–168.

Gompers, P. A., and Lerner, J. 2002. Short-Term America Revisited? Boom and Bust in the Venture Capital Industry and the Impact on Innovation. In A. Jaffe, J. Lerner, and S. Stern, eds., *Innovation Policy and the Economy*, volume 3. MIT Press.

Gompers, P., and Lerner, J. 2004. *The Venture Capital Cycle*. MIT Press.

Gompers, P., and Lerner, J. 2006. *The Venture Capital Cycle*, second edition. MIT Press.

Google. 1999. Google Receives \$25 Million in Equity Funding. Press release, June 7. <http://www.google.com/press/pressrel/pressrelease1.html>

Google. 2000. Google's Targeted Keyword Ad Program Shows Strong Momentum with Advertisers. Press release, August 16. <http://www.google.com/press/pressrel/pressrelease31.html>.

Google. 2007. Google Intends to Bid in Spectrum Auction If FCC Adopts Consumer Choice and Competition Requirements. Press release, July 20, 2007. http://www.google.com/intl/en/press/pressrel/20070720_wireless.html.

- Google. 2008. *Form 10-K for the Fiscal Year Ended December 31, 2007*.
- Gordon, R. J. 2000. Does the "New Economy" Measure up to the Great Inventions of the Past? *Journal of Economic Perspectives* 14(4): 49–74.
- Granovetter, M. 1985. Economic Action and Social Structure: The Problem of Embeddedness. *American Journal of Sociology* 91(3): 481–510.
- Green, J. R., and Scotchmer, Suzanne. 1995. On the Division of Profit in Sequential Innovation. *Rand Journal of Economics* 26(1): 20–33.
- Greenstein, S. 2007a. *Economic Experiments and Neutrality in Internet Access*. Working Paper 13158, National Bureau of Economic Research.
- Greenstein, S. 2007b. Economic Experiments in Internet Access Markets. *First Monday*, 12(6). <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1902/1784>.
- Greif, A. 2006. *Institutions and the Path to the Modern Economy: Lessons from Medieval Trade*. Cambridge University Press.
- Griffith, M., and Fox, S. 2007. Hobbyists Online. Data Memo, Pew Internet and American Life Project.
- Grimmer, L. 2006. Interview with David Heinemeier Hansson from Ruby on Rails. *My SQL Developer Zone*, February. <http://dev.mysql.com/tech-resources/interviews/david-heinemeier-hansson-rails.html>.
- Grossman, G. M., and Helpman, E. 1990. Comparative Advantage and Long-Run Growth. *American Economic Review* 80(4): 796–815.
- Gulati, R. 1998. Alliances and Networks. *Strategic Management Journal*, 19(4), 293–317.
- Gulati, R., Lawrence, P. R., and Puranam, P. 2005. Adaptation in Vertical Relationships: Beyond Incentive Conflict. *Strategic Management Journal* 26(5): 415–440.
- Gulati, R., and Singh, H. 1998. The Architecture of Cooperation: Managing Coordination Costs and Appropriation Concerns in Strategic Alliances. *Administrative Science Quarterly*, 43(4), 781–814.
- Hadar, A. 2008. Apple Probably Stealing the Idea, Rejects MailWrangler from the App Store. *MacBlogz*, September 21. <http://www.macblogz.com/2008/09/21/apple-rejects-mailwrangler-from-the-app-store-probably-stealing-the-idea/>.
- Hahn, R. W., and Wallsten, S. J. 2006. The Economics of Net Neutrality. *The Economist's Voice* 3(6): Article 8.
- Hain, T. 2000. Architectural Implications of NAT. Request for Comments 2993. IETF.
- Hall, P. 1994. *Innovation, Economics and Evolution: Theoretical Perspectives on Changing Technology in Economic Systems*. Harvester Wheatsheaf.

- Hall, P. A., and Soskice, D. 2001. An Introduction to Varieties of Capitalism. In P. Hall and D. Soskice, eds., *Varieties of Capitalism: The Institutional Foundations of Comparative Advantage*. Oxford University Press.
- Hall, P. A., and Taylor, R. C. R. 1996. Political Science and the Three New Institutionalisms. *Political Studies* 44(5): 936–957.
- Hall, R. E., and Lieberman, M. 2001. *Economics. Principles and Applications*, second edition. South-Western College Publishing.
- Hamm, S. 1998. The Education of Marc Andreessen. *Business Week*, April 13. <http://www.businessweek.com/1998/15/topstory.htm>.
- Hammock, R. 2006. The Next Small Thing. *mybusinessmag.com*, August/September. <http://www.mybusinessmag.com/fullstory.php3?sid=1413>.
- Handley, M. 2006. Why the Internet Only Just Works. *BT Technology Journal* 24(3): 119–129.
- Hansell, S. 2006. Plan for Fees on Some E-Mail Spurs Protest. *New York Times*, February 28.
- Hansell, S. 2007. Google Keeps Tweaking Its Search Engine. *New York Times*, June 3. <http://www.nytimes.com/2007/06/03/business/yourmoney/03google.html>.
- Hansell, S. 2008a. How Google Could Keep Yahoo From Microsoft. *New York Times*, February 4. <http://bits.blogs.nytimes.com/2008/02/04/how-google-could-keep-yahoo-from-microsoft/>.
- Hansell, S. 2008b. The Open Way Yahoo Could Do a Search Deal with Google. *New York Times*, April 15. <http://bits.blogs.nytimes.com/2008/04/09/the-open-way-yahoo-could-do-a-search-deal-with-google/>.
- Hardin, G. 1968. The Tragedy of the Commons. *Science* 162(3859): 1243–1248.
- Harhoff, D., Henkel, J., and von Hippel, E. 2003. Profiting from Voluntary Information Spillovers: How Users Benefit by Freely Revealing Their Innovations. *Research Policy*, 32(10), 1753–1769.
- Harris, R. G. 1998. The Internet as a GPT: Factor Market Implications. In E. Helpman, ed., *General Purpose Technologies and Economic Growth*. MIT Press.
- Hausman, J. A., Sidak, G. J., and Singer, H. J. 2001. Residential Demand for Broadband Telecommunications and Consumer Access to Unaffiliated Internet Content Providers. *Yale Journal on Regulation* 18(1): 129–173.
- Hawk, T. 2005. Flickr and WebShots: A Classic Web2.0 Case. *Thomas Hawk's Digital Connection*, November 25. <http://thomashawk.com/2005/11/flickr-and-webshots-classic-web20-case.html>.
- Hedlund, M. 2005. The Builders of Basecamp. *O'Reilly Network*, March 3. <http://www.oreillynet.com/pub/a/network/2005/03/10/basecamp.html>.

Heilemann, J. 2005. Journey to the (Revolutionary, Evil-Hating, Cash-Crazy, and Possibly Self-Destructive) Center of Google. *GQ*, March. <http://www.gq.com/news-politics/newsmakers/200502/google-larry-sergey>

Heinemeier Hansson, David. 2005. Entrepreneurs, Angels, and the Cost of Launch. *Signal vs. Noise*, June 30. http://www.37signals.com/svn/archives2/entrepreneurs_angels_and_the_cost_of_launch.php.

Heinemeier Hansson, David. 2008. It Doesn't Have to Be All or Nothing with a Startup. *Signal vs. Noise*, June 11. <http://www.37signals.com/svn/posts/1078-it-doesnt-have-to-be-all-or-nothing-with-a-startup>.

Helft, M. 2006. Yearning for Freedom ... From Venture Capital Overlords. *New York Times*, November 24.

Helft, M. 2008. The Humans behind the Google Money Machine. *New York Times*, June 2. <http://www.nytimes.com/2008/06/02/technology/02google.html?partner=rssnyt>.

Heller, M. A. 1998. The Tragedy of the Anticommons: Property in the Transition from Marx to Markets. *Harvard Law Review* 111(3): 621–688.

Hellmann, T., and Puri, M. 2000. The Interaction between Product Market and Financing Strategy: The Role of Venture Capital. *Review of Financial Studies* 13(4), 959–984.

Hellmann, T., and Puri, M. 2002. Venture Capital and the Professionalization of Start-Up Firms: Empirical Evidence. *Journal of Finance* 57(1): 169–197.

Hellweg, E. 2003. The Kazaa Conundrum: The RIAA's Lawsuits Have Split the Broadband ISP Industry and Will Likely Split Investors as Well. *CNN Money.com*, September 10. <http://money.cnn.com/2003/09/10/technology/techinvestor/hellweg/index.htm>.

Helpman, E. 2004. *The Mystery of Economic Growth*. Belknap.

Helpman, E., ed. 1998. *General Purpose Technologies and Economic Growth*. MIT Press.

Helpman, E., and Trajtenberg, M. 1998a. Diffusion of General Purpose Technologies. In E. Helpman, ed., *General Purpose Technologies and Economic Growth*. MIT Press.

Helpman, E., and Trajtenberg, M. 1998b. A Time to Sow and a Time to Reap. Growth Based on General Purpose Technologies. In E. Helpman, ed., *General Purpose Technologies and Economic Growth*. MIT Press.

Hemphill, C. S. 2008. Network Neutrality and the False Promise of Zero-Price Regulation. *Yale Journal on Regulation* 25(2): 135–180.

Henderson, R. M., and Clark, K. B. 1990. Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. *Administrative Science Quarterly* 35(1): 9–30.

- Henkel, J., and von Hippel, E. 2005. Welfare Implications of User Innovation. *Journal of Technology Transfer* 30(1–2): 73–87.
- Herbert, A.J. 1990. ANSA Approach to the Design of Distributed Systems. *IEEE Colloquium on Building Distributed Systems*.
- Herman, B. D. 2006. Opening Bottlenecks: On Behalf of Mandated Network Neutrality. *Federal Communications Law Journal* 59(1): 103–156.
- Herron, C. 2006. Caterina Fake and Meg Hourihan Share Flickr, Blogger Lore. *Christine(.net)*, August 16. http://www.christine.net/2006/08/caterina_fake_a.html.
- Hindman, M. 2008. *The Myth of Digital Democracy*. Princeton University Press.
- Hofmeister, C., Nord, R., and Soni, D. 2000. *Applied Software Architecture*. Addison-Wesley.
- Holahan, C. 2006. Why Yahoo's Panama Won't Be Enough. *BusinessWeek.com*, December 26. http://www.businessweek.com/print/technology/content/dec2006/tc20061226_633699.htm.
- Holahan, C. 2007. Panama's Promising Early Results. *BusinessWeek.com*, March 8. http://www.businessweek.com/print/technology/content/mar2007/tc20070307_187360.htm.
- Holbrook, D., Cohen, W. M., Hounshell, D. A., and Klepper, S. 2000. The Nature, Sources, and Consequences of Firm Differences in the Early History of the Semiconductor Industry. *Strategic Management Journal* 21(10–11): 1017–1041.
- Holdrege, M., and Srisuresh, P. 2001. Protocol Complications with the IP Network Address Translator. Request for Comments 3027. IETF.
- Holmstrom, B., and Milgrom, P. 1994. The Firm as an Incentive System. *American Economic Review* 84(4): 972–991.
- Honan, M. 2008. Photo Essay: Unlikely Places Where Wired Pioneers Had Their Eureka! Moments. *Wired*, March 24. http://www.wired.com/culture/lifestyle/multimedia/2008/03/ff_eureka.
- Hosseini, M., Ahmed, D. T., Shirmohammadi, S., and Georganas, N. D. 2007. A Survey of Application-Layer Multicast Protocols. *IEEE Communications Surveys and Tutorials* 9(3): 58–74.
- Hourihan, M. 2005. Meg Hourihan Interview. *Memory Lane*, May 17. <http://itc.conversationsnetwork.org/shows/detail541.html>.
- Huston, G. 2008. IPv4 Address Report, generated May 11. <http://ipv4.potaroo.net>.
- Hull, J. C. 2002. *Options, Futures, and Other Derivative Securities*, fifth edition. Prentice-Hall.

- Hutheesing, N. 2004. Corporate Inter-Face-Time. *Forbes*, March 22. <http://www.forbes.com/best/2004/0322/002.html>.
- IETF. 2008. Hypertext Transfer Protocol Bis (httpbis) Working Group Overview. August 28. <http://www.ietf.org/html.charters/httpbis-charter.html>.
- IETF. 2009. Official Internet Protocol Standards. September 17. <http://www.rfc-editor.org/rfcxx00.html>.
- Internet Freedom and Nondiscrimination Act of 2006. H.R. 5417. 109th Congress, 2nd Session. 2006.
- Internet Freedom Preservation Act. S. 215. 110th Congress, 1st Session. 2007.
- Internet Non-Discrimination Act of 2006. S. 2360. 109th Congress, 2nd Session. 2006.
- Isenberg, David. 1997. Rise of the Stupid Network: Why the Intelligent Network Was Once a Good Idea, but Isn't Anymore. One Telephone Company Nerd's Odd Perspective on the Changing Value Proposition. *Computer Telephony*, August: 16,18, 20, 24, 26.
- Isenberg, David S. 1998. The Dawn of the "Stupid Network." *netWorker* 2(1): 24–31.
- ISO. 1997. *International Standard ISO/IEC 9126: Software Quality Characteristics and Metrics. Part 1: Quality Characteristics and Sub-Characteristics. Part 2: External Metrics*. International Organization for Standardization. International Electrotechnical Commission.
- Jacobides, M. G. 2001. Revisiting Vertical Scope: Capabilities, Integration and Profitability in the Mortgage Banking Industry. Working Paper 01-16, Wharton Financial Institutions Center, University of Pennsylvania.
- Jacobides, M. G. 2002. Where Do Markets Come From? Working Paper CNE WP06/2002, Centre for the Network Economy.
- Jacobides, M. G. 2005. Industry Change through Vertical Disintegration: How and Why Markets Emerged in Mortgage Banking. *Academy of Management Journal* 48(3): 465–498.
- Jacobides, M. G., and Hitt, L. M. 2005. Losing Sight of the Forest for the Trees? Productive Capabilities and Gains from Trade as Drivers of Vertical Scope. *Strategic Management Journal* 26(13): 1209–1227.
- Jaffe, A. B. 2000. The U.S. Patent System in Transition: Policy Innovation and the Innovation Process. *Research Policy* 29(4–5): 531–557.
- Jaiswal, S., Iannaccone, G., Diot, C., Kurose, K., and Towsley, D. 2007. Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone. *IEEE/ACM Transactions on Networking* 15(1): 54–66.

- Jaspers, F., and van den Ende, J. 2006. The Organizational Form of Vertical Relationships: Dimensions of Integration. *Industrial Marketing Management* 35(7): 819–828.
- Jensen, M. C., and Meckling, W. H. 1976. Theory of the Firm. Managerial Behavior, Agency Costs and Ownership Structure. *Journal of Financial Economics* 3(4): 305–360.
- Jeppesen, L. B. 2005. User Toolkits for Innovation: Consumers Support Each Other. *Journal of Product Innovation Management* 22(4): 347–362.
- Jeppesen, L. B., and Molin, M. J. 2003. Consumers as Co-developers: Learning and Innovation Outside the Firm. *Technology Analysis and Strategic Management* 15(3): 363–383.
- Johnson, D. R., and Post, D. 1996. Law and Borders—The Rise of Law in Cyberspace. *Stanford Law Review* 48(5): 1367–1402.
- Jolls, C., Sunstein, C. R., and Thaler, R. H. 2000. A Behavioral Approach to Law and Economics. In C. Sunstein, ed., *Behavioral Law and Economics*. Cambridge University Press.
- Jones, C. I. 2002. Sources of U.S. Economic Growth in a World of Ideas. *American Economic Review* 92(1): 220–239.
- Jones, C. I., and Williams, J. C. 1998. Measuring the Social Return to R&D. *Quarterly Journal of Economics* 113(4): 1119–1135.
- Jones, K. C. 2007. Pearl Jam Blasts AT&T for Cut Lyrics in Lollapalooza Webcast. *InformationWeek*, Aug. 9. <http://www.informationweek.com/shared/printableArticle.jhtml?articleID=201310731>.
- Jorgenson, D. W. 2001. Information Technology and the US Economy. *American Economic Review* 91(1): 1–32.
- Jorgenson, D. W., Ho, M. S., and Stiroh, K. J. 2008. A Retrospective Look at the U.S. Productivity Growth Resurgence. *Journal of Economic Perspectives* 22(1): 3–24.
- Kaat, M. 2000. Overview of 1999 IAB Network Layer Workshop. Request for Comments 2956. IETF.
- Kahneman, D., and Tversky, A. 1984. Choices, Values and Frames. *American Psychologist* 39(4): 341–350.
- Kahneman, D., and Tversky, A., eds. 2000. *Choices, Values, and Frames*. Cambridge University Press.
- Kahney, L. 2001. Dot-Com Begs for Bucks. *Wired*, January 4. <http://www.wired.com/techbiz/media/news/2001/01/40979>.
- Kane, M. 2002a. Overture Bows to Google in AOL Deal. *CNET News*, May 27. <http://news.cnet.com/2100-1023-896152.html>.

- Kane, M. 2002b. Overture Bullish Despite Earthlink Loss. *CNET News*, May 27. http://news.cnet.com/Overture-bullish-despite-EarthLink-loss/2100-1023_3-831496.html.
- Kaplan, D. A. 1999. *The Silicon Boys and Their Valley of Dreams*. William Morrow.
- Kaplan, S., and Henderson, R. 2005. Inertia and Incentives: Bridging Organizational Economics and Organizational Theory. *Organization Science* 16(5): 509–521.
- Kaplan, S. B., and Strömberg, P. 2001. Venture Capitalists as Principals: Contracting, Screening, and Monitoring. *American Economic Review* 91(2): 426–430.
- Karagiannis, T., Rodriguez, P., and Papagiannaki, K. 2005. Should Internet Service Providers Fear Peer-Assisted Content Distribution? *Internet Measurement Conference 2005*.
- Karr, T. 2006. AOL Censors Opposition Site. *MediaCitizen*, April 14. <http://www.freepress.net/news/14960>.
- Katsh, M. E. 1996. Software Worlds and the First Amendment: Virtual Doorkeepers in Cyberspace. *University of Chicago Legal Forum*, 1996: 335–360.
- Katyal, N. K. 2002. Architecture as Crime Control. *Yale Law Journal* 111(5): 1039–1140.
- Katz, M. L. 2002. Intellectual Property Rights and Antitrust Policy: Four Principles for a Complex World. *Journal on Telecommunication and High Technology Law* 1: 325–353.
- Katz, M. L., and Shapiro, C. 1985. Network Externalities, Competition, and Compatibility. *American Economic Review* 75(3): 424–440.
- Katz, M. L., and Shapiro, C. 1994. Systems Competition and Network Effects. *Journal of Economic Perspectives* 8(2): 93–115.
- Katz, M. L., and Shapiro, C. 1999. Antitrust in Software Markets. In J. Eisenach and T. Lenard, eds., *Competition, Innovation, and the Microsoft Monopoly: Antitrust in the Digital Marketplace*. Kluwer.
- Kavassalis, P., and Lehr, W. 2000. The Flexible Specialization Path of the Internet. In E. Bohlin, K. Brodin, A. Lundgren, and B. Thorngren, eds., *Convergence in Communications and Beyond*. Elsevier.
- Kaye, K. 2008. Yahoo's Revamped AT&T Deal Aligns with Display Ad Network Goal. *The ClickZ News*, February 1. <http://www.clickz.com/showPage.html?page=3628297>.
- Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., and Carriere, J. 1998. The Architecture Tradeoff Analysis Method. *Fourth International Conference on Engineering Complex Computer Systems*.

