WIRELESS LAN 802.11 IMPLEMENTATION IN GSM BASED MOBILE NETWORKS

By

CAPT. OMER KHURSHED
NC RAABIA IRFAN
PC KHAWAJA KASHIF NAVEED

Submitted to the Faculty of Computer Science

National University of Science and Technology, Rawalpindi

in partial fulfillment for the requirement of a

B.E Degree in Computer Science

April 2006

Abstract

The use of GSM mobile networks for call routing is customary, however mind says why not to implement wireless protocol in GSM based mobile networks. The emergence of mobile networks has brought new challenges to traditional network design. Recent advancements in mobile networks have shown great concern to all researchers all over the globe. Why not to provide the user with a cost effective solution to get access to mobile equipments currently in its range specified by wireless LAN. Although a lot of work has been done in the field of wireless, however, no such work as integration of wireless protocol with GSM based mobile networks has ever been done, before. A mobile user must have a free access to its neighboring mobile equipments to make a call rather than directing the call to MSC via BTS.

Researcher's Declaration

We, Capt. Omer Khurshed, PC Kashif Naveed and NC Raabia Irfan, do hereby solemnly declare that the work presented in this report is our own, and has not been presented previously to any other institution.

Signature

_____

(Researcher)

_____

(Researcher)

_____

(Researcher)

Certified:

_____ _____

Internal Supervisor External Supervisor

_____

Examiner

Acknowledgements

First of all we would like to thank Allah Almighty. Whatever, we have achieved; we owe it to Him, in totality.

We are really grateful to our parents, family and well-wishers for their admirable support not only during the course of the project, but also throughout our lives for without them, all this would have never been possible. They have always been behind our every success and are the major stimulating force that facilitates us in achieving what we aspire.

We would like to thank Lt Col Naveed Sarfraz Khattak, for; all the inspiration to initiate every single task has been coming from him. Mr. Shoaib A. Khan has been very helpful in directing us to do the right thing in the right manner. Both of them together, provided us with the opportunity to polish our technical skills and guided us into this area of learning. This project has been brought to shape by their consistent assistance.

There are other people who played their part in various ways. Maj Ather Mohsin Zaidi was always there, whenever we needed him, to solve various problems. Mr. Kaleem helped in providing the computer and lab facilities.

In the end, I would owe this project to a special person who has great influence in my life and has always encouraged me to give out my very best. Thank you very much for being there and helping me through the hard times of my life and whatever I am today, I owe it to you.

List of Figures

List of Tables

Tables **Page No.**

Table of Contents

1

Introduction

1.1 Introduction

This chapter throws light on the different aspects of the mobile communication situations that are currently prevailing, world-wide. A broad brush regarding the extensive work, which has been conducted in the field of VOIP, SIP and advancements in the GSM bases networks, has been given in this chapter.

1.2 General Mobile Communication Scenario

Integration of the SIP based VOIP Soft phone in the GSM based Mobile Network using VOIP to provide mobile user functionality to access users across the globe using internet. Here SIP is being used for

- *setting* up voice-over-IP calls
- *setting* up multimedia conferences
- Event notification (subscribe/notify)
- *T*ext and general messaging
- *S*ignaling transport

There are many applications of the Internet that require the creation and management of a session, where a session is considered an exchange of data between an association of participants. The implementation of these applications is complicated by the practices of participants: users may move between endpoints, they may be addressable by multiple names, and they may communicate in several different media – sometimes simultaneously. Numerous protocols have been authored that carry various forms of real-time multimedia

session data such as voice, video, or text messages. SIP works in concert with these protocols by enabling Internet endpoints (called *user agents*) to discover one another and to agree on a characterization of a session they would like to share. For locating prospective session participants, and for other functions, SIP enables creation of an infrastructure of network hosts (called *proxy servers*) to which user agents can send registrations, invitations to sessions, and other requests. SIP is an agile, general-purpose tool for creating, modifying, and terminating sessions that works independently of underlying transport protocols and without dependency on the type of session that is being established.

Since lots of work has been done on VOIP and there is a lot of advancements in the GSM bases networks. We use VOIP to transmit voice over normal LAN / WAN connected computers. People started using mobile phones due to its easiness, easy to carry, always connectivity etc.

The basic initiative behind setting up the communication through the wireless LAN, rather than the GSM communication, was to facilitate the mobile user with an exceedingly cost effective solution to get an access to any mobile equipment, currently in its range. A mobile user must have an access to its neighboring mobile equipment to make a call, using wireless LAN, rather than directing the call to MSC via BTS in which mobile equipment is lying. The mobile user can make a call to any other neighboring mobile user, totally free of cost.

No previous work has been found in the area of the project.

Whenever a mobile user makes a call to another mobile consumer, his BTS sends the call to MTS. At the MTS, the location of the required user is traced.

After the completion of this phase, the call is directed to the BTS in whose location the called person exists. Then that BTS transmits the call to him. The drawback of this system is that each call is unnecessarily sent to the MTS even if the caller and the called person are in the same BTS range.