- Kempf, J., and Austein, R. 2004. The Rise of the Middle and the Future of End-to-End: Reflections on the Evolution of the Internet Architecture. Request for Comments 3724. IETF.
- Kesan, J. P., and Shah, R. C. 2001. Fool Us Once Shame on You—Fool Us Twice Shame on Us: What We Can Learn from the Privatizations of the Internet Backbone Network and the Domain Name. *Washington University Law Quarterly* 79(1): 89–220.
- Kesan, J. P., and Shah, R. C. 2005. Shaping Code. *Harvard Journal of Law and Technology*, 18 (spring), 319–404.
- Kihlstrom, R. E., and Laffont, J.-J. 1979. A General Equilibrium Entrepreneurial Theory of Firm Formation Based on Risk Aversion. *Journal of Political Economy* 87(4): 719–748.
- King, B. 2002. The Day the Napster Died. *Wired*, May 15. <http://www.wired.com/gadgets/portablemusic/news/2002/05/52540?currentPage=all>.
- Kirzner, I. M. 1997. Entrepreneurial Discovery and the Competitive Market Process: An Austrian Approach. *Journal of Economic Literature* 35(1): 60–85.
- Kitch, E. W. 1977. The Nature and Function of the Patent System. *Journal of Law and Economics* 20(2): 265–290.
- Klein, D. S. 1999. Visionary in Obscurity. *SmartBusiness*, April. http://www.sbnonline.com/Local/Article/3592/82/0/Visionary_in_obscurity.aspx.
- Klein, M., and Kazman, R. 1999. *Attribute-Based Architectural Styles*. Technical Report CMU/SEI-99-TR-022, ESC-TR-49-022, Carnegie Mellon Software Engineering Institute.
- Klemperer, P. 1990. How Broad Should the Scope of Patent Protection Be? *Rand Journal of Economics* 21(1): 113–130.
- Knecht, G. Bruce. 1996. Reading the Market: How Wall Street Whiz Found a Niche Selling Books on the Internet. *Wall Street Journal*, May 16.
- Knight, F. H. 1971. *Risk, Uncertainty and Profit*. Reprint: University of Chicago Press.
- Kogut, B., and Zander, U. 1996. What Firms Do? Coordination, Identity, and Learning. *Organization Science* 7(5): 502–518.
- Kolasky, W. J. 1999. Network Effects: A Contrarian View. *George Mason Law Review* 7(3): 577–615.
- Kortum, S., and Lerner, J. 2000. Assessing the Contribution of Venture Capital to Innovation. *Rand Journal of Economics* 31(4): 674–692.
- Krantz, M. 2002. Web of Board Members Ties Together Corporate America. *USA Today*, November 25.

- Kraus, J. 2005a. It's a Great Time to be an Entrepreneur. *Bnoopy*, June 29. http://bnoopy.typepad.com/bnoopy/2005/06/its_a_great_tim.html.
- Kraus, J. 2005b. The Long Tail of Software. Millions of Markets of Dozens. *Bnoopy*, March 9. http://bnoopy.typepad.com/bnoopy/2005/03/the_long_tail_o.html.
- Kreimer, S. F. 2006. Censorship by Proxy: The First Amendment, Internet Intermediaries, and the Problem of the Weakest Link. *University of Pennsylvania Law Review* 155(1): 11–95.
- Krengel, U. 1991. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. 3rd ed. Vieweg.
- Kreps, D. M. 1990. *A Course in Microeconomic Theory*. Princeton University Press.
- Kruchten, P. B. 1995. The 4+1 View Model of Architecture. *IEEE Software* 12(6): 42–50.
- Kruse, H. 1999. Protocol Interactions and Their Effects on Internet-Based E-Commerce. Paper presented at Second International Conference on Telecommunications and Electronic Commerce.
- Kurose, J. F., and Ross, K. W. 2008. *Computer Networking: A Top-Down Approach*, fourth edition. Pearson/Addison-Wesley.
- Lakhani, K. R., and Wolf, R. G. 2005. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. In J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani, eds., *Perspectives on Free and Open Source Software*. MIT Press.
- Landes, W. M., and Posner, R. A. 2003. *The Economic Structure of Intellectual Property Law*. Belknap.
- Langlois, R. N. 2002. Modularity in Technology and Organization. *Journal of Economic Behavior and Organization* 49(1): 19–37.
- Langlois, R. N., and Robertson, P. L. 1992. Networks and Innovation in a Modular System: Lessons from the Microcomputer and Stereo Component Industries. *Research Policy* 21(4): 297–313.
- Latin American and Caribbean Internet Addresses Registry. 2007. LACNIC Announces the Imminent Depletion of the IPv4 Addresses. June 20 http://lacnic.net/en/anuncios/2007_agotamiento_ipv4.html.
- Laverty, K. J. 1996. Economic “Short-Termism”: The Debate, the Unresolved Issues, and the Implications for Management Practice and Research. *Academy of Management Review* 21(3): 825–860.
- Lea, D. 1994. Christopher Alexander: An Introduction for Object-Oriented Designers. *ACM SIGSOFT Software Engineering Notes* 19(1): 39–46.

Lee, R. S., and Wu, T. 2009. Subsidizing Creativity through Network Design: Zero-Pricing and Net Neutrality. *Journal of Economic Perspectives* 23(3): 61–76.

Leichtman Research Group. 2008. Over 8.5 Million Added Broadband from Top Cable and Telephone Companies in 2007. March 3. <http://www.leichtmanresearch.com/press/030308release.html>.

Leiponen, A., and Helfat, C. E. 2005. Innovation Objectives, Knowledge Sources, and the Benefits of Breadth. Working paper.

Lemley, M. A. 1997. The Economics of Improvement in Intellectual Property Law. *Texas Law Review* 75 (April): 989–1084.

Lemley, Mark A., and Lawrence Lessig. 1999. Ex Parte to Federal Communications Commission. CS Dkt. No. 99–251.

Lemley, M. A., and Lessig, L. 2000. Open Access to Cable Modems. *Whittier Law Review* 22(1): 3–34.

Lemley, M. A., and Lessig, L. 2001. The End of End-to-End: Preserving the Architecture of the Internet in the Broadband Era. *UCLA Law Review* 48(4): 925–972.

Lemley, M. A., and McGowan, D. 1998. Legal Implications of Network Economic Effects. *California Law Review* 86(3): 479–611.

Lenhart, A. 2009. Adults and Social Network Websites. Data Memo, Pew Internet and American Life Project.

Lerner, J. 2000. Small Business, Innovation, and Public Policy in the Information Technology Industry. In E. Brynjolfsson and B. Kahin, eds., *Understanding the Digital Economy: Data, Tools, and Research*. MIT Press.

Lerner, J. 2002. The Returns to Investments in Innovative Activities: An Overview and an Analysis of the Software Industry. In D. Evans, ed., *Microsoft, Antitrust and the New Economy: Selected Essays*, volume 2. Kluwer.

Lerner, J., and Tirole, J. 2002. Some Simple Economics of Open Source. *Journal of Industrial Economics* 50(2): 197–234.

Lerner, J., and Tirole, J. 2005a. The Economics of Technology Sharing: Open Source and Beyond. *Journal of Economic Perspectives* 19(2): 99–120.

Lerner, J., and Tirole, J. 2005b. The Scope of Open Source Licensing. *Journal of Law Economics and Organization* 21(1): 20–56.

Lessig, L. 1995. The Regulation of Social Meaning. *University of Chicago Law Review* 62(3): 943–1045.

Lessig, L. 1998. The New Chicago School. *Journal of Legal Studies* 27(2): 661–691.

Lessig, L. 1999a. *Code and Other Laws of Cyberspace*. Basic Books.

- Lessig, L. 1999b. The Law of the Horse: What Cyberlaw Might Teach. *Harvard Law Review* 113(2): 501–549.
- Lessig, L. 2001. *The Future of Ideas: The Fate of the Commons in a Connected World*. Vintage Books.
- Lessig, L. 2002. Testimony before the United States Senate, Committee on Commerce, Science, and Transportation, at Its Hearing on: The Government's Role in Promoting the Future of Telecommunications Industry and Broadband Deployment, October 1. 107th Cong., 2nd Session.
- Lessig, L. 2004. *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*. Penguin.
- Lessig, L. 2006. Testimony before the United States Senate, Committee on Commerce, Science, and Transportation, at Its Hearing on: Network Neutrality, February 2. 109th Cong., 2nd Session.
- Lessig, L. 2008a. *Remix: Making Art and Commerce Thrive in the Hybrid Economy*. Penguin.
- Lessig, L. 2008b. Testimony before the United States Senate, Committee on Commerce, Science, and Transportation, at its Hearing on: The Future of the Internet, April 22. 110th Cong., 2nd Session.
- Levenson, M. 2008. Facebook, ConnectU Settle Dispute. *Boston Globe*, June 27. http://www.boston.com/business/technology/articles/2008/06/27/facebook_connectu_settle_dispute/.
- Lichtman, D. G. 2000. Property Rights in Emerging Platform Technologies. *Journal of Legal Studies* 29(2): 615–648.
- Liebowitz, S. J., and Margolis, S. E. 2001. *Winners, Losers and Microsoft: Competition and Antitrust in High Technology*, revised edition. Independent Institute.
- Lindgren, B. W. 1962. *Statistical Theory*. Macmillan.
- Lipsey, R. G., Carlaw, K. I., and Bekar, C. T. 2005. *Economic Transformations: General Purpose Technologies and Long-Term Economic Growth*. Oxford University Press.
- Liptak, A. 2007a. Verizon Rejects Text Messages of Abortion Rights Group. *New York Times*, September 27.
- Liptak, A. 2007b. Verizon Reverses Itself on Abortion Messages. *New York Times*, September 28. <http://www.nytimes.com/2007/09/28/business/28verizon.html?th&emc=th>.
- Litan, R. E., and Rivlin, A. M. 2001. *Beyond the Dot.coms: The Economic Promise of the Internet*. Brookings Institution.

- Livingston, J. 2007. *Founders at Work: Stories of Startups' Early Days*. University of California Press.
- Loewenstein, G., and Prelec, D. 1992. Anomalies in Intertemporal Choice: Evidence and an Interpretation. *Quarterly Journal of Economics* 107(2): 573–597.
- Lohr, S. 2008. Yahoo Offer Is Strategy Shift for Microsoft. *New York Times*, February 2. <http://www.nytimes.com/2008/02/02/technology/02soft.html>.
- Lucky, R. W. 1997. When Is Dumb Smart? *IEEE Spectrum* 34(11): 21.
- Luenberger, D. G. 1997. *Investment Science*. Oxford University Press.
- Luethje, C., Herstatt, C., and von Hippel, E. 2005. User-Innovators and “Local” Information: The Case of Mountain Biking. *Research Policy* 34(6): 951–965.
- MacCormack, A., Rusnak, J., and Baldwin, C. 2007. The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry. Working Paper 08-038, Harvard Business School.
- MacCormack, A., Rusnak, J., and Baldwin, C. Y. 2008. Exploring the Duality between Product and Organizational Architectures: A Test of the Mirroring Hypothesis. Working Paper 08-039, Harvard Business School.
- MacKie-Mason, J. 2000. *An AOL/Time Warner Merger Will Harm Competition in Internet Online Services*. Report submitted to the U.S. Federal Trade Commission.
- MacKie-Mason, J., and Metzler, J. 1999. Links between Vertically Related Markets: Kodak. In J. Kwoka Jr. and L. White, eds., *The Antitrust Revolution: Economics, Competition, and Policy*, third edition. Oxford University Press.
- MacKie-Mason, J., Shenker, S., and Varian, H. 1995. Network Architecture and Content Provision: An Economic Analysis (June 29, 1996). 23rd Telecommunications Policy Research Conference.
- MacKie-Mason, J. K., Shenker, S., and Varian, H. R. 1996. Service Architecture and Content Provision. The Network Provider as Editor. *Telecommunications Policy* 20(3): 203–217.
- Madison River Communications. 2006. Who We Are. Madison River Communications. http://www.madisonriver.net/about_us/who_we_are.php
- Maier, A. 2008. Exklusiv: Gebührenpflicht auf der Datenautobahn. Briten bieten Google die Stirn. *Financial Times Deutschland*, July 16. http://www.ftd.de/technik/it_telekommunikation/:Geb%FChrenpflicht%20Datenautobahn%20Briten%20Google%20Stirn/386651.html.
- Mandy, D. M. 2000. Killing the Goose That May Have Laid the Golden Egg: Only the Data Know Whether Sabotage Pays. *Journal of Regulatory Economics* 17(2): 157–172.

- Maney, K. 2006. Disrupter Man Goes after TV This Time. *USA Today*, December 6.
- Manjoo, F. 2008. The Cell Phone Wars. *Slate*, September 25. <http://www.slate.com/id/2200914/>.
- Mansfield, E., Rapoport, J., Romeo, A., Wagner, S., and Beardsley, G. 1977. Social and Private Rates of Return from Industrial Innovations. *Quarterly Journal of Economics* 91(2): 221–240.
- Marginson, D., and McAulay, L. 2007. Exploring the Debate on Short-Termism: A Theoretical and Empirical Analysis. *Strategic Management Journal* 29(3): 273–292.
- Markillie, P. 2005. Crowned At Last. *The Economist*, April 2, 3–6.
- Markoff, J. 2007. Who Founded Facebook? A New Claim Emerges. *New York Times*, September 1. <http://www.nytimes.com/2007/09/01/technology/01facebook.html>.
- Masten, S. E. 1999. About Oliver E. Williamson. In G. R. Carroll and D. J. Teece, eds., *Firms, Markets, and Hierarchies: The Transaction Cost Economics Perspective*. Oxford University Press.
- Matutes, C., Regibeau, P., and Rockett, K. 1996. Optimal Patent Design and the Diffusion of Innovations. *Rand Journal of Economics* 27(1): 60–83.
- McCord, K. R., and Eppinger, S. D. 1993. Managing the Integration Problem in Concurrent Engineering. Working Paper 3594-93-MSA, MIT Sloan School of Management.
- McCullagh, D. 2003. Skype's VoIP Ambitions. *CNET News*, December 2. <http://www.news.com/2008-7352-5112783.html>.
- McCullagh, D. 2005. Telco Agrees to Stop Blocking VoIP Calls. *CNET News*, March 3. http://news.cnet.com/Telco-agrees-to-stop-blocking-VoIP-calls/2100-7352_3-5598633.html.
- McKinnon, M. 2001. King of the Blogs. *Shift*, summer. <http://www.matthewmckinnon.com/kingoftheblogs.html>.
- Mead, R. 2000. You've Got Blog. *The New Yorker*, November 13: 102.
- Mennecke, T. 2005. DSL Broadband Providers Perform Balancing Act. *Slyck News*, November 1. <http://www.slyck.com/news.php?story=973>.
- Mercurio, J. 2002. Lott Apologizes for Thrumond Comment. *CNN*, December 10. <http://archives.cnn.com/2002/ALLPOLITICS/12/09/lott.comment/>.
- Mercurio, N., and Medema, S. G. 2006. *Economics and the Law: From Posner to Post-modernism and Beyond*, second edition. Princeton University Press.
- Merges, R. 1994. Intellectual Property Rights and Bargaining Breakdown: The Case of Blocking Patents. *Tennessee Law Review* 62(1): 75–106.

- Merges, R. P., and Nelson, R. R. 1990. On the Complex Economics of Patent Scope. *Columbia Law Review* 90(3): 839–916.
- Merges, R. P., and Nelson, R. R. 1994. On Limiting or Encouraging Rivalry in Technical Progress: The Effect of Patent Scope Decisions. *Journal of Economic Behavior and Organization* 25(1): 1–24.
- Merton, R. C. 1973a. An Intertemporal Capital Asset Pricing Model. *Econometrica* 41(5): 867–887.
- Merton, R. C. 1973b. Theory of Rational Option Pricing. *Bell Journal of Economics* 4(1): 141–183.
- Merton, R. C. 1992. *Continuous-Time Finance*. Blackwell.
- Mervana, S., and Le, C. 2001. *Design and Implementation of DSL-Based Access Solutions*. Cisco Press.
- Messerschmitt, D. G. 2000. *Understanding Networked Applications: A First Course*. Morgan Kaufmann.
- Microsoft Online Services. 2009. MSN Dial Up: Overview. <http://get.msn.com/?prod=dialup>.
- Milgrom, P., and Roberts, J. 1992. *Economics, Organization and Management*. Prentice-Hall.
- Miller, C. C. 2008a. I.P.O. Crisis Could Have Lingering Effect on Start-Ups. *New York Times*, October 3. <http://bits.blogs.nytimes.com/2008/10/03/ipo-crisis-could-have-lingering-effect-on-start-ups/>.
- Miller, C. C. 2008b. Venture Capital Exit Drought Continues. *New York Times*, October 1. <http://bits.blogs.nytimes.com/2008/10/01/venture-capital-exit-drought-continues/>.
- Miller, D. A. 2007. Invention under Uncertainty and the Threat of Ex Post Entry. *European Economic Review* 52 (April): 387–412.
- Miller, D. K. 2005. The Continuing Telus Website Blocking Saga. *penmachine.com*, July 27. <http://www.penmachine.com/2005/07/continuing-telus-website-blocking-saga.html>.
- Minar, N., and Hedlund, M. 2001. A Network of Peers: Peer-to-Peer Models through the History of the Internet. In *Peer-to-Peer: Harnessing the Benefits of Disruptive Technologies*, ed. A. Oram. O'Reilly.
- Miniwatts Marketing Group. 2009. Internet Usage Statistics. The Internet Big Picture. World Internet Users and Population Stats as of June 30, 2009. <http://www.internetworldstats.com/stats.htm>.

- MIT Communications Futures Program and Cambridge University Communications Research Network Broadband Working Group. 2005. *The Broadband Incentive Problem*. White Paper.
- Mitchell, W. J. 1995. *City of Bits: Space, Place, and the Infobahn*. MIT Press.
- Mockus, A., Fielding, R. T., and Herbsleb, J. D. 2002. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* 11(3): 309–346.
- Mogul, J. C. 2004. Clarifying the Fundamentals of HTTP. *Software, Practice and Experience* 34(2): 103–134.
- Mohney, D. undated. Niklas Zennstrom, Founder of Kazaa, Skype and Joost Interview. *vonmag.com*. <http://vonmag.com/editorial/pioneer/niklas-zennstrom-founder-of-kazaa-skype-and-joost-interview>.
- Monteverde, K. 1995. Technical Dialog as an Incentive for Vertical Integration in the Semiconductor Industry. *Management Science* 41(10): 1624–1638.
- Moody, G. 2001. *Rebel Code: Linux and the Open Source Revolution*. Penguin.
- Moors, T. 2002. A Critical Review of ‘End-to-End Arguments in System Design.’ *IEEE International Conference on Communications*.
- Morelli, M. D., Eppinger, S. D., and Gulati, R. K. 1995. Predicting Technical Communication in Product Development Organizations. *IEEE Transactions on Engineering Management* 42(3): 215–222.
- Morris, C. R., and Ferguson, C. R. 1993. How Architecture Wins Technology Wars. *Harvard Business Review* 71(2): 86–96.
- Mortenson, G. M. 2007. Petition for Rulemaking to Federal Communications Commission, July 20.
- Moukheiber, Z. 1996. Kleiner’s Web. *Forbes* 157(6): 40–42.
- Murmann, J. P. 2003. *Knowledge and Competitive Advantage: The Coevolution of Firms, Technology, and National Institutions*. Cambridge University Press.
- Mutz, D. C. 2006. *Hearing the Other Side: Deliberative Versus Participatory Democracy*. Cambridge University Press.
- Myers, S. 1977. Determinants of Corporate Borrowing. *Journal of Financial Economics* 5(2): 147–175.
- National Cable and Telecommunications Association. 2007. Comments to Federal Communications Commission. WC Dkt. No. 07–52. June 15.
- National Venture Capital Association. 2008. *Yearbook 2008*.
- Naughton, J. 2008. How Flickr Developed into a Classic Web 2.0 Success. *The Observer*, March 9: 16.

- Nelson, R. R. 1991. Why Do Firms Differ, and How Does It Matter? *Strategic Management Journal*, 12 (special issue: Fundamental Research Issues in Strategy and Economics): 61–74.
- Nelson, R. R. 2005. *Technology, Institutions and Economic Growth*. Harvard University Press.
- Nelson, R. R., and Winter, S. G. 1977. In Search of Useful Theory of Innovation. *Research Policy* 6(1): 37–76.
- Nelson, R. R., and Winter, S. G. 1982. *An Evolutionary Theory of Economic Change*. Belknap.
- Net Applications. 2008. Browser Market Share. November. <http://marketshare.hitslink.com/firefox-market-share.aspx?qprid=0&sample=28>.
- Netcraft. 2008. *July 2008 Web Server Survey*, July 7. http://news.netcraft.com/archives/2008/07/07/july_2008_web_server_survey.html
- Netscape. 1995a. Intuit to Provide Internet Access Directly From Quicken. Press release, October 19. <http://cgi.netscape.com/newsref/pr/newsrelease55.html>.
- Netscape. 1995b. Macromedia and Netscape Combine Technologies to Bring True Multimedia to the Internet. Press release, June 5. <http://cgi.netscape.com/newsref/pr/newsrelease28.html>.
- Netscape. 1995c. Netscape and Sun. Press release, April 11. <http://cgi.netscape.com/newsref/pr/newsrelease23.html>.
- Netscape. 1995d. Netscape to License Sun's Java Programming Language. Press release, May 23. <http://cgi.netscape.com/newsref/pr/newsrelease25.html>.
- Network Neutrality Act of 2006. H.R. 5273. 109th Congress 2nd Session. 2006.
- Niewijk, R., Songhurst, C., and Todd, P. 2003. Why European ISPs Need Partners. *McKinsey Quarterly*, February. http://www.mckinseyquarterly.com/Why_European_ISPs_need_partners_1280
- North, D. C. 1990. *Institutions, Institutional Change and Economic Performance*. Cambridge University Press.
- North, D. C. 2005. Institutions and the Performance of Economies over Time. In C. Menard and M. M. Shirley, eds., *Handbook of New Institutional Economics*. Springer.
- Novak, S., and Stern, S. 2003. The Impact of Technological Interdependency on Contracting Complementarities: Evidence from Automobile Product Development. Working paper, Northwestern University.
- Nuechterlein, J. E., and Weiser, P. J. 2005. *Digital Crossroads: American Telecommunications Policy in the Internet Age*. MIT Press.

- O'Brien, L. 2007. Poking Facebook. *02138*, November/December. <http://www.02138mag.com/magazine/article/1724.html>.
- O'Connell, P. 2005. Online Extra: At SBC, It's All about Scale and Scope. *Business Week*, November 7. http://www.businessweek.com/@@n34h*IUQu7KtOwgA/magazine/content/05_45/b3958092.htm.
- Oates, J. 2006. Germans Want to Charge Google. *The Register*, February 27. http://www.theregister.co.uk/2006/02/27/deutche_telekom_google/.
- Odlyzko, Andrew. 1998. Smart and Stupid Networks: Why the Internet Is Like Microsoft. *netWorker* 2(5): 38–46.
- Oliner, S. D., and Sichel, D. E. 2000. The Resurgence of Growth in the Late 1990s: Is Information Technology the Story? *Journal of Economic Perspectives* 14(4): 3–22.
- Olsen, S. 2002a. Google Challenges Pay-For-Play Search. *CNET News*, May 27. <http://news.cnet.com/2100-1023-830123.html>.
- Olsen, S. 2002b. MSN, Overture Test Paid Listings. *CNET News*, January 7. http://news.cnet.com/MSN,-Overture-test-paid-listings/2110-1023_3-801374.html.
- Olsen, S. 2002c. Overture Sues Google over Search Patent. *CNET News*, May 27. <http://news.cnet.com/2100-1023-876861.html>.
- Olsen, S. 2002d. Yahoo Plays Overture's Paid Search Listings. *CNET News*, May 27. <http://news.cnet.com/2100-1023-275821.html>.
- Olsen, S. 2004. Google, Yahoo Bury the Legal Hatchet. *CNET News*, August 9. http://news.cnet.com/Google,-Yahoo-bury-the-legal-hatchet/2100-1024_3-5302421.html.
- Olsen, S. 2006. AOL Charged with Blocking Opponents' E-mail. *ZDNet News*, April 13. http://news.zdnet.com/2100-9595_22-6061089.html.
- OpenNet Initiative. 2005. OpenNet Initiative: Bulletin 010. Telus Blocks Consumer Access to Labour Union Web Site and Filters an Additional 766 Unrelated Sites. August 2. <http://opennet.net/bulletins/010/>.
- Ordovery, J. A. Sykes, A. O., and Willig, R. D. 1985. Nonprice Anticompetitive Behavior by Dominant Firms toward the Producers of Complementary Products. In *Anti-trust and Regulation: Essays in Memory of John J. McGowan*, ed. F. M. Fisher. MIT Press.
- Osterlie, T. 2002. The User-Developer Convergence: Innovation and Software Systems Development in the Apache Project. Master's thesis, Norwegian University of Science and Technology.
- Ostrom, E. 2005. Doing Institutional Analysis: Digging Deeper than Markets and Hierarchies. In C. Menard and M. M. Shirley, eds., *Handbook of New Institutional Economics*. Springer.

Page, L., Brin, S., Motwani, R., and Winograd, T.. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical report, Stanford InfoLab.

Park, A. 2008. The Brash Boys at 37signals Will Tell You: Keep It Simple, Stupid. *Wired*, February 25. http://www.wired.com/techbiz/media/magazine/16-03/mf_signals.

Parnas, D. L. 1972. On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM* 15(12): 1053–1058.

Partridge, C. 2008a. E-Mail to Barbara van Schewick. August 2.

Partridge, C. 2008b. The Technical Development of Internet Email. *IEEE Annals of the History of Computing* 30(2): 3–29.

Paxson, V. 1999. End-to-End Internet Packet Dynamics. *IEEE/ACM Transactions on Networking*, 7(3), 277–292.

Peha, J. M. 2007. The Benefits and Risks of Mandating Network Neutrality, and the Quest for a Balanced Policy. *International Journal of Communication* 1: 644–668.

Pelline, J. 1998. Pay-for-Placement Gets Another Shot. *CNET News*, January 2. http://news.cnet.com/Pay-for-placement-gets-another-shot/2100-1023_3-208309.html.

Penmachine. 2005a. The Continuing Telus Website Blocking Saga. July 27. <http://www.penmachine.com/2005/07/continuing-telus-website-blocking-saga.html>.

Penmachine. 2005b. Telus-Voices for Change Website Settlement Agreement. July 29. <http://www.flickr.com/photos/penmachine/29590389/>.

Perez, C. 2002. *Technological Revolutions and Financial Capital: The Dynamics of Bubbles and Golden Ages*. Elgar.

Peter, I. no date. On the Design of TCP/IP. *NetHistory: Internet History Project*. <http://www.nethistory.info/Archives/tcpiptalk.htmls>.

Peteraf, M. A. 1993. The Cornerstones of Competitive Advantage: A Resource-Based View. *Strategic Management Journal* 14(3), 179–191.

Peters, B. G. 2005. *Institutional Theory in Political Science: The New Institutionalism*, second edition. Continuum.

Peterson, L. L., and Davie, B. S. 2003. *Computer Networks: A Systems Approach*, third edition. Morgan Kaufmann.

Peterson, L. L., and Davie, B. S. 2007. *Computer Networks: A Systems Approach*, fourth edition. Morgan Kaufmann.

PHP Group. 2008. History of PHP and Related Projects. September 11. <http://us2.php.net/history>.

- Poppo, L., and Zenger, T. 1998. Testing Alternative Theories of the Firm: Transaction Cost, Knowledge-Based, and Measurement Explanations for Make-or-Buy Decisions in Information Services. *Strategic Management Journal* 19(9): 853–877.
- Porter, K., Bunker Whittington, K., and Powell, W. W. 2005. The Institutional Embeddedness of High-Tech Regions: Relational Foundations of the Boston Biotechnology Community. In S. Breschi and F. Malerba, eds., *Clusters, Networks, and Innovation*. Oxford University Press.
- Posner, E. A. 1996a. Law, Economics and Inefficient Norms. *University of Pennsylvania Law Review* 144(5): 1697–1744.
- Posner, E. A. 1996b. The Regulation of Groups: The Influence of Legal and Nonlegal Sanctions on Collective Action. *University of Chicago Law Review* 63(1): 133–197.
- Posner, E. A. 1998a. Symbols, Signals, and Social Norms in Politics and the Law. *Journal of Legal Studies* 27(2, Part 1): 765–798.
- Posner, E. A. 2000. *Law and Social Norms*. Harvard University Press.
- Posner, R. A. 1997. Social Norms and the Law: An Economic Approach. *American Economic Review* 87(2): 365–369.
- Posner, R. A. 1998b. *Economic Analysis of Law*, fifth edition. Aspen Law and Business.
- Posner, R. A. 2001. *Antitrust Law*, second edition. University of Chicago Press.
- Postel, J. 1977a. Comments on Internet Protocol and TCP. Internet Engineering Note 2.
- Postel, J. 1977b. TCP Meeting Notes, 13 and 14 October 1977. Internet Engineering Note 66.
- Postel, J. 1978a. Internet Meeting Notes, 30 and 31 October 1978. Internet Engineering Note 63.
- Postel, J. 1978b. Meeting Notes, 1 February 1978. Internet Engineering Note 22.
- Postel, J. 1979a. Internet Datagram Protocol Version 4. Internet Engineering Note 80.
- Postel, J. 1979b. Transmission Control Protocol—TCP Version 4. Internet Engineering Note 81.
- Postel, J. 1981a. Internet Protocol: DARPA Internet Program Protocol Specification. Request for Comments 791. IETF.
- Postel, J. 1981b. Transmission Control Protocol: DARPA Internet Program Protocol Specification. Request for Comments 793. IETF.
- Pouzin, L., and Zimmerman, H. 1978. A Tutorial on Protocols. *Proceedings of the IEEE* 66(11): 1346–1370.

Powell, W.W., Koput, K. W., Bowie, J. I., and Smith-Doerr, L. 2002. The Spatial Clustering of Science and Capital: Accounting for Biotech Firm-Venture Capital Relationships. *Regional Studies* 36(3): 299–313.

Powell, W. W., and DiMaggio, P. J., eds. 1991. *The New Institutionalism in Organizational Analysis*. University of Chicago Press.

Powell, W. W., and Grodal, S. 2005. Networks of Innovators. In J. Faberberg, D. C. Mowery, and R. R. Nelson, eds., *The Oxford Handbook of Innovation*. Oxford University Press.

Pressman, R. S. 1997. *Software Engineering: A Practitioner's Approach*, fourth edition. McGraw-Hill.

PriMetrica. 2008. VoIP Ban Remains for Now. March 11. http://www.telegeography.com/cu/article.php?article_id=22126&email=html.

Public Knowledge, Free Press, Consumer Federation of America, Consumers Union, Educause, Media Access Project, et al. 2007. Petition for Declaratory Ruling. WT Dkt. No. 08–7. December 11.

Quistagaard, K. 1997a. Amazon.com Upstages Barnes & Noble. *Wired*, March 17. <http://www.wired.com/techbiz/media/news/1997/03/2609>.

Quistagaard, K. 1997b. Online Book Wars Escalate Further. *Wired*, May 16. <http://www.wired.com/techbiz/media/news/1997/05/3925>.

Ralph, E. K. 2006. Letter: Comment on the Economics of Network Neutrality. *The Economists' Voice*. <http://www.bepress.com/ev/vol3/iss1/art6>.

Ramakrishnan, K., Floyd, S., and Black, D. 2001. The Addition of Explicit Congestion Notification (ECN) to IP. Request for Comments 3168. IETF.

Ranaweera, C., and Prabhu, J. 2003. The Influence of Satisfaction, Trust and Switching Barriers on Customer Retention in a Continuous Purchasing Setting. *International Journal of Service Industry Management* 14(4): 374–395.

Rangar, B. S. 2005. Skype: Reflections for European Venture Capital. *Ariadne Capital Journal—Through the Maze*. http://www.ariadnecapital.com/journal/v5e3/outlook_alwayson.htm.

Rao, P. M. 1999. Convergence and Unbundling of Corporate R&D in Telecommunications: Is Software Taking the Helm? *Telecommunications Policy* 23(1): 83–93.

Rapp, D. 2006. Inventing Yahoo! *AmericanHeritage.com*, April 12. w2.eff.org/patent/wanted/neomedia/reexam/ExhibitH_InventingYahoo!.pdf.

Rashtchy, S., Kessler, A. M., Bieber, P. J., Schindler, N. H., and Tzeng, J. C. 2007. *The User Revolution: The New Advertising Ecosystem and the Rise of the Internet as a Mass Medium*. PiperJaffray Investment Research.

- Ratnasamy, S., Shenker, S., and McCanne, S. 2005. Towards an Evolvable Internet Architecture. *2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*.
- Raymond, E. S. 1999. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly.
- Rayport, J. F., and Louie, D. L. 1998. Amazon.com (A). Case 9-897-128, Harvard Business School.
- Rayport, J. F., and Louie, D. L. 2001. barnesandnoble.com (A). Case 9-898-082, Harvard Business School.
- Read, D. 2004. Intertemporal Choice. In N. Harvey and D. J. Koehler, eds., *Blackwell Handbook of Judgment and Decision Making*. Blackwell.
- Reardon, M. 2007. Comcast Denies Monkeying with BitTorrent Traffic. *CNET News*, August 21. http://www.news.com/8301-10784_3-9763901-7.html.
- Rebello, K., Cortese, A., and Hof, R. D. 1996a. Inside Microsoft (Part 1). *BusinessWeek*, July 15. <http://www.businessweek.com/1996/29/b34841.htm>.
- Rebello, K., Cortese, A., and Hof, R. D. 1996b. Inside Microsoft (Part 2). *BusinessWeek*, July 15. <http://www.businessweek.com/1996/29/b34842.htm>
- Reed, D. P. 2000. The End of the End-to-End Argument. <http://www.reed.com/Papers/endofendtoend.html>
- Reed, D. P. 2001. [e2e] Where End-to-End Ends. E-Mail to end2end-interest@postel.org Mailing List. May 23, 12:35:41 PDT. <http://www.postel.org/pipermail/end2end-interest/2001-May/000908.html>.
- Reed, D. P. 2002. [e2e] Clarifying the End-to-End Principle. E-Mail to end2end-interest@postel.org Mailing List. March 5, 06:10:17 PST. <http://www.postel.org/pipermail/end2end-interest/2002-March/001853.html>.
- Reed, D. P. 2006a. "[ih] Date of RFC 791 for Celebration. E-Mail to internet-history@postel.org Mailing List. March 30, 05:55:09 PST. <http://mailman.postel.org/pipermail/internet-history/2006-March/000555.html>.
- Reed, D. P. 2006b. [ih] UDP Creation (Was: Date of RFC 791 for Celebration). E-Mail to internet-history@postel.org Mailing List. March 31, 08:46:59 PST. <http://mailman.postel.org/pipermail/internet-history/2006-March/000569.html>.
- Reed, D. P. 2009a. E-Mail to Barbara van Schewick. October 16.
- Reed, D. P. 2009b. Re: [IP] ISPs helping with botnets. E-Mail to ip@v2.listbox.com Mailing List. October 13. <http://seclists.org/interesting-people/2009/Oct/102>.
- Reed, D. P., and Postel, J. 1978. User Datagram Protocol. Internet Engineering Note 71.

- Reed, D. P., Saltzer, J. H., and Clark, D. D. 1998. Commentaries on 'Active Networking and End-to-End Arguments'. *IEEE Network* 12(3): 69–71.
- Reidenberg, J. R. 1998. Lex Informatica: The Formulation of Information Policy Rules through Technology. *Texas Law Review* 76(3): 533–594.
- Reinganum, J. F. 1989. The Timing of Innovation: Research, Development, and Diffusion. In R. L. Schmalensee and R. D. Willig, eds., *Handbook of Industrial Organization*, volume 1. Elsevier.
- Reingold, J. 2008. The Secret Coach. *Fortune* 158(2): 124–134.
- Reiss, S. 2008. Cloud Computing. Available at Amazon.com Today. *Wired*, April 21. http://www.wired.com/techbiz/it/magazine/16-05/mf_amazon.
- Rindfleisch, A., and Heide, J. B. 1997. Transaction Cost Analysis: Past, Present, and Future Applications. *Journal of Marketing* 61(4): 30–54.
- Rivlin, G. 2006. Wallflower at the Web Party. *New York Times*, October 15.
- Roberts, L. G. 1967. Multiple Computer Networks and Intercomputer Communication. *First ACM Symposium on Operating System Principles*.
- Rochet, J.-C., and Tirole, J. 2003. Platform Competition in Two-Sided Markets. *Journal of the European Economic Association* 1(4): 990–1029.
- Rochet, J.-C., and Tirole, J. 2006. Two-sided Markets: A Progress Report. *Rand Journal of Economics* 37(3): 645–667.
- Roe, M. J. 1994. *Strong Managers, Weak Owners: The Political Roots of American Corporate Finance*. Princeton University Press.
- Romer, Paul M. 1990. Endogenous Technological Change. *Journal of Political Economy* 98(5): S71–S102.
- Roscoe, T. 2006. The End of Internet Architecture. *Fifth Workshop on Hot Topics in Networks*.
- Rosenberg, J. 2008. UDP and TCP as the New Waist of the Internet Hourglass. Internet-Draft, February 11.
- Rosenberg, N. 1982. *Inside the Black Box: Technology and Economics*. Cambridge University Press.
- Rosenberg, N. 1994. *Exploring the Black Box: Technology, Economics and History*. Cambridge University Press.
- Rosenberg, N. 1996. Uncertainty and Technological Change. In R. Landau, T. Taylor, and G. Wright, eds., *The Mosaic of Economic Growth*. Stanford University Press.
- Roth, D. 2004. Catch Us If You Can. *Fortune* 49(3): 64–74.

- Roth, D. 2008. Google's Open Source Android OS Will Free the Wireless Web. *Wired*, June 23. http://www.wired.com/techbiz/media/magazine/16-07/ff_android.
- Rubinfeld, D. L., and Singer, H. J. 2001. Vertical Foreclosure in Broadband Access? *Journal of Industrial Economics* 49(3): 299–318.
- Ruby on Rails. 2008. The Core Team. <http://rubyonrails.org/core>.
- Rysman, M. 2004. Competition between Networks: A Study of the Market for Yellow Pages. *Review of Economic Studies* 71(247): 483–512.
- Sako, M. 2002. Modularity and Outsourcing: The Nature of Co-Evolution of Product Architecture and Organisation Architecture in the Global Automotive Industry. Working paper, Said Business School, University of Oxford.
- Sako, M. 2004. Supplier Development at Honda, Nissan and Toyota: Comparative Case Studies of Organizational Capability Enhancement. *Industrial and Corporate Change* 13(2): 281–308.
- Salop, S. C., and Craig Romaine, R. 1999. Preserving Monopoly: Economic Analysis, Legal Standards, and Microsoft. *George Mason Law Review* 7(3): 617–671.
- Saltzer, J. H. 1999. "Open Access" Is Just the Tip of the Iceberg. October 22. <http://web.mit.edu/Saltzer/www/publications/openaccess.html>.
- Saltzer, J. H., Reed, D. P., and Clark, D. D. 1981. End-to-End Arguments in System Design. *2nd International Conference on Distributed Computing Systems*.
- Saltzer, J. H., Reed, D. P., and Clark, D. D. 1984. End-to-End Arguments in System Design. *ACM Transactions on Computer Systems* 2(4): 277–288.
- Samuelson, P., and Scotchmer, S. 2002. The Law and Economics of Reverse Engineering. *Yale Law Journal* 111(7): 1575–1664.
- Samuelson, W., and Zeckhauser, R. 1988. Status Quo Bias in Decision Making. *Journal of Risk and Uncertainty* 1(1): 7–59.
- Sanchez, R. 1995. Strategic Flexibility in Product Competition. *Strategic Management Journal* 16 (special issue, summer): 135–159.
- Sanchez, R. 1999. Modular Architectures in the Marketing Process. *Journal of Marketing* 63: 92–111.
- Sanchez, Ron and Joseph T. Mahoney. 1996. Modularity, Flexibility, and Knowledge Management in Product and Organization Design. *Strategic Management Journal* 17 (special issue, winter): 63–76.
- Sanchez, R. A. 1991. Strategic Flexibility, Real Options and Product-Based Strategy. PhD dissertation, Massachusetts Institute of Technology.

Sandvig, C. 2002. Communication Infrastructure and Innovation: The Internet as End-to-End Network That Isn't. Paper presented at Symposium with the Next Generation of Leaders in Science and Technology Policy.

Sandvig, C. 2006. Shaping Infrastructure and Innovation on the Internet: The End-to-End Network That Isn't. In D. H. Guston and D. Sarewitz, eds., *Shaping Science and Technology Policy: The Next Generation of Research*. University of Wisconsin Press.

Sandvine. 2004. *Meeting the Challenge of Today's Evasive P2P Traffic: Service Provider Strategies for Managing P2P Filesharing*. Sandvine Inc.

Sarac, K., and Almeroth, K. C. 2005. Monitoring IP Multicast in the Internet: Recent Advances and Ongoing Challenges. *IEEE Communications Magazine* 43(10): 85–91.

Sarasvathy, S. D., and Dew, N. S., Velamuri, R., and Venkataraman, S. 2003. Three Views of Entrepreneurial Opportunity. In Z. J. Acs and D. B. Audretsch, eds., *Handbook of Entrepreneurship Research*. Kluwer.

Saxenian, AnnaLee. 1994. *Regional Advantage: Culture and Competition in Silicon Valley and Route 128*. Harvard University Press.

Saxenian, AnnaLee. 2002. *Local and Global Networks of Immigrant Professionals in Silicon Valley*. Public Policy Institute of California.

Scacchi, W. 2002. Understanding the Requirements for Developing Open Source Software Systems. *IEE Software Proceedings* 149(1): 24–39.

Scherer, F. M. 2001. The Innovation Lottery. In R. C. Dreyfuss, D. L. Zimmerman, and H. First, eds., *Expanding the Boundaries of Intellectual Property: Innovation Policy for the Knowledge Society*. Oxford University Press.

Schiesel, S. 2002. A New Model for AOL May Influence Cable's Future. *New York Times*, August 26.

Schilling, M. A. 2000. Toward a General Modular Systems Theory and its Application to Interfirm Product Modularity. *Academy of Management Review* 25(2): 312–334.

Schilling, M. A., and Kevin Steensma, H. 2001. The Use of Modular Organizational Forms: An Industry-Level Analysis. *Academy of Management Journal* 44(6): 1149–1168.

Schmidt, D. C., Stal, M., Rohnert, H., and Buschmann, F. 2000. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, volume 2. Wiley.

Schoen, S. 2007a. Comcast and BitTorrent. *EFF Deeplinks Blog*, September 13. <http://www.eff.org/deeplinks/2007/09/comcast-and-bittorrent>.

Schoen, S. 2007b. EFF Tests Agree with AP: Comcast Is Forging Packets to Interfere with User Traffic. *EFF Deeplinks Blog*, October 19. <http://www.eff.org/deeplinks/2007/10/eff-tests-agree-ap-comcast-forging-packets-tointerfere>.

- Schrage, M. 1997. The IQ Q&A: Steve Riggio. *Adweek.com*, August 18. http://www.adweek.com/aw/eseach/article_display.jsp?vnu_content_id=518353.
- Schwartz, M. 2000. The Economic Logic for Conditioning Bell Entry into Long Distance on the Prior Opening of Local Markets. *Journal of Regulatory Economics* 18(3): 247–288.
- Scotchmer, S. 1991. Standing on the Shoulders of Giants: Cumulative Research and the Patent Law. *Journal of Economic Perspectives* 5(1): 29–41.
- Scotchmer, S. 1996. Protecting Early Innovators: Should Second-Generation Products Be Patentable? *Rand Journal of Economics* 27(2): 322–331.
- Scotchmer, S. 2004. *Innovation and Incentives*. MIT Press.
- Scott, W. R. 2000. *Institutions and Organizations*, second edition. Sage.
- SearchIgnite and RBC Capital Markets. 2007. *Market Share Trends Within the Engines, and Their Impact on Brand Marketers*, July 17. <http://www.searchignite.com/whitepapers/rbc-searchignite-july07.pdf>
- Selby, R. W., and Basili, V. R. 1991. Analyzing Error-Prone System Structure. *IEEE Transactions on Software Engineering* 17(2): 141–152.
- Sendmail, Inc. 2008. Sendmail Corporate Fact Sheet. July.
- Senie, D. 2002. Network Address Translator (NAT)-Friendly Application Design Guidelines. Request for Comments 3235. IETF.
- Shah, D. 2006. On Startups: Patterns and Practices of Contemporary Software Entrepreneurs. Master's thesis. Massachusetts Institute of Technology.
- Shah, S., and Tripsas, M. 2004. When Do User-Innovators Start Firms? Towards a Theory of User Entrepreneurship. Working paper, University of Illinois.
- Shane, S. 2000. Prior Knowledge and the Discovery of Entrepreneurial Opportunities. *Organization Science* 11(4): 448–469.
- Shane, S., and Venkataraman, S. 2000. The Promise of Entrepreneurship as a Field of Research. *Academy of Management Review* 25(1): 217–226.
- Shapiro, A. L. 1998. The Disappearance of Cyberspace and the Rise of Code. *Seton Hall Constitutional Law Journal* 8(3): 703–724.
- Shapiro, C., and Varian, H. R. 1999. *Information Rules: A Strategic Guide to the Network Economy*. Harvard Business School Press.
- Sharma, P., Perry, E., and Malpani, R. 2003. IP Multicast Operational Network Management: Design, Challenges, and Experiences. *IEEE Network* 17(2): 49–55.
- Shaw, M., and Garlan, D. 1996. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall.

- Shelanski, H. A. 2007. Adjusting Regulation to Competition: Toward a New Model for U.S. Telecommunications Policy. *Yale Journal on Regulation* 24(1): 55–105.
- Shelanski, H. A., and Klein, P. G. 1999. Empirical Research in Transaction Cost Economics: A Review and Assessment. In G. R. Carroll and D. J. Teece, eds., *Firms, Markets, and Hierarchies: The Transaction Cost Economics Perspective*. Oxford University Press.
- Shelanski, H. A., and Sidak, G. J. 2001. Antitrust Divestiture in Network Industries. *University of Chicago Law Review* 68(1): 1–100.
- Sibley, D. S., and Weisman, D. L. 1998. Raising Rivals' Costs: The Entry of an Upstream Monopolist into Downstream Markets. *Information Economics and Policy* 10(4): 451–470.
- Sidak, J. G. 2006. A Consumer-Welfare Approach to Network Neutrality Regulation of the Internet. *Journal of Competition Law and Economics* 2(3): 349–474.
- Silberschatz, A., Galvin, P. B., and Gagne, G. 2005. *Operating Systems Concepts*, seventh edition. Wiley.
- Simon, H. A. 1961. *Administrative Behavior*, second edition. Macmillan. Original Publication: 1947.
- Simon, H. A. 1969. *The Sciences of the Artificial*. MIT Press.
- Simon, H. A. 1957. *Models of Man: Social and Rational*. Wiley.
- Slater, M. 2008. Can Rails Scale? Absolutely! April 6. <http://www.buildingwebapps.com/articles/6419-can-rails-scale-absolutely>.
- Sobrero, M., and Roberts, E. B. 2001. The Trade-Off between Efficiency and Learning in Interorganizational Relationships for Product Development. *Management Science* 47(4): 493–511.
- Sobrero, M., and Roberts, E. B. 2002. Strategic Management of Supplier-Manufacturer Relations in New Product Development. *Research Policy* 31(1): 159–182.
- Sobrero, M., and Schrader, S. 1998. Structuring Inter-firm Relationships: A Meta-analytic Approach. *Organization Studies* 19(4): 585–615.
- Solum, L. B., and Chung, M. 2003. The Layers Principle: Internet Architecture and the Law. Public Law and Legal Theory Research Paper 55, University of San Diego School of Law.
- Solum, L. B., and Chung, M. 2004. The Layers Principle: Internet Architecture and the Law. *Notre Dame Law Review* 79(3): 815–948.
- Sommerville, I. 1996. *Software Engineering*, fifth edition. Addison-Wesley.

- Sosa, M. E., Eppinger, S. D., Pich, M., McKendrick, D. G., and Stout, S. K. 2002. Factors That Influence Technical Communication in Distributed Product Development: An Empirical Study in the Telecommunications Industry. *IEEE Transactions on Engineering Management* 49(1): 45–58.
- Sosa, M. E., Eppinger, S. D., and Rowles, C. M. 2000. Understanding the Effects of Product Architecture on Technical Communication in Product Development Organizations. Working Paper 4130, MIT Sloan School of Management.
- Sosa, M. E., Eppinger, S. D., and Rowles, C. M. 2002. The Coupling of Product Architecture and Organizational Structure in Complex Product Development. Working Paper 2002/76/TM, INSEAD.
- Sosa, M., Eppinger, S. D., and Rowles, C. M. 2004. The Misalignment of Product Architecture and Organizational Structure in Complex Product Development. *Management Science* 50(12): 1674–1689.
- Sosa, M., Gargiulo, M., and Rowles, C. 2007. Component Connectivity, Team Network Structure and the Attention to Technical Interfaces in Complex Product Development. Research Paper 2007/68/TOM/OB, INSEAD.
- Souder, W. E., Sherman, J. D., and Davies-Cooper, R. 1998. Environmental Uncertainty, Organizational Integration, and New Product Development Effectiveness: A Test of Contingency Theory. *Journal of Product Innovation Management* 15(6): 520–533.
- SourceForge.net. 2008. What Is SourceForge.net? *SourceForge.net*, December 18. <http://alexandria.wiki.sourceforge.net/What+is+SourceForge.net%3F>.
- Spector, R. 2000. *Amazon.com: Get Big Fast*. HarperCollins.
- Speta, J. B. 2000a. Handicapping the Race for the Last Mile?: A Critique of Open Access Rules for Broadband Platforms. *Yale Journal on Regulation* 17(1): 39–92.
- Speta, J. B. 2000b. The Vertical Dimension of Cable Open Access. *University of Colorado Law Review* 71(4): 975–1010.
- Spicer, D. E. 1998. Online in Class: Melding Cyber and Realspace in a Law School Seminar.
- Spolsky, J. 2000. Strategy Letter I: Ben and Jerry's vs. Amazon. May 12. <http://www.joelonsoftware.com/articles/fog0000000056.html>.
- Spolsky, J. 2003. Fixing Venture Capital. June 3. <http://www.joelonsoftware.com/articles/VC.html>.
- Sproull, R. F., and Cohen, D. 1978. High-Level Protocols. *Proceedings of the IEEE* 66(11): 1371–1386.
- Srikanth, K., and Puranam, P. 2007. Coordination in Distributed Organizations. Working paper, London Business School.

- Srikanth, K., and Puranam, P. 2008. Coordination in Business Process Offshoring. Druid Working Paper 08-16.
- Srisuresh, P., and Egevang, K. 2001. Traditional IP Network Address Translator (Traditional NAT). Request for Comments 3022. IETF.
- Srisuresh, P., and Holdrege, M. 1999. IP Network Address Translator (NAT) Terminology and Considerations. Request for Comments 2663. IETF.
- Stanley, K. B. 2000. Toward International Settlement Reform: FCC Benchmarks Versus ITU Rates. *Telecommunications Policy* 24(10–11): 843–863.
- Staudenmayer, N., Tripsas, M., and Tucci, C. L. 2005. Interfirm Modularity and Its Implications for Product Development. *Journal of Product Innovation Management*, 22(4), 303–321.
- Stavrou, A., Keromytis, A. D., Nieh, J., Misra, V., and Rubenstein, D. 2005. MOVE: An End-to-End Solution To Network Denial of Service. *12th Annual Network and Distributed System Security Symposium*.
- Stephenson, S. 2005. Side-Business Software: The Neglected Software Market. *Signal vs. Noise*, July 24. http://www.37signals.com/svn/archives2/sidebusiness_software_the_neglected_software_market.php.
- Sterbenz, J. P. G., and Touch, J. D. 2001. *High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication*. Wiley.
- Stern, S. 2005. Economic Experiments: The Role of Entrepreneurship in Economic Prosperity. In *Understanding Entrepreneurship: A Research and Policy Report*. Ewing Marion Kauffman Foundation.
- Stevens, W.P., Myers, G. J., and Constantine, L. L. 1999. Structured Design (Reprint). *IBM Systems Journal* 38(2–3): 231–256.
- Steward, D. V. 1981. The Design Structure System: A Method for Managing the Design of Complex Systems. *IEEE Transactions on Engineering Management* 28(3): 71–74.
- Stewart, B. 2001. An Interview with the Creator of Ruby. *O'Reilly Linux Devcenter*, November 29. <http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html>.
- St. Johns, M., and Huston, G. 2003. Considerations on the Use of a Service Identifier in Packet Headers. Request for Comments 3639. IETF.
- Stone, J., and Partridge, C. 2000. When the CRC and TCP Checksum Disagree. *Computer Communication Review* 30(4): 309–319.
- Stross, R. 2005. Why Time Warner Has Fallen in Love with AOL, Again. *New York Times*, September 25.

- Sturgeon, T. J. 2002. Modular Production Networks: A New American Model of Industrial Organization. *Industrial and Corporate Change* 11(3): 451–496.
- Sullivan, D. 1998. GoTo Going Strong. *Search Engine Watch*, July 1. <http://searchenginewatch.com/showPage.html?page=2166331>.
- Sullivan, D. 2002a. Overture and Inktomi Out, Google In at AOL. *Search Engine Watch*, May 1. <http://searchenginewatch.com/2164731>.
- Sullivan, D. 2002b. Overture Wins Yahoo, What Will Happen with Google? *Search Engine Watch*, May 6. <http://searchenginewatch.com/2164811>.
- Sullivan, K. J., Chalasani, P., Jha, S., and Sazawal, V. 1999. Software Design as an Investment Activity: A Real Options Perspective. In L. Trigeorgis, ed., *Real Options and Business Strategy: Applications to Decision Making*. Risk Books.
- Sullivan, K. J., Griswold, W. G., Cai, Y., and Hallen, B. 2001. The Structure and Value of Modularity in Software Design. *8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering*.
- Sunstein, C. R. 1996a. On the Expressive Function of Law. *University of Pennsylvania Law Review* 144(5): 2021–2053.
- Sunstein, C. R. 1996b. Social Norms and Social Roles. *Columbia Law Review* 96(4): 903–968.
- Svensson, P. 2007. Comcast Blocks Some Internet Traffic. Associated Press, October 19. <http://www.sfgate.com/cgi-bin/article.cgi?f=/n/a/2007/10/19/financial/f061526D54.DTL>.
- Takeishi, A. 2001. Bridging Inter- and Intra-Firm Boundaries: Management of Supplier Involvement in Automobile Product Development. *Strategic Management Journal* 22(5): 403–433.
- Tanenbaum, A. S. 2003. *Computer Networks*, fourth edition. Prentice-Hall.
- Tanenbaum, A. S. 2008. *Modern Operating Systems*, third edition. Prentice-Hall.
- Taylor, S., and Hettick, L. 2008. Verizon: IP Multimedia Subsystem on the Cusp of Being Viable in 2008. *Network World*, January 2. <http://www.networkworld.com/newsletters/converg/2008/1231converge2.html>.
- Teece, D. J., Pisano, G., and Shuen, A. 1997. Dynamic Capabilities and Strategic Management. *Strategic Management Journal* 18(7): 509–533.
- Terwiesch, C., and Xu, Y. 2008. Innovation Contests, Open Innovation, and Multiagent Problem Solving. *Management Science* 54(9): 1529–1543.
- Thaler, D., and Aboba, B. 2008. What Makes for a Successful Protocol? Request for Comment 5218. IETF.

- Thelen, K. 1999. Historical Institutionalism in Comparative Politics. *Annual Review of Political Science* 2: 369–404.
- Thierer, A. D. 2004. “Net Neutrality”: Digital Discrimination or Regulatory Gamesmanship in Cyberspace? Policy Analysis 507, Cato Institute.
- 37signals. 2006. *Getting Real: The Smarter, Faster, Easier Way to Build a Successful Web Application*. 37signals, L.L.C.
- Thompson, J. D. 1967. *Organizations in Action*. McGraw-Hill.
- Time Warner. 2008. *Form 10-K for the Fiscal Year Ended December 31, 2007*.
- Tirole, J. 1988. *The Theory of Industrial Organization*. MIT Press.
- Tomlinson, R. undated-b. The First Email. <http://openmap.bbn.com/~tomlinso/firstemailframe.html>.
- Topolski, Robert M. 2008. Your Letter Addressed to FCC Chairman Martin, dated March 28, 2008. Letter to David Cohen. April 3.
- Trigeorgis, L. 1996. *Real Options: Managerial Flexibility and Strategy in Resource Allocation*. MIT Press.
- Troy, D. A., and Zweben, S. H. 1981. Measuring the Quality of Structured Designs. *Journal of Systems and Software* 2(2): 113–120.
- Tsang, E. W. K. 2000. Transaction Cost and Resource-based Explanations of Joint Ventures: A Comparison and Synthesis. *Organization Studies* 21(1): 215–242.
- Tuomi, I. 2002. *Networks of Innovation: Change and Meaning in the Age of the Internet*. Oxford University Press.
- Turner, J. S., and Taylor, D. E. 2005. Diversifying the Internet. *IEEE GLOBECOM 2005*.
- Tushman, M. L. 1977. Special Boundary Roles in the Innovation Process. *Administrative Science Quarterly* 22(4): 587–605.
- Tushman, M. L., and Anderson, P. 1986. Technological Discontinuities and Organizational Environments. *Administrative Science Quarterly* 31(3): 439–465.
- Tushman, M. L., and Anderson, P. 1990. Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change. *Administrative Science Quarterly* 35(4): 604–633.
- U.S. Department of Justice and Federal Trade Commission. 1984. Non-Horizontal Merger Guidelines. Originally issued as part of U.S. Department of Justice Merger Guidelines, June 14, 1984. <http://www.usdoj.gov/atr/public/guidelines/2614.htm>.
- U.S. Senate Committee on Banking. 1999. *Financial Services Modernization Act. Gramm-Leach-Bliley. Summary of Provisions*.

- Ulhøi, J. P. 2004. Open Source Development: A Hybrid in Innovation and Management Theory. *Management Decision* 42(9): 1095–1114.
- Ulrich, K. 1995. The Role of Product Architecture in the Manufacturing Firm. *Research Policy* 24(3): 419–440.
- Ulrich, K. T., and Ellison, D. J. 2005. Beyond Make-Buy: Internalization and Integration of Design and Production. *Production and Operations Management* 14(3): 315–330.
- Ulrich, K. T., and Eppinger, S. D. 2000. *Product Design and Development*, second edition. Irwin McGraw-Hill.
- United States Government Accountability Office. 2006. *Report to Congressional Committees; Telecommunications; Broadband Deployment Is Extensive throughout the United States, but It Is Difficult to Assess the Extent of Deployment Gaps in Rural Areas*. GAO-06-426.
- Uzzi, B. 1996. The Sources and Consequences of Embeddedness for the Economic Performance of Organizations: The Network Effect. *American Sociological Review* 61(4): 674–698.
- van den Bulte, C., and Moenaert, R. K. 1998. The Effects of R&D Team Co-Location on Communication Patterns among R&D, Marketing, and Manufacturing. *Management Science* 44(11, Part 2): S1–S18.
- van der Berg, R. 2008. How the 'Net Works: An Introduction to Peering and Transit. *Ars Technica*, September 2. <http://arstechnica.com/guides/other/peering-and-transit.ars/1>.
- van Osnabrugge, M., and Robinson, R. 2000. *Angel Investing: Matching Start-up Funds with Start-up Companies—The Guide for Entrepreneurs and Individual Investors*. Jossey-Bass.
- van Schewick, B. 2004. Architecture and Innovation: The Role of the End-to-End Arguments in the Original Internet. PhD dissertation, Technical University Berlin.
- van Schewick, B. 2007. Towards an Economic Framework for Network Neutrality Regulation. *Journal on Telecommunications and High Technology Law* 5(2): 329–391.
- van Schewick, B. 2008. Written Testimony before the Federal Communications Commission at its Second En Banc Hearing on Broadband Management Practices, April 17.
- van Schewick, B. 2009. Point/Counterpoint. Network Neutrality Nuances. *Communications of the ACM* 52(2): 31–37.
- van Vliet, H. 2000. *Software Engineering: Principles and Practice*, second edition. Wiley.

- Varghese, S. 2003. Survey Questions Business Relevance of Netcraft Web Server Data. November 26. <http://www.theage.com.au/articles/2003/11/26/1069522639857.html>.
- Varian, H. R. 1989. Price Discrimination. In R. L. Schmalensee and R. D. Willig, eds., *Handbook of Industrial Organization*, volume 1. Elsevier.
- Varian, H. R. 1999. *Intermediate Microeconomics: A Modern Approach*, fifth edition. Norton.
- Varian, H. R. 2006. The Economics of Internet Search. *Rivista di Politica Economica* 96(6): 177–191.
- Varian, H. R. 2007. Kaizen, That Continuous Improvement Strategy, Finds its Ideal Environment. *New York Times*, February 8.
- Viscusi, W. K., Harrington, J. E., Jr., and Vernon, J. M. 2005. *Economics of Regulation and Antitrust*, fourth edition. MIT Press.
- Vise, D. A., and Malseed, M. 2005. *The Google Story*. Delacorte.
- von Hippel, E. 1988. *The Sources of Innovation*. Oxford University Press.
- von Hippel, E. 1990. Task Partitioning: An Innovation Process Variable. *Research Policy* 19(5): 407–418.
- von Hippel, E. 1994. “Sticky Information” and the Locus of Problem Solving: Implications for Innovation. *Management Science* 40(4): 429–439.
- von Hippel, E. 1998. Economics of Product Development by Users: The Impact of “Sticky” Local Information. *Management Science* 44(5): 629–644.
- von Hippel, E. 2001. PERSPECTIVE: User Toolkits for Innovation. *Journal of Product Innovation Management* 18(4): 247–257.
- von Hippel, E. 2002. Horizontal Innovation Networks—by and for Users. Working Paper 4366-02, MIT Sloan School of Management.
- von Hippel, E. 2005. *Democratizing Innovation*. MIT Press.
- von Hippel, E. 2007. Horizontal Innovation Networks—by and for Users. *Industrial and Corporate Change* 16(2): 293–315.
- von Hippel, E., and von Krogh, G. 2003. Open Source Software and the “Private-Collective” Innovation Model: Issues for Organization Science. *Organization Science* 14(2): 209–223.
- von Krogh, G., and von Hippel, E. 2006. The Promise of Research on Open Source Software. *Management Science* 52(7): 975–983.
- Weiser, P. 2008. The Next Frontier for Network Neutrality. *Administrative Law Review* 60(2): 273–322.

- Weiser, P. J. 2001. Internet Governance, Standard Setting, and Self-Regulation. *Northern Kentucky Law Review* 28(4): 822–845.
- Weiser, P. J. 2003. Toward a Next Generation Regulatory Strategy. *Loyola University of Chicago Law Journal* 35(1): 41–85.
- Weisman, D. L. 1998. The Incentive to Discriminate by a Vertically-Integrated Regulated Firm: A Reply. *Journal of Regulatory Economics* 14(1): 87–91.
- Weisman, D. L., and Williams, M. A. 2001. The Costs and Benefits of Long-Distance Entry: Regulation and Non-Price Discrimination. *Review of Industrial Organization* 18(3): 275–282.
- Weisman, R. 2005. ICQ Founder Has a New Message. *Boston Globe*, April 4.
- Weiss, M. B., and Shin, S. J. 2004. Internet Interconnection Economic Model and Its Analysis: Peering and Settlement. *NETNOMICS: Economic Research and Electronic Networking* 6(1): 43–57.
- Weitzman, M. L. 1980. The “Ratchet Principle” and Performance Incentives. *Bell Journal of Economics* 11(1): 302–308.
- Wernerfelt, B. 1984. A Resource-Based View of the Firm. *Strategic Management Journal* 5(2): 171–180.
- West, J. 2007. The Economic Realities of Open Standards: Black, White, and Many Shades of Gray. In S. Greenstein and V. Stango, eds., *Standards and Public Policy*. Cambridge University Press.
- Whinston, M. D. 1990. Tying, Foreclosure, and Exclusion. *American Economic Review* 80(4): 837–859.
- Whinston, M. D. 2001. Exclusivity and Tying in U.S. v. Microsoft: What We Know; and Don’t Know. *Journal of Economic Perspectives* 15(2): 63–80.
- Whitt, R., and Faber, J. 2008. Cone of Silence (Finally) Lifts on the Spectrum Auction. *Google Public Policy Blog*, April 3. <http://googlepublicpolicy.blogspot.com/2008/04/cone-of-silence-finally-lifts-on.html>.
- Williamson, O. E. 1975. *Markets and Hierarchies: Analysis and Antitrust Implications*. Free Press.
- Williamson, O. E. 1979. Assessing Vertical Market Restrictions: Antitrust Ramifications of the Transaction Cost Approach. *University of Pennsylvania Law Review* 127(4): 953–993.
- Williamson, O. E. 1985. *The Economic Institutions of Capitalism: Firms, Markets, Relational Contracting*. Free Press.
- Williamson, O. E. 1991. Comparative Economic Organization: The Analysis of Discrete Structural Alternatives. *Administrative Science Quarterly* 36 (June): 269–296.

- Williamson, O. E. 1999. Strategy Research: Governance and Competence Perspectives. *Strategic Management Journal* 20(12): 1087–1108.
- Wilson, M. 2008. Apple Rejects MailWrangler App for “Leading to User Confusion.” *GIZMODO*, September 22. <http://gizmodo.com/5053232/apple-rejects-mailwrangler-app-for-leading-to-user-confusion>.
- Wischik, D. 2007. Short Messages. Royal Society Workshop on Networks: Modelling and Control.
- Wise, A. S., and Duwadi, K. 2005. Competition between Cable Television and Direct Broadcast Satellite: The Importance of Switching Costs and Regional Sports Networks. *Journal of Competition Law and Economics* 1(4): 679–705.
- Wolfe, G. 1994. The (Second Phase of the) Revolution Has Begun. *Wired*, October. <http://www.wired.com/wired/archive/2.10/mosaic.html>.
- Wolverton, T. 2002. GoTo.com to See Profit Sooner Than Expected. *CNET News*, January 2. http://news.cnet.com/GoTo.com-to-see-profit-sooner-than-expected/2100-1023_3-270585.html.
- Woodard, C. J. 2006. Architectural Strategy and Design Evolution in Complex Engineered Systems. PhD dissertation, Harvard University.
- Woodard, C. J. 2008. Architectural Control Points (Long Version, January 23). Working paper.
- Wright, J. 2002. Access Pricing under Competition: An Application to Cellular Networks. *Journal of Industrial Economics* 50(3): 289–315.
- Wright, J. 2004. One-sided Logic in Two-sided Markets. *Review of Network Economics* 3(1): 42–63.
- Wu, T. 2003a. Network Neutrality and Broadband Discrimination. *Journal on Telecommunications and High Technology Law* 2: 141–175.
- Wu, T. 2003b. When Code Isn’t Law. *Virginia Law Review* 89(4): 101–169.
- Wu, T. 2004. The Broadband Debate: A User’s Guide. *Journal on Telecommunications and High Technology Law* 3(1): 69–95.
- Wu, T. 2006a. Testimony before the House Committee on the Judiciary: Telecom and Antitrust Task Force at its hearing on: Network Neutrality: Competition, Innovation, and Nondiscriminatory Access. April 25. 109th Congress, 2nd Session.
- Wu, T. 2006b. Why You Should Care about Network Neutrality. *Slate*, May 1. <http://www.slate.com/id/2140850>.
- Wu, T. 2007. Wireless Carterfone. *International Journal of Communication* 1: 389–426.
- Wu, T., and Lessig, L.. 2003. Ex Parte Submission to Federal Communications Commission. CS Dkt. No. 02-52, August 22.

- Xiaowei, Y., Whetherall, D., and Anderson, T. 2005. A DoS-Limiting Network Architecture. *Computer Communication Review* 35(4): 241–252.
- Yahoo. 2005. The History of Yahoo!—How It All Started. <http://docs.yahoo.com/info/misc/history.html>.
- Yahoo. 2008a. *Form 8-K. March 18*.
- Yahoo. 2008b. *Form 10-K for the Fiscal Year Ended December 31, 2007*.
- Yahoo! Finance. 2008a. AT&T Inc. (T). *Yahoo! Finance*, June 21. <http://finance.yahoo.com/q?s=T>.
- Yahoo! Finance. 2008b. Comcast Corp. (CMSA). *Yahoo! Finance.*, June 21. <http://finance.yahoo.com/q/ks?s=CMCSA>.
- Yamauchi, Y., Yokozawa, M., Shinohara, T., and Ishida, T. 2000. Collaboration with Lean Media: How Open-Source Software Succeeds. *2000 ACM Conference on Computer Supported Cooperative Work*.
- Yoo, C. S. 2002. Vertical Integration and Media Regulation in the New Economy. *Yale Journal on Regulation* 19(1): 171–300.
- Yoo, C. S. 2004. Would Mandating Broadband Network Neutrality Help or Hurt Competition? A Comment on the End-to-End Debate. *Journal on Telecommunications and High Technology Law* 3(1): 23–68.
- Yoo, C. S. 2005. Beyond Network Neutrality. *Harvard Journal of Law and Technology* 19(1): 1–77.
- Yoo, C. S. 2006. Network Neutrality and the Economics of Congestion. *Georgetown Law Journal* 94 (June): 1847–1908.
- Yoo, C. S. 2007. What Can Antitrust Contribute to the Network Neutrality Debate? *International Journal of Communication* 1: 493–530.
- Yourdon, E., and Constantine, L. L. 1975. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice-Hall.
- Zittrain, J. 2003. Internet Points of Control. *Boston College Law Review* 44(2): 653–688.
- Zittrain, J. 2008. *The Future of the Internet and How to Stop It*. Yale University Press.
- Zittrain, J. 2009. ISPs helping with botnets. E-Mail to ip@v2.listbox.com Mailing List. October 12. <http://seclists.org/interesting-people/2009/Oct/95>.

Index

- Access charges
 - and ability to innovate, 292, 293
 - and cost of innovation, 278–280, 290–292
 - and discrimination, 232–240
 - long-distance, 241–243, 254, 255, 258
 - network providers' interest in, 372, 373
 - policy implications of, 220, 221, 278–280, 290–293, 370, 371
- Adobe, 333
- Adoption. *See* Cost of adoption; Deployment; Economic growth
- ADSL. *See* Asymmetric Digital Subscriber Line; Asymmetric bandwidth
- Advertising, 233–240, 253, 254, 314–318, 321, 322. *See also* Search-engine advertising
- Advertising networks, 238, 239, 254
- AdWords, 316, 317. *See also* Google
- AltaVista, 321
- Amazon.com
 - early history and funding, 207, 210, 311, 320, 328, 333
 - and economies of scale, 143, 144
 - and incumbents, 320
 - and real-options approach to uncertainty, 146
- America Online. *See* AOL
- Andreessen, Marc, 307–309, 320, 321, 338
- Android, 391
- Angel investors, 207–210, 312
- Antitrust law, 25, 31, 218, 219, 273
- AOL
 - business model of, 231, 236
 - and discrimination, 258, 267
 - early funding of, 333
 - as incumbent, 320, 322
 - merger with TimeWarner, 250, 251, 258
 - increasing switching costs, 262
 - Instant Messenger, 146, 250, 251
- Apache Web Server
 - as component of open-source software stack, 144
 - cost of making available, 143
 - early history of, 321, 337–339, 341
 - and free revealing, 341, 345
 - as user-driven innovation, 337
- Apple, 198, 332, 350. *See also* App Store; iMac; iPhone
- Appliances, 387
- Application autonomy, 71, 72, 106, 107. *See also* Broad version
- Application awareness, 72–75, 217, 218, 286–289. *See also* Controllable architecture; Deviations from broad version
- Application blindness, 72–75, 217, 218, 286–289, 379. *See also* Discrimination; Network neutrality

- Application control, 72–74, 217, 218, 286–289. *See also* Controllable architecture; Control over application deployment; Control over application development; Control over network use
- Application layer, 84, 87, 88. *See also* Applications; Broad version; Deviations from broad version
- Application-layer multicast, 159–161
- Application-level intermediaries, 92, 107–110, 383–385
- Application programming interfaces (APIs)
and autonomous innovation, 120
availability of, 202, 203
cost of innovating on, 119, 121, 128, 135
and systemic innovation, 121, 199, 200, 214
terminology of, 47, 48
- Applications
cost of innovation in, 138–148, 289–292, 385, 386
dependencies on, 88–90, 145
deployment requirements for, 140, 144, 145, 152, 155, 293–295, 349–351, 385, 386 (*see also* Control over application deployment; Deployment context)
and end-to-end arguments (*see* End-to-end arguments, technical constraints on)
innovation in (*see* Broad version; Deviations from broad version)
rate of change of, 144–147, 155–157, 351, 353
specific (*see specific applications*)
types of (*see* Client-server applications; Peer-to-peer applications; Server-based applications; Web-based applications)
- Application-specific functionality, 67, 68, 76. *See also* Broad version; General functionality
compression as, 71
connection-orientation as, 97–99
lowest layer implementing, 86, 87, 98, 380
reliable data transfer as, 97–103, 139, 140
- Application-specific transport prices, 273–275, 373. *See also* Pricing of Internet services
- App Store, 350
- Architecture. *See also* Network architecture
and characteristics of actors, 30
cost of innovation, 115–118
design principles for (*see* Broad version; Integrated design; Layering; Modularity; Narrow version; Relaxed layering)
and design teams, 167, 168, 323, 324
and economic constraints, 26, 28, 31, 32
and economics, 3, 4, 11–15, 32, 33, 387–389, 392
and economic systems, 28, 32, 33
and firm strategies (*see* Strategic choices of system architects; Strategic design)
and functional requirements, 3, 21, 22, 32
and innovation, 28–33, 115–118, 165, 166, 215, 216
legal constraints, 26–28, 31, 32
and market structure (*see* Market structure)
and organization of detailed design and innovation, 165, 166
as politics, 19, 20
and product development process, 20
as quality, 21–23

- and regulation, 24–28, 31, 32
- and relationships among actors, 31
- and social norms, 26, 28, 31, 32
- and system qualities, 21–23
- terminology of, 19, 21
- ARPANET, 206, 302, 307, 309, 337
- ARPANET Host-to-Host Protocol (AHHP), 91
- Ask Jeeves, 335
- Asset specificity, 188. *See also*
 - Transaction cost
- Asymmetric bandwidth
 - and application innovation, 70, 286, 352, 353
 - and applications, 70, 264, 265
 - and broad version, 70
 - in current Internet, 70, 264, 265, 286, 372
 - and political uses, 364
- Asymmetric Digital Subscriber Line (ADSL), 70, 286. *See also* Asymmetric bandwidth
- AT&T, 269, 312, 372
- Auctions, 391, 392
- AuctionWeb, 206
- Autonomous change, 120, 121, 135, 136
- Autonomous innovation. *See also*
 - Systemic innovation
- in Internet architecture, 204–213
- in modular architectures, 183–185, 197, 198
- terminology of, 183
- Average cost. *See* Economies of scale
- Backlinks, 310
- BackRub, 310
- Backward compatibility. *See also* Visible module
 - of application-layer protocols, 146, 155–157
 - of Internet Protocol, 151, 152
 - of link-layer protocols, 150
 - terminology of, 136
 - of transport-layer protocols, 151, 152, 155–157
- Bandwidth management. *See* Network management
- Barnes & Noble, 320
- Basecamp, 208, 335
- Baxter's law, 226
- Behavioral economics, 33, 78, 264, 374, 375
- Behlendorf, Brian, 337, 338
- Benchmark Capital, 210
- Berkman Center for Internet and Society, 335, 336
- Berners-Lee, Tim, 302, 303, 308, 321, 337, 341
- Best-effort service, 85, 142, 153, 154. *See also* Quality of service
- Bezos, Jeff, 207, 311, 320
- Bhatia, Sabeer, 304, 306
- Bina, Eric, 308, 309
- BitTorrent, 70, 88, 107, 143, 260, 261, 265, 266, 351, 363
- Black, Fischer, 123
- Blogger, 207, 209, 210, 347, 348
- Bloglines, 210, 335, 341
- Blogs, 348, 360, 361
- Blumenthal, Marjory, 58
- Bolt, Beranek and Newman, 302, 307, 309
- Boundary choices of firms, 168–171. *See also* Coordination cost; Transaction cost
- Bounded rationality, 33, 186, 300. *See also* Transaction cost
- Brin, Sergey, 207, 307, 310, 316–318, 321, 322, 332
- Broadband deployment, 365, 366, 370, 371
- Broad version of end-to-end arguments, 67–81, 377–379
 - and ability to innovate, 204–213, 292, 293

- Broad version of end-to-end arguments
(cont.)
- and amount of application innovation, 144–147, 351–356
 - and application blindness, 72–75, 217, 218, 286–289, 379
 - and application deployment, 144, 145, 152, 155, 293–295, 349–351
 - benefits of, 69–74, 355, 365
 - and broadband deployment, 365, 366, 370, 371
 - and client-server applications, 110
 - and cloud computing, 108–110, 384, 385
 - and narrow version, 75–81
 - and competitive strategies, 217, 218
 - and control over end hosts, 72, 387
 - and cost of application innovation, 138–148, 289–292, 385, 386
 - costs of, 74, 75, 365–368
 - in current Internet, 218, 286, 287, 368, 371, 372, 383–392
 - decision to deviate from broad version case-by-case, 77, 78, 367, 368, 373–375
 - and deep packet inspection (*see* Deep packet inspection)
 - and deployment requirements, 140, 144, 152, 153, 293–295, 349–351, 385, 386
 - deviations from (*see* Deviations from broad version)
 - and distributed applications with application-level intermediaries, 107–110, 383–385
 - and economic environment for application innovation, 285–295, 381–386
 - and economic environment for network use, 293, 294, 359–365
 - and e-mail, 108–110, 383, 384
 - and encryption, 367
 - and fate sharing, 104–106, 366, 382
 - and firewalls (*see* Firewalls)
 - future of as design principle, 103, 104, 355, 356, 368–371, 388, 389
 - and generality of end hosts, 387
 - history of, 58, 59, 90, 110–112
 - horizontal dimension of, 57, 58, 76
 - and independent innovation, 204, 211, 293, 345–348
 - and innovation without permission, 204, 211, 293, 345–348
 - and layering, 103–105
 - and middleboxes (*see* Middleboxes)
 - and mobile Internet, 389–392
 - and network-address translators (*see* Network-address translators)
 - and network neutrality, 72–74, 80, 81, 379
 - network providers' interests in, 371–375
 - and non-discrimination, 72–74, 80, 81, 379
 - and organization of application innovation, 204, 210, 211, 292, 294
 - and original Internet architecture, 96–103, 110–112, 380, 381
 - and packet filters, 74, 75
 - and peer-to-peer applications, 110
 - policy implications of, 72–74, 80, 81, 103, 104, 387–392
 - and political Internet uses, 359–361, 364, 365
 - and quality of service, 104–107, 142, 366, 382
 - rationale behind, 79, 378, 379
 - and risk taking with respect to uncertainty, 147, 148, 294, 295, 352
 - and security, 366–368
 - social benefits of, 355, 365
 - social costs of, 365–368
 - social tradeoff regarding, 368–371
 - and stupid network, 107, 366, 382
 - and technical constraints on application design, 78, 103, 110, 380–384

- and technical constraints on network evolution, 104–107, 366–368, 382
- and type of application innovation, 144–148, 152, 155, 295, 352, 353
- underlying tradeoff, 68, 69, 77–79, 355, 356, 367–371, 378, 379
- user choice (*see* Control over end hosts; User choice)
- vertical dimension, 57, 58, 76
- welfare implications of (*see* Social value)
- Browser. *See* Web browser
- BT, 372
- Bursty traffic, 265
- Bush, George W., 361
- Butterfield, Stewart, 207, 305, 306, 347

- Cable modem, 50, 153, 265
- Cable networks
 - and access charges for unaffiliated Internet service providers, 236
 - asymmetric bandwidth in, 70, 265
 - and cost of switching to DSL, 262, 263
 - open access to, 2, 222, 231
 - and video over Internet, 312, 327
- Cailliau, Robert, 302, 303
- Campbell, Jim, 332
- Capabilities
 - and network provider contracting, 346
 - as source of competitive advantage, 196
 - as source of inflexibility, 322–328
 - and vertical boundary choices of firms, 168–171, 195, 196, 346
- Cerf, Vinton, 99
- CERN, 302, 303, 321, 337, 338, 341
- CertifiedEmail, 267
- Chicken-and-egg problem, 245, 246, 249, 386
- Cisco Systems, 333
- Clark, David, 5, 58, 110–112
- Clark, Jim, 307, 308, 309, 320, 321, 333
- Clearwire, 392
- Client-server applications
 - and broad version, 110
 - Internet optimized for, 70, 385, 386
 - rate of change of, 146
- Cloud computing, 108–110, 384, 385
- Code and Other Laws of Cyberspace* (Lessig), 19
- Codified information, 174
- Cohen, Danny, 99
- Cohesion, 41–44
- Co-invention, 358, 359
- Collective action
 - and deployment of Internet layer innovations, 158
 - and evolution of modular architectures with distributed ownership, 199, 200
- Comcast, 260, 261, 265, 266, 312, 351
- Commercial Internet, 157–163, 383.
 - See also* Control over end hosts; Deployment context
- Competitive strategies. *See also* Strategic choices of system architects; Strategic design
 - in application-aware architectures, 217, 218
 - and architecture, 215, 216
 - and pricing of Internet services (*see* Application-specific transport prices; Flat-rate pricing; Price discrimination; Usage-based pricing)
- Complementary efficiencies. *See* Internalizing complementary efficiencies argument
- Complementary products, 232–240, 253–255, 258
- Complementor, 196
- Complexity
 - and architecture, 20, 21
 - and bounded rationality, 119, 124, 187, 188, 300

- Complexity (cont.)
 and broad version, 72, 74
 coupling and interface, 41, 44
 and integrated design, 45, 46, 119
 and layering, 48–52, 55, 56
 and limits to number of concurrent projects in core-centered architectures, 313
 and modularity, 38, 40, 41, 43, 44, 50
 and organization of design teams, 168
 and performance, 22
 and transaction cost, 186–188
 and value of innovator diversity, 300
- Confinity, 304
- Congestion management. *See* Network management
- Connectionless service, 85, 97–102, 380
- Connection-oriented service, 87, 97–102
- Constraints, 24
 and actors, 30, 31
 architecture as, 28–33
 economic, 24–28, 31, 32
 and economic systems, 24–33
 functioning of, 30, 31
 and innovation, 28–33
 interactions among, 26–28, 31
 legal, 24–28, 31, 32
 and market structure (*see* Market structure)
 social norms as, 24–28, 31, 32
- Consumer heterogeneity. *See* User heterogeneity
- Consumptive Internet uses, 360, 364
- Content-based discrimination, 266–270, 364, 365
- Content policies, 267, 269, 365
- Contracting for innovation, 194, 195, 345–348
- Controllability, 286, 287. *See also* Controllable architecture; Deviations from broad version
- Controllable architecture. *See also* Deviations from broad version
 amount of innovation in, 352–356
 and economic environment for innovation, 270–272, 289–295
 terminology of, 288, 289
 welfare implications of (*see* Social value)
- Control over application deployment.
See also Control over end hosts
 in mobile Internet, 390, 391
 relevance of, 349–351, 365, 366, 370, 371, 382, 387
 in wireline Internet, 72, 293, 294
- Control over application development.
See also Control over end hosts
 in mobile Internet, 390, 391
 relevance of, 345–348, 351, 366, 387
 in wireline Internet, 184, 185, 194, 195, 204, 211, 293, 294
- Control over end hosts
 and application innovation, 293–295, 345–351, 366, 370, 371, 382, 387
 and broad version, 72, 387
 in mobile Internet, 390, 391
 and network use, 293, 294, 362–365, 371
 in wireline Internet, 72, 152, 294, 295, 387
- Control over network use. *See also* Control over end hosts
 in mobile Internet, 390, 391
 relevance of, 362–365, 371
 in wireline Internet, 72, 293, 294
- Coordination
 and deployment of Internet layer innovations, 158
 and evolution of Internet architecture's visible information, 213, 214
 and evolution of modular architectures with distributed ownership, 199, 200
 and middleboxes, 386

- Coordination cost
 - and architecture, 179, 180
 - and design teams, 167, 168
 - and governance, 175–179
 - and integrated architectures, 180, 181
 - and modular architectures, 181–185
 - theory of, 172–179
 - and transaction cost, 169–171
- Copyright law. *See* Intellectual property law
- Core-centered architecture. *See also*
 - Deviations from broad version
 - amount of innovation in, 352–356
 - and economic environment for innovation, 289–295, 313
 - terminology of, 289
 - welfare implications of (*see* Social value)
- Core cost of innovation
 - and likelihood of innovation, 134–137
 - terminology of, 119
- Core of network,
 - layers implemented in, 57, 58
 - terminology of, 50, 378, 384, 385
- Corporate investors, 207–211, 333, 334.
 - See also* Innovators
- Correctness
 - and broad version, 79, 80
 - and integrated design, 45
 - and layering, 48
 - and modularity, 41, 42
 - and narrow version, 60–67, 79, 80
- Cost of adoption. *See* Cost of deployment
- Cost of change. *See also* Cost of innovation
- Cost of distribution, 118, 137
- Cost of innovation
 - at application layer, 138–148, 289–292, 385, 386
 - economic environment and, 116, 118
 - at Internet layer, 151–153, 383
 - at link layer, 148–151
 - terminology of, 118, 119
 - at transport layer, 151–153
- Cost of operating server infrastructure, 143, 144
- Cost of production, 118, 137
- Cost of realizing an innovation, 115, 118, 119
- Cost of system adaptation, 119, 135–137
- Coupling
 - and correctness, 41, 42, 45
 - and cost of innovation, 119, 121
 - and modifiability, 40, 43, 44
 - terminology of, 38
- Crocker, Steve, 99
- Current Internet architecture. *See also*
 - Deviations from broad version
 - application awareness and application control in, 218, 287, 371, 372, 385
 - deviations from broad version in, 70, 218, 286, 287, 368, 371, 372, 383–387
 - opaqueness of, 286, 372, 385, 386
 - welfare implications of (*see* Social value)
- Datagram, 85
- Data transfer
 - reliable (*see* Error control)
 - unreliable, 85, 87 (*see also* General functionality)
- Dearol.com, 267
- Deep packet inspection (DPI)
 - and application innovation (*see* Controllable architecture)
 - and available strategies (*see* Access charges; Discrimination; Price differentiation; Price discrimination)
 - and broad version, 72, 73, 107, 218, 385
 - in current Internet, 2, 287, 371, 372, 385
 - and layering, 73, 385

- Del.icio.us, 205, 210, 328
- Dependencies. *See* Interdependencies
- Deployment context
- commercial Internet as, 152, 158–161, 294, 295, 383
 - as constraint on evolution of a technical system, 152–163, 383
 - and control over application deployment (*see* Control over application deployment)
 - and control over network use (*see* Control over network use)
 - enterprise network as, 159–161
 - and evolution of applications, 146, 147, 152, 155–157, 294, 295
 - and evolution of Internet layer, 157–163, 383
 - and evolution of transport layer, 155–157, 386
 - and IP mobile, 157, 158
 - and IPv6, 157, 158
 - and multicast, 157–161
 - and quality of service, 157, 158
 - as rules of game, 162, 163
 - terminology of, 152
- Deployment control. *See* Control over application deployment; User choice
- Deployment, cost of, 119, 137, 138
- Deployment requirements
- for applications, 140, 144, 145, 152, 155, 293–295, 349–351, 385, 386
 - for Internet-layer protocols, 157
 - for link-layer protocols, 151
 - and network architectures, 138, 152
 - for transport-layer protocols, 156, 385, 386
- Design, detailed, 20
- Design principles, 23. *See also* End-to-end arguments; Modularity; Integrated design; Layering; Relaxed layering
- and economic systems (*see* Economic system for development and production of technical system; Economic system for use of technical system)
 - and innovation, 28–33
 - and system qualities, 23
- Design teams, 167, 168, 323, 324
- Deutsche Telekom, 263
- Deviations from broad version. *See also* Access charges; Controllable architecture; Control over application deployment; Core-centered architecture; Discrimination; Price differentiation; Price discrimination; Opaqueness
- and ability to innovate, 210, 211, 292, 293
 - and amount of application innovation, 144–147, 351–356, 385, 386
 - and application deployment, 140, 144, 145, 152, 155, 293–295, 349, 351, 385, 386
 - and competitive strategies, 217, 218
 - and cost of application innovation, 138–140, 144–148, 289–292, 385, 386
 - in current Internet, 217, 218, 286, 287, 368, 371, 372, 383–387, 389–392
 - and deep packet inspection (*see* Deep packet inspection)
 - deviations increasing controllability (*see* Deep packet inspection; IP Multimedia Subsystem)
 - deviations increasing opaqueness (*see* Asymmetric bandwidth; Deep packet inspection; Firewalls; Network-address translators)
 - and economic environment for innovation, 285–295, 381–383, 385, 386

- and firewalls (*see* Firewalls)
- and middleboxes (*see* Middleboxes)
- and network-address translators (*see* Network-address translators)
- and packet filters, 74, 75
- and risk taking with respect to uncertainty, 147, 148, 294, 295, 352
- and type of application innovation, 144–148, 152, 155, 295, 352, 353
- types of, 286–289
- Digital Equipment Corporation, 309
- Digital Millennium Copyright Act, 27
- Digital rights management system, 26, 27
- Digital Subscriber Line (DSL) networks, 69, 70, 262, 263. *See also* Asymmetric Digital Subscriber Line
- Direct network effects, 229. *See also* Network effects
- Discrimination
 - and access charges (*see* Access charges)
 - and antitrust law, 218, 219, 273
 - and bandwidth management, 260, 261, 264–266, 350, 351, 363, 365
 - and competition in primary market, 218, 219, 255–266, 391, 392
 - content-based, 266–270, 364, 365
 - cost of, 259–264
 - incentives for, 218–273, 371–375
 - and incentives to innovate, 270–273, 289–292, 352–356, 370, 371
 - instead of exclusion, 260, 261
 - in mobile Internet, 259, 260, 390–392
 - without monopolization, 232, 251–255
 - politics and, 266–270, 364, 365
 - and price discrimination, 243, 275–278
 - regulation to prevent (*see* Network neutrality)
 - underlying tradeoff, 225
 - welfare implications of, 355–371, 375, 387, 388
- Disruptive technology, 327, 328
- Distributed denial-of-service attacks, 366, 367
- Diversity. *See* Innovator diversity; User heterogeneity
- Doerr, John, 332
- DSL. *See* Digital Subscriber Line
- eBay
 - early history and funding of, 206, 207, 210, 302, 304, 311, 312, 320, 328, 333, 334
 - economies of scale, 143
 - and incumbents, 207, 320
 - uncertainty surrounding, 302, 304,
- Economic constraints, 24–28, 31, 32. *See also* Constraints
- Economic growth
 - and adoption of applications, 345, 363, 364
 - and free revealing, 345
 - and Internet, 357–359, 363, 364, 386
 - and user-driven innovation, 345
- Economic system
 - and architecture, 28, 32
 - and constraints, 24–28
 - evolution of, 24, 26, 28
 - terminology of, 24
- Economic system for development and production of technical system
 - impact of architecture on (*see* Architecture; *specific design principles*)
 - impact of on architecture (*see* Cost of change; Design teams; Strategic design)
 - terminology of, 29
- Economic system for use of technical system
 - impact of architecture on, 28, 361–365
 - impact of on architecture (*see* Deployment context; Strategic design)
 - terminology of, 28

- Economic theory, 23–33. *See also* Behavioral economics; Bounded rationality; Rational-choice theory
- Economies of scale
and discrimination (*see* Monopoly preservation in complementary market exception; Monopoly preservation in primary market exception; More sales at market prices exception; Primary good not essential exception)
and governance, 169, 171
as source of competitive advantage, 196, 319, 322
- Electronic Frontier FoundFation, 261, 267
- E-mail
application structure of, 92, 108–110, 383
cost of switching providers, 262, 263
early history of, 206, 302, 307, 309, 336, 337
requirements of, 97
social value of, 359, 360
and standardization, 87, 88
uncertainty surrounding, 302
- Embeddedness. *See* Relationships among actors
- Encryption, 62–66, 367
- End hosts
control over (*see* Control over end hosts)
generality of, 366, 387, 390, 391
layers implemented on, 57, 58
terminology of, 50, 51, 108, 109, 378, 384, 385
- End-to-end architecture, 287–295. *See also* Broad version
amount of innovation in, 352–356
and economic environment for innovation, 289–295
terminology of, 287
welfare implications of (*see* Social value)
- End-to-end arguments. *See also* Broad version; Narrow version
comparison of two versions, 58, 59, 75–81
in current Internet, 371, 372, 383–392
history of, 58, 59, 90, 110–112
horizontal dimension of, 57–60, 75, 76
and original Internet architecture, 90–103, 110–112, 379–381
rationale for distinguishing between two versions, 58, 59, 75–81
vertical dimension of, 57–60, 75, 76
- End-to-end execution of functionality
examples of, 56, 57, 62–67, 90–101
terminology of, 63
transport-layer protocols and, 91, 92, 384
- End-to-end function. *See also* Narrow version
conditions creating, 62–65
encryption as, 62–66
error control as, 62–67
other examples of, 65
terminology of, 60, 61, 377
- Enterprise network. *See* Deployment context
- Error control
above transport layer, 93, 380, 384
in ARPANET Host-to-Host Protocol (AHHP), 91
and broad version, 94–103, 380
on dial-up modem, 66, 67
at Internet layer, 85, 90, 91, 94–105, 139–142
in IP, 85, 90, 91, 94–103
in ITCP, 90, 91, 94–101, 105
and narrow version, 62–67, 90–96, 380, 384
in TCP, 87, 91–101, 380
on wireless access link, 66, 67
in X.25 protocol suite, 91, 105
- Evolutionary theory of innovation, 299–301

- Evolvability of networks based on
 broad version, 69–71, 77, 78, 104–107, 138–148, 152, 155, 204, 205, 210, 211, 366–370, 382, 383. *See also* Broad version; General functionality
- Excite, 304, 321, 328
- Exclusion. *See* Discrimination
- Exclusive content, 263
- Exercise price, 122
- Externalities
 associated with congestion, 363
 between different applications, 356, 357
 between platform and applications, 199, 356, 357
 associated with productive Internet uses, 360–363, 373
- Facebook, 206, 210, 236, 328
- Facilities-based competition, 219, 256, 264
- Fair-use doctrine, 26
- Fake, Caterina, 207, 305, 306, 347
- Fanning, Shawn, 306, 320
- Fate sharing, 101, 102, 105, 106, 366, 382
- Federal Communications Commission, 241, 250, 251, 266, 391, 392
- File sharing, 27. *See also* Peer-to-peer file sharing
- File transfer application, 93, 97
- Financial options, 122, 123
- Firefox, 345
- FirePower, 306
- Firewalls
 and applications, 385, 386
 and broad version, 72, 73, 385
 in current Internet, 1, 2, 286, 372, 385
 and innovation, 78, 315, 368, 385, 386
 and layering, 73, 385
- Firm size
 bias due to, 312, 325, 326
 limits of, 313, 346
- Fixed cost, 233, 234, 239. *See also* Economies of scale
- Flat-rate pricing, 243, 265, 363. *See also* Pricing of Internet services
- Fletcher, Mark, 335, 341
- Flickr
 early history and funding of, 207, 210, 304–306, 314, 347, 351
 and real-options approach to uncertainty, 146
 uncertainty surrounding, 304–306
- Forced access. *See* Open access
- Four11, 320
- Free revealing, 341–345
- Friendster, 210, 328
- Friis, Janus, 216, 306, 307, 315
- Functional requirements, 3, 21, 22, 32
- Funding of innovation. *See* Angel investors; Corporate investors; Innovators; Venture capital
- Future Internet architecture
 and broad version, 103, 104, 355, 356, 368–371, 388, 389
 coexistence of several architectures in, 162, 163
 facilitating evolution of Internet layer in, 161–163
 and narrow version, 384
- Game Neverending, 305
- Gates, Bill, 321
- General functionality, 67, 68, 76. *See also* Application-specific functionality; Broad version
 best-effort service as, 142
 connectionless data transfer as, 97–99, 380
 unreliable data transfer as, 97–103, 141, 380
- General-purpose technology
 and economic growth, 357–359
 Internet as, 359

- Gmail, 263, 350
- Gnutella, 70, 265, 266
- Google
- advertising and, 232, 238, 253, 314–318
 - early funding of, 207, 209, 210, 321, 322, 328, 332, 333, 334
 - early history of, 207, 307, 310, 314–318, 321, 322, 332
 - and economies of scale, 143
 - embeddedness and, 212, 213
 - and evolution of search-engine business model, 314–318
 - and incumbents, 207, 321, 322
 - and mobile Internet, 389–392
 - and real-options approach to uncertainty, 146
- GoTo, 316–318
- Governance, 168–171, 178–180, 185, 190, 192. *See also* Coordination cost; Transaction cost
- Gross, Bill, 316–318
- Harvard Law School, 335, 336
- Harvard University, 206, 212
- Heart, Frank, 302
- Heinemeier Hansson, David, 208, 335, 341
- Heterogeneity. *See* Innovator diversity; User heterogeneity
- Hidden information
- and cost of innovation, 120
 - dependencies of (*see* Interdependencies)
 - in layered architectures, 46–48
 - in modular architectures, 39, 40
 - of protocols, 53
 - terminology of, 39
- Hidden module
- applications as, 145, 146
 - likelihood of innovation on, 136, 137
 - terminology of, 133
- Hold-up, 188. *See also* Transaction cost
- Hop-by-hop execution of functionality
- examples of, 56, 57, 62–67, 90–92, 95, 96, 102, 103, 384
 - terminology of, 63
 - transport-layer protocols and, 92, 384
- Hotmail, 263, 304, 306, 320, 328
- Hourglass metaphor, 89
- Hourihan, Meg, 207, 209, 347, 348
- HTTP. *See* Hypertext Transfer Protocol
- Hybrid fiber coaxial (HFC), 70. *See also* Cable networks
- Hyperbolic discounting, 374
- Hypertext Transfer Protocol (HTTP), 88
- evolution of, 146, 156, 157, 213, 214
 - firewall and NAT traversal of, 306, 386
 - rate of change of, 146
- Hypertext Transfer Protocol Daemon (HTTPD), 321, 338
- IBM, 198
- ICQ, 207–210, 320, 328
- iMac, 45, 120, 121
- Implementation, 20
- IMS. *See* IP Multimedia Subsystem
- Incentive intensity, 171, 189, 196
- Independence, 176. *See also* Coordination cost; Interdependencies
- Independent innovation. *See* Innovation without permission; Innovator diversity
- Indirect network effects, 229. *See also* Network effects
- Information hiding, 38–40. *See also* Hidden information; Visible information
- Innovation
- and architecture, 28–33
 - control over (*see* Control over application deployment; Control over application development; Innovation without permission)
 - and deployment (*see* Deployment context; Deployment requirements)

- by different types of innovators (*see* Innovator diversity; Innovators)
- funding of (*see* Angel investors; Corporate investors; Innovators; Venture capital)
- impact of design principles on (*see specific design principles*)
- at layers of Internet architecture (*see specific layers*)
- terminology of, 28, 115
- theories of (*see* Evolutionary theory of innovation; Neoclassical theory of innovation; Neo- institutional theory of innovation)
- third-party (*see* Innovation without permission; Innovator diversity)
- uncertainty and (*see* Uncertainty)
- Innovation without permission
 - in Internet architecture, 204, 211, 293
 - in modular architectures, 184, 185, 194, 195
 - relevance of, 345–348
- Innovator diversity
 - in modular and integrated architectures, 116, 118, 197–200
 - in network architectures, 203–214, 292, 293, 313, 314
 - relevance of, 30, 116, 271, 272, 298–345, 349
- Innovators
 - established firms as, 116, 271, 272, 311, 314, 319–330
 - funding of, 204–213, 312–314
 - new entrants as, 116, 271, 272, 311–314, 319–330
 - non-commercial, 204–207, 211–214, 289, 292, 293, 313, 365
 - self-funded, 116, 204–213, 311–314, 329, 330
 - users as, 205, 206, 312, 313, 334, 345
 - venture-capital-backed, 198, 207–211, 214, 313, 328–334
- Instant messaging
 - direct network effects in, 229, 231
 - early history of (*see* ICQ)
 - social value of, 359, 360
- Instant Messenger. *See* AOL Instant Messenger
- Integrated architecture. *See* Integrated design
- Integrated design, 44
 - and ability to innovate, 197
 - benefits of, 44, 45
 - cost of innovation in, 119–121
 - costs of, 45
 - and independent component design, 45
 - and industry structure, 196
 - and organization of detailed design, 180, 191, 192, 195
 - and organization of innovation, 180, 181, 192, 197
 - and risk taking with respect to uncertainty, 127
 - underlying tradeoff in, 46
- Intel, 199, 200, 333
- Intellectual-property law
 - and ability to innovate, 197
 - as constraint on innovators, 31, 197
 - and contracting for innovation, 194, 346
 - and free revealing, 342, 343
- Internet and, 26, 27, 362
 - and likelihood of discrimination, 230
 - rationale behind, 342
 - and Schumpeterian competition, 249, 250
- Intelligent networks, 108, 390
- Interconnection fees, 265
- Interdependencies. *See also* Independence; Reciprocal interdependence; Sequential interdependence
 - and cost of system adaptation, 119–121, 131

- Interdependencies (cont.)
 design interdependencies and task interdependencies, 167, 168
 and economic relationships among component makers, 215, 216
 in integrated architectures, 44, 45, 180, 181
 in layered architectures, 46, 47
 in layered network architectures, 51–56
 in modular architectures, 38–44, 181–185
 in original Internet architecture, 88–90, 145, 148–155, 383
 source of, 44, 45
- Interfaces
 in integrated architectures, 44
 in modular architectures, 38–44
 in operating systems (*see* Application programming interfaces)
 of protocols, 52–56
- Intermediaries. *See* Application-level intermediaries
- Internalizing complementary efficiencies argument, 222–225, 230. *See also* Discrimination
- Internet
 economic potential of, 357–364
 political, cultural, and social potential of, 359–365
 productive uses of, 360, 361, 364, 373
 as stupid network, 108
 terminology of, 84
- Internet architecture. *See* Current Internet architecture; Future Internet architecture; Original Internet architecture
- Internet Engineering Task Force (IETF), 2, 88, 146, 153, 154, 157, 202
- Internet Explorer, 321, 345
- Internet Freedom and Nondiscrimination Act of 2006, 221
- Internet Freedom Preservation Act, 221
- Internet layer, 84–86. *See also* Internet Protocol
 cost of innovation at, 151–153, 383
 dependencies on, 88–90, 151, 153, 383
 deployment requirements for protocols at, 157
 error control at, 85, 90, 91, 94–105, 139–142
 evolution of, 151–163, 383
 as portability layer, 89–96, 101, 148–151, 203, 379
 rate of change of protocols at, 151–154, 157–163, 383
 and transport layer, 91, 94–102, 380
- Internet Non-Discrimination Act of 2006, 220, 221
- Internet Policy Statement, 266
- Internet Protocol (IP), 85, 86. *See also* IPv4; IPv5; IPv6
 availability of interface specifications for, 202
 dependencies on, 151, 153, 383
 and end-to-end arguments, 90–103, 380
 error control in, 85, 90, 91, 94–103
 evolution of, 151–154, 157–163, 383
 and fate sharing, 101
 fragmentation and reassembly, 49
 generality of, 96–103, 141, 142
 and portability layer, 89, 90, 101, 149, 379
 and quality of service, 142
 service interface of, 101, 149, 153, 202
 and TCP, 91, 94–102
- Internet telephony
 application architectures for, 314, 315
 cost of developing, 139–143
 cost of operating, 143, 315
 discrimination against, 240–243, 254, 255, 258–260, 312
 as disruptive technology, 327
 and quality of service, 142

- requirements of, 70, 97, 99, 102, 142
- social value of, 359, 360
- Internetwork Transmission Control Protocol (ITCP), 90, 91, 94–101, 105
- Intertemporal bias, 78, 374, 375
- Intuit, 332
- Investors. *See* Angel investors; Corporate investors; Venture capital
- IP. *See* Internet Protocol
- IP address, 85, 153, 154, 157
- IP datagram, 85
- iPhone, 350, 390C.3
- IP multicast
 - deployment of, 157–163
 - standardization of, 153
- IP Multimedia Subsystem (IMS)
 - deployment of, 372
 - as fully controllable architecture, 372
 - as strategic design, 389, 390, 392
- IP protocol field, 75
- IPv4, 153, 385
- IPv5, 154
- IPv6, 153, 154, 157, 158, 161–163
- Isenberg, David, 107, 108
- ITCP. *See* Internetwork Transmission Control Protocol

- Java, 321, 332
- JavaSoft, 306

- Kazaa, 307, 314, 315
- Kleiner Perkins Caulfield and Byers, 210, 316, 332
- Kraus, Joe, 209, 304

- Last-mile networks, 264, 265. *See also* Cable networks; Digital Subscriber Line networks
- Layer bridging. *See* Relaxed layering
- Layering, 46, 47. *See also* Portability layer; Relaxed layering
 - benefits of, 48
 - costs of, 48, 49
 - and deep packet inspection (*see* Deep packet inspection)
 - and end-to-end arguments, 104, 105
 - and firewalls (*see* Firewalls)
 - and middleboxes (*see* Middleboxes)
 - and network-address translators (*see* Network-address translators)
 - in network architectures, 50–57
 - original Internet architecture and, 88–90
 - and packet filters, 74, 75
 - “pure” version of, 47
- Layers, in Internet architecture, 84–88. *See also* Application layer; Internet layer; Link layer; Transport layer
- Laws. *See* Legal constraints
- Lead users, 340
- Legal constraints, 24–28, 31, 32. *See also* Antitrust law; Intellectual property law; Network neutrality
- Lessig, Lawrence, 19
- Leveraging. *See* Discrimination
- Levin, Max, 304
- Link layer, 84, 85, 89. *See also* Relaxed layering
 - cost of innovation at, 148–151
 - dependencies on, 88–90, 148, 149
 - deployment requirements for protocols at, 151
 - evolution of, 148–151
 - rate of change of protocols at, 149, 151
 - terminology of, 84, 85
- Linux, 144, 345
- Load coils, 69, 70
- Long tail. *See* Niche markets
- Lott, Trent, 361
- Ludicorp, 207, 304, 306, 347

- Macromedia, 332
- Madison River Communications, 240–242
- Mailroom layer, 53–57, 87, 155

- MailWrangler, 350
- Make-or-buy decisions, 168–171. *See also* Coordination cost; Transaction cost
- MapQuest, 231
- Marginal cost. *See* Economies of scale
- Market failure, 375
- Market for ideas. *See* Contracting for innovation
- Market structure
and decision to acquire a real option, 133, 136, 137
and incentive to discriminate, 218, 219, 255–266, 391, 392
- Maximum order statistics, 132
- Maximum transmission unit (MTU), 49
- McCool, Robert, 338
- Measurement, 189, 190. *See also* Transaction cost
- Media Player, 88, 230, 231
- Merton, Robert, 123
- Microsoft
early funding of, 333
and search-engine advertising, 237, 238, 253, 317
and World Wide Web, 320, 321, 339
- Microsoft Network (MSN), 321, 322
- Middleboxes. *See* Deep packet inspection; Firewalls; Network-address translators; Packet filters
and broad version, 72, 73, 385
in current Internet, 385
innovation and (*see specific middleboxes*)
and layering, 73–75, 385
- Mirroring hypothesis, 182, 183, 195
- Mobile Internet, 259, 260, 389–392
- Mobile IP, 153, 154, 157, 158, 161–163
- Modifiability. *See also* Option value; Real options
and architecture, 22
and broad version (*see* Evolvability of networks based on broad version)
and coupling, 44
and integrated architectures, 119–121
and layering, 48, 56
and modularity, 40, 43, 44
terminology of, 22
- Modular cluster, 196, 197, 203, 204
- Modularity, 38–40
and ability to innovate, 184, 185, 194–200
benefits of, 39–42
and control over module-level innovation, 184, 185, 194, 195, 198, 345–348
cost of innovation in, 120, 121
costs of, 40, 43
in design, 39
and distributed ownership of architecture, 199, 200
and independent component design, 38–40, 181–185, 192–195, 345–348
and industry structure, 196, 197
and innovation without permission, 184, 185, 194, 195, 198, 345–348
and mirroring hypothesis, 182, 183, 195
in network architectures, 50–57
and organization of detailed design, 181–183, 192–195
and organization of innovation, 183–185, 194, 195, 323, 324
and organization of production, 172
and original Internet architecture, 37, 379
in production, 39
and risk taking with respect to uncertainty, 127
underlying tradeoff in, 43, 44
in use, 39
- Module-level tests, 182, 193
- Modules, 38
- Monopolist's complementary product
source of outside revenue exception, 240–243, 254, 255, 258

- Monopoly
 - extension of, 225–242
 - preservation of, 242–251
 - terminating, 278, 279
- Monopoly preservation in
 - complementary market exception, 249–253, 258, 259
- Monopoly preservation in primary market exception, 244–249
- More outside revenue exception, 253–255, 258
- More sales at market prices exception, 252, 253, 258
- Mosaic browser, 307–309, 320, 321
- Mosaic Communications, 308, 309, 320
- Moskovitz, Dustin, 206
- MoveOn.org, 267
- Mozilla, 338
- Multicast. *See* Application-layer multicast; IP multicast
- Multiple access. *See* Open access
- MySpace, 210, 328
- MySQL, 144

- Napster, 306, 210, 320, 328
- NARAL Pro-Choice, 267, 269
- Narrow version of end-to-end arguments, 60–67, 377, 388
 - and additional lower-layer implementation, 66, 67, 77
 - benefits of, 65, 66
 - and broad version, 75–81
 - costs of, 93
 - in current Internet, 383–385
 - and distributed applications with application-level intermediaries, 92, 383, 384
 - economic effect of, 112
 - and encryption, 62–66
 - and error control (*see* Error control)
 - and fate sharing, 105, 106
 - history of, 58, 59, 90, 110–112
 - horizontal dimension of, 57–60, 75, 76
 - and original Internet architecture, 90–96, 103, 110–112, 380
 - and network neutrality, 72, 73, 80, 81
 - rationale behind, 79, 378, 379
 - and technical constraints on application design, 65, 66, 93, 103, 380
 - vertical dimension of, 57–60, 75, 76
- NASDAQ crash, 209, 317
- NAT. *See* Network-address translators
- National Center for Supercomputing Applications (NCSA), 307–309, 338
- Neoclassical theory of innovation, 298–302
- Neo-institutional theory of innovation, 299–301
- Net Option Value. *See* Option value
- Netscape, 307–309, 320, 321, 328, 332, 333, 338, 339
- Network-address translators (NATs)
 - and broad version, 72, 73, 385
 - in current Internet, 1, 286, 372, 385
 - and innovation, 78, 315, 368, 385, 386
 - and layering, 73, 385
- Network architecture(s)
 - and economic systems (*see* Economic system for development and production of technical system; Economic system for use of technical system)
 - design principles for (*see* Broad version; Layering; Modularity; Narrow version; Relaxed layering)
 - firm strategies and (*see* Competitive strategies; Strategic choices of system architects; Strategic design)
 - and innovation (*see specific design principles; specific types of network architectures*)
 - network providers' interests in, 371–375

- Network architecture(s) (cont.)
- private interests of network providers in, 371–375
 - public interest in, 355–371, 387, 388, 392
 - social benefits of, 355, 365
 - social costs of, 365–375
 - specific (*see* Current Internet architecture; Future Internet architecture; Original Internet architecture; X.25 protocol suite)
 - types of (*see* Controllable architecture; Core-centered architecture; End-to-end architecture; Opaqueness)
 - social value of (*see* Social value)
- Network connections. *See* Relationships among actors
- Network, core of. *See* Core of network
- Network effects
- competition in markets subject to, 228, 230
 - direct, 229
 - and discrimination (*see* Complementary product source of outside revenue exception; Monopoly preservation in complementary market exception; Monopoly preservation in primary market exception; More outside revenue exception; More sales at market prices exception; Primary good not essential exception)
 - indirect, 229
 - in search engines, 237, 238, 253
 - terminology of, 229
- Network management. *See also* Quality of Service
- and broad version, 74, 287
 - discriminatory, 260, 261, 264–266, 350, 351, 363, 365
 - regulation of, 2, 266
- Network neutrality. *See also* Discrimination
- and access charges, 220, 221, 278–280
 - alternatives to, 218, 219, 256, 264, 273, 274
 - and antitrust law, 218, 219, 273
 - and broad version, 72–74, 80, 81, 379
 - and competition in Internet-services market, 218, 219, 255–266
 - and disclosure requirements, 264
 - and narrow version, 72, 73, 80, 81
 - and network management, 266
 - overview of, 219–221, 273, 274
 - and quality of service, 220, 221
 - welfare implications of (*see* Discrimination)
 - wireless, 391, 392
- Network opacity. *See* Opaqueness
- Network optimization. *See* Asymmetric bandwidth; Firewalls; Load coils; Network-address translators
- Network protocols. *See* Protocols
- NeXT, 303
- Niche markets
- cost of innovation and, 144, 145
 - deployment requirements and, 152, 155, 294, 295
 - innovator diversity and, 312, 318, 319, 325–328, 340, 352
- Nintendo, 308
- Non-discrimination. *See* Discrimination
- Non-monopolization models, 251–255
- Non-optimization. *See also* Broad version
- costs of, 68, 69, 365, 369, 370
 - and evolvability (*see* Evolvability of networks based on broad version)
 - and socially valuable uses, 364
 - social relevance of, 365, 371, 387, 388
- Normal distribution, 130
- Norms, as constraints, 24–28, 31, 32
- Novell, 332
- Omidyar, Pierre, 206, 302, 304
- One monopoly rent argument. *See also* Discrimination
- exceptions to, 225–251
 - theory of, 222–225

- Opaqueness. *See also* Asymmetric bandwidth; Core-centered architecture; Deep packet inspection; Firewalls; Network-address translators and amount of innovation, 286, 352, 353, 385, 386 and applications, 70, 264, 265, 385, 386 in current Internet, 286, 372 terminology of, 286 welfare implications of (*see* Social value)
- Open access, 2, 219, 222, 256, 264
- OpenNet Initiative, 268
- Open-source software and cost of application development, 144, 336, 337 and free revealing, 341–345 motivations of developers of, 205, 337 social value of, 360 and user-driven innovation, 336, 337
- Open standards, 185, 193, 199, 202, 213, 214
- Operating system. *See also* Application programming interfaces and Internet-layer and transport-layer protocols, 202, 386 as portability layer, 47, 48
- Opportunism, 170, 186–189. *See also* Transaction cost
- Optimization of network. *See* Asymmetric bandwidth; Firewalls; Load coils; Network-address translators
- Options. *See* Financial options; Option value; Portfolio; Real options
- Option value. *See also* Real options and differences among modules, 133–137 of integrated architectures, 126, 127 of modular architectures, 126, 127 of multiple experiments on a module, 128–137 terminology of, 123
- Original Internet architecture, 83–88 and ability to innovate, 203–214 and availability of interface specifications, 202, 203 broad version and, 96–103, 380, 381 and cost of innovation, 137–163 and economic environment for application innovation (*see* End-to-end architecture) and firms' governance choices, 201, 202 layering principle and, 88–90, 379, 381 narrow version and, 90–96, 103, 380 and structure of industries for development and production of Internet components, 203, 204
- Oversubscription, 265
- Overture, 316–318
- Packet filters, 74, 75. *See also* Middleboxes
- Page, Larry, 207, 307, 310, 316–318, 321, 322, 332
- PageRank, 310
- PayPal, 147, 304, 328, 333, 351
- Pay per click, 316, 317
- Pearl Jam, 269
- Peer interface, 52–55
- Peer-to-peer applications and broad version, 110 and network optimization, 70, 264–266, 385, 386
- Peer-to-peer file sharing and discriminatory bandwidth management, 260, 261, 264–266, 350, 351, 363, 365 as distribution mechanism, 143, 342, 365 social value of, 143, 342, 365
- Perception, 311
- Performance and architecture, 22 and broad version, 68, 69, 77, 78, 365, 367–370, 378, 379

- Performance (cont.)
 and integrated design, 45
 and layering, 48, 49
 and modularity, 43, 45
 and narrow version, 66, 67, 77
- Perl, 144, 337, 341
- Permission. *See* Innovation without permission
- Personal computer (PC)
 architecture of, 21, 199, 200
 cost of innovating on, 120, 121
 and layering, 47, 48
 and modularity, 39, 40, 43, 120, 121
- PGPfone, 315
- PHP, 144, 337, 341
- Phyllo, David, 205
- Physical network, 83, 84
- Python, 144
- Platform products, 199, 356, 357
- Podcaster, 350
- Portability layer
 Internet layer as, 89–96, 101, 148–151, 203, 379
 operating system as, 47, 48
 terminology of, 47
- Portfolio, 125–127
- Port numbers, 75
- Postal layer, 53–57, 86
- Postel, Jon, 99
- PostgreSQL, 144
- Price differentiation, 273–275. *See also*
 Pricing of Internet services
- Price discrimination, exclusion to
 facilitate, 243, 275–278
- Pricing of Internet services. *See*
 Application-specific transport prices;
 Flat-rate pricing; Price discrimination;
 Usage-based pricing
- Pricing strategies. *See* Access charges;
 Pricing of Internet services
- Primary good not essential exception,
 226–232, 235, 246, 249, 252, 253,
 254, 258
- Principal-agent theory, 189, 331
- Product development process, 20
- Protocol peer, 51
- Protocols, 51–57
 interfaces of, 52–56
 Internet, 84–88
- Pyra Labs, 207, 209, 347, 348
- Quality of service
 and broad version, 104–107, 142, 366,
 382
 deployment of 157, 158, 161–163
 and network neutrality, 220, 221
 standardization of, 153, 154
- Quicken, 332
- Rate of change
 of applications, 144–147, 155–157,
 351–353
 of hidden modules, 136, 137
 of Internet-layer protocols, 151–154,
 157–163, 383
 of link-layer protocols, 149–151
 of transport-layer protocols, 151–157
 of visible modules, 135, 136
- Rate regulation in primary market
 exception, 226
- Rational-choice theory, 33, 264, 374
- Real options
 and broad version, 144–146
 decision to acquire, 124, 125, 133,
 136, 137
 in integrated architectures, 123–127
 and Internet layer, 151–154, 157
 in modular architectures, 123–137
 price of, 124
 and relaxed layering with portability
 layer, 150, 151
 and transport layer, 151–157
 underlying concepts of, 122–124
- Reciprocal interdependence, 167, 168,
 176, 177–179. *See also* Coordination
 cost

- Reed, David, 5, 58, 68, 69, 99, 110–112
- Regulation, 2–4, 387, 388, 391, 392.
See also Network neutrality
- Reimbursement processing, 53–57
- Relationships among actors
 and ability to innovate, 212, 213, 319, 322, 329
 and functioning of constraints, 31
 and governance, 178, 179
 venture capitalists and, 329
- Relaxed layering, 47, 88–90, 148–163, 203, 379. *See also* Internet layer
- Reliability of networks, 72
- Requirements specification, 20
- Researcher-reimbursement layer, 53–57
- Reverse engineering, 197
- Risk taking
 in architectures based on and deviating from broad version, 147, 148, 294, 295, 352
 and innovator diversity, 326–331, 340
 in modular and integrated architectures, 127
- Roberts, Lawrence, 302
- Rocketmail, 320, 328
- Routers, 50, 51, 85, 86
- Ruby, 144, 337, 341
- Ruby on Rails, 335, 337, 341, 345
- Saltzer, Jerome, 5, 58, 110–112
- Sandvine, 261
- Schachter, Joshua, 205
- Schmidt, Eric, 332
- Schoch, John, 99
- Scholes, Myron, 123
- Schumpeterian competition, 249–251
- Search engines
 advertising and, 232, 235–238, 253, 314–318
 business model of, 314–318, 321, 322
 early history of, 304, 207, 321, 322
 monetization gap and, 237, 238, 253
 network effects in, 237, 238, 253
 uncertainty surrounding, 304, 314–318
- Security, 1, 2, 22, 23, 74, 287, 366–368.
See also Encryption; Firewalls
- Sendmail, 345
- Sequential interdependence, 176, 177.
See also Coordination cost
- Sequoia Capital, 316, 332
- Server-based applications, 143, 144, 207, 336. *See also* Web-based applications
- Servers, 50. *See also* Web servers
- Service interface, 52–55
- Session Initiation Protocol (SIP), 315
- Settlement charges, 241, 242
- Short-term bias, 78, 374, 375
- Silicon Graphics, 308
- Simple Mail Transfer Protocol (SMTP), 88
- Skewness lover, 329
- Skype
 cost of operating, 143, 315
 discrimination against (*see* Internet telephony)
 early history and funding of, 216, 306, 307, 314, 315, 328, 334
 and NAT traversal, 315, 386
 and quality of service, 142
 rate of change of, 88, 146
- Smith, Jack, 304, 306
- Social networking, 359, 360
- Social value
 of application innovation, 356, 360
 of consumptive uses, 360
 of innovation in general, 356
 of innovation in platform systems, 356, 357
 of network architectures, 355–371, 387, 388
 of productive uses, 360, 361
- Software, development of, 20
- Spillovers, 314, 342, 349, 356, 370
- Standard-setting bodies, 185, 200

- Stand-alone market. *See* Primary good not essential exception
- Stanford Linear Accelerator, 303
- Stanford University, 205, 212, 213, 310
- Status quo bias, 264
- Sticky information, 339
- Stock market bias, 374, 375
- Stock market crash, 209, 317
- Strategic choices of system architects, 195–197, 200, 201. *See also* Strategic design; Competitive strategies
- Strategic design
 examples of, 200, 201, 371, 372, 389–392
 theory underlying, 3, 32, 389
- Streaming video, 70, 87, 88, 102
- Stupid network, 107, 108, 366
- Sunk cost, 245, 247
- Sun Microsystems, 321, 332
- Switching costs, 256, 259–264, 350
- Systemic innovation. *See also*
 Autonomous innovation
 in Internet architecture, 204–213, 386
 in modular architectures, 185, 198–200
 terminology of, 183
- System integration, 183
- System integrator, 196, 203, 204
- System qualities. *See also* Complexity; Correctness; Modifiability; Performance; Security
 and architecture, 21–23
 and design principles, 23
- Systems market. *See* Primary good not essential exception
- Tacit information, 173, 174
- TCP. *See* Transmission Control Protocol
- TCP/IP protocol suite, 84–88. *See also* *specific protocols*
- Telecommunications Workers Union, 267, 268
- Telus, 267, 268
- TENEX, 309
- Thau, Robert, 338, 339
- Thiel, Peter, 304
- 37signals, 207, 208, 334, 335, 341, 345
- Thompson, James, 176
- Thurmond, J. Strom, 361
- Times Mirror, 207
- TimeWarner, 250, 251, 258
- Tipping, 228. *See also* Network effects
- Tomlinson, Ray, 302, 307, 309
- Tradeoffs
 in architecture design, 22, 23
 by network providers, 372–375, 388
 social, 368–371, 387, 388
 and underlying design principles, 23
- Transaction cost
 and architecture, 190, 191
 and coordination cost, 169–171
 and governance, 186–190
 and integrated architectures, 191, 192
 and modular architectures, 192–195
 theory of, 185–190
- Transmission Control Protocol (TCP), 87
 availability of interface specifications for, 202
 error control in, 91–101
 evolution of peer interfaces of, 155–157
 evolution of service interfaces of, 151–154
 and IP, 91, 94–102
 operating end-to-end or hop-by-hop, 92, 93, 384
 standardization of interface specifications for, 202
- Transporting layer, 56, 57, 84, 85
- Transport layer, 84–87. *See also* Transmission Control Protocol; User Datagram Protocol
 availability of interface specifications for protocols at, 202
 cost of innovation at, 151–153
 dependencies on, 88–90, 151, 155

- deployment requirements for
 - protocols at, 156, 385, 386
- error control at (*see* Error control, at transport layer)
- evolution of, 151–157, 386
- and Internet layer, 91, 94–102, 380
- as lowest layer implementing
 - application-specific functionality, 86, 87, 98, 380
- new protocols at, 153, 386
- protocols operating end-to-end or hop-by-hop, 91, 92, 384
- rate of change of protocols at, 151–157
- Transport price. *See* Application-specific transport prices
- Twitter, 345
- Two-sided markets, 278–280
- Types of innovators. *See* Innovator diversity; Innovators

- UDP. *See* User Datagram Protocol
- Uncertainty. *See also* Financial options; Real options
 - and bounded rationality, 187, 300
 - business, 124, 125
 - and complexity, 124
 - and coordination cost, 177, 178, 347, 348
 - in current Internet, 356
 - and established firms, 326–330
 - in Internet history, 300–311
 - and intertemporal bias, 78, 374
 - and likelihood of innovation on a module, 135–137
 - market, 124, 125
 - modeling, 130
 - resolution of, 124, 125, 147, 148, 347, 348
 - risk taking with respect to (*see* Risk taking)
 - and simultaneous entry into two markets, 245, 248
 - technical, 124
 - and transaction cost, 186–188, 347, 348
 - and user choice among new applications, 152–155, 294, 295, 351, 370, 371
 - and user-driven innovation, 340
 - value of experimentation under, 118, 128, 145, 146, 298–301, 314, 315
 - value of innovator diversity under, 212, 298–311, 314, 315, 326–331, 340, 349, 351
 - value of innovator independence under, 198, 345–348
 - and venture capital, 329–331
- Universal Serial Bus (USB) standard, 43
- University of Illinois, 307–309, 338
- Usage-based pricing, 243. *See also* Pricing of Internet services
- User choice. *See also* Control over end hosts
 - and network use, 72, 293, 294, 362–365, 371
 - among new applications, 72, 293–295, 349–351, 365, 366, 370, 371, 382, 387
- User control. *See* User choice
- User Datagram Protocol (UDP), 87, 98
 - availability of interface specifications for, 202
 - evolution of, 151–157
 - history of, 98, 99
- User heterogeneity. *See also* Niche markets
 - in current Internet, 355, 356
 - and value of innovator diversity, 300, 318, 319, 340, 349–352
 - and value of user-driven innovation, 340
- Venture capital
 - availability of, 31, 313
 - corporate, 333, 334
 - inability of innovators to get, 207–210, 312

- Venture capital (cont.)
 - innovators backed by (*see* Innovators)
 - investment model underlying, 208, 329, 349
 - value of, 328–334
- Verizon, 372
- Verizon Wireless, 267, 269
- Vertical boundary choices of firms, 168–171. *See also* Coordination cost; Transaction cost
- Vertical leveraging. *See* Discrimination
- Video conferencing, 70, 97
- Video over Internet, 312, 327. *See also* Streaming video
- Virtual circuit, 97–102
- Virtual private network, 276, 277
- Visible information. *See also* Interdependencies; Systemic innovation; Visible module
 - cost of innovation on, 121, 131–136
 - deployment context as constraint on evolution of, 152–163
 - in layered architectures, 46–48
 - in modular architectures, 38–44
 - of protocols, 52, 53
 - terminology of, 38
- Visible module. *See also* Backward compatibility
 - Internet Protocol as, 151–154
 - likelihood of innovation on, 135, 136
 - terminology of, 133
 - transport-layer protocols as, 151–154
- Voice over Internet Protocol (VoIP). *See* Internet telephony
- Voices for Change, 267, 268
- Vonage, 240–243, 315, 328

- Web-based applications
 - and broad version, 108–110, 384, 385
 - costs of developing and operating, 143, 144, 336
 - financing of, 207
 - rate of change of, 146, 147, 155–157
- Web-based e-mail, 304, 306, 320
- Web browser, 70, 88, 307–309, 320, 321, 338, 341
- Web cache, 383
- Weblogs. *See* Blogs
- Web servers, 70, 88, 338
 - costs of developing and operating, 143, 144
 - early history of, 321, 338–341
 - open-source software stack for, 143, 144
- Welfare. *See* Social value
- Wi-Fi technology, 83, 84, 148–151
- Wikipedia, 360, 361
- Williams, Evan, 207, 209, 347, 348
- Windows. *See* Application programming interfaces; Operating system
- Windows 95, 321
- Windows NT, 321
- World Wide Web, 70, 87, 88, 338
 - as distributed application with application-level intermediaries, 92, 383
 - early history of, 302, 303, 307–310, 320, 321, 337–341
 - social value of, 360
- World Wide Web Consortium, 88, 202

- X.25 protocol suite, 91, 105

- Yahoo
 - and advertising, 236, 239
 - early history and funding of, 205, 212, 213, 317, 318, 322, 328, 333
 - and search-engine advertising, 237, 238, 253, 317
- Yan, Jerry, 205

- Zennström, Niklas, 216, 306, 307, 315
- Zittrain, Jonathan, 336, 366, 367
- Zuckerberg, Mark, 206, 212, 213