The mobile communication scenario, as explained above, is shown

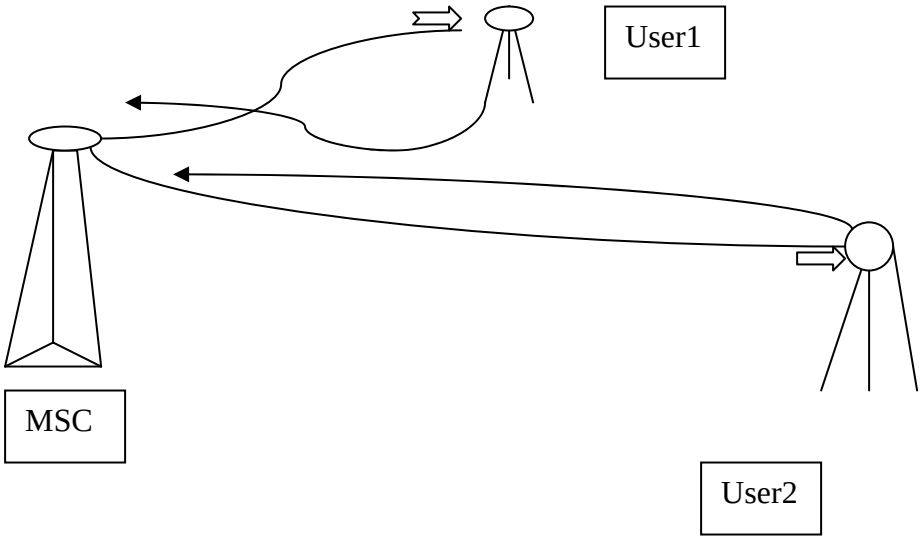

Figure 1.1 - Mobile Communication Scenario

The smaller towers are the BTS; where as the larger tower is the MSC. The call is routed from the BTS of the caller to the MSC and then to the BTS of the callee.

1.3 Summary

Many Internet applications require creation and management of mean of data exchange between an association of participants. SIP is a session creating, modifying, and terminating tool that works independent of the underlying transport protocols and the type of session that is being established. VOIP is used to transmit the voice over normal LAN / WAN connected computers. The basic initiative behind setting up the communication through the wireless LAN,

rather than the GSM communication, was to facilitate the mobile user can make a call to any other neighboring mobile user, totally free of cost.

2

Project Background & Concept

2.1 Introduction

This division of the project basically throws light on the literature reviews that were conducted during the project. It describes the different features and functionalities that are associated with GSM technology and the evolution of the wireless communication and different issues related to it.

2.2 Wireless Communication

Wireless personal communications has captured the attention of the media and with it, the imagination of the public. Hardly a week goes by without one seeing an article on the subject of advance communication subject, appearing in a newspaper or magazine. Countless marketing surveys continue to project enormous demand, often projecting that at least half of the households, or half of the people, want wireless personal communications. It is clear that wireless personal communications is, by any measure, the fastest growing segment of telecommunications. The explosive growth of wireless personal communications has continued unabated worldwide. Cellular and high-tier PCS pocket phones, pagers, and cordless telephones have become very common and have become an expected part of everyday life. In addition to the things noted above, the topic of wireless loops [1], [2], [3] has also become popular in the widespread discussions of wireless communications. As discussed in [2], this topic includes several fixed wireless applications as well as the low-tier PCS application that was discussed originally under the wireless loop designation [1], [3]. Thus, it appears that almost everyone wants wireless personal communications, but *what*

is it? There are many different ways to segment the complex topic into different communications applications, modes, functions, extent of coverage, or mobility [4], [2], [5]. Among the different changes that are occurring in our communications paradigm, perhaps the major constituent is the change from wired fixed place-to-place communications to wireless mobile person-to-person communications. Within this major change are also many other changes, e.g., an increase in the significance of data and message communications, a perception of possible changes in video applications, and changes in the regulatory and political climates. The fixed wireless loop applications do not fit the new mobile communications paradigm. After many years of decline of fixed wireless communications applications, e.g., intercontinental HF radio and later satellites, point-to-point terrestrial microwave radio, and tropo-spheric scatter, it is interesting to see this rebirth of interest in fixed wireless applications. This rebirth is riding on the gigantic "wireless wave" resulting from the rapid public acceptance of mobile wireless communications. Like any major paradigm shift, there will continue to be considerable confusion as many entities attempt to interpret the different expectations associated with the new paradigm.

2.2.1 Mobility and Freedom from Tethers

Perhaps the clearest constituents in all of the wireless personal communications activity are the desire for mobility in communications and the companion desire to be free from tethers, i.e., from physical connections to communications networks. These desires are clear from the very rapid growth of mobile

technologies that provide primarily two-way voice services. For example, cellular mobile has experienced rapid growth.

The often neglected wireless companions to cellular mobiles, i.e., cordless telephones, have experienced even more rapid, but harder to quantify, growth. Similar or even greater growth in these wireless technologies has been experienced throughout the world.

At this point one should again ask; wireless personal communications—*what is it?* The evidence suggests that what is being sought by users, and produced by providers, can be categorized according to the following two major characteristics.

- Communications portability and mobility on many different scales
    - Within a house or building [cordless telephone, (WLANs)]
    - Within a campus, a town, or a city (cellular radio, WLANs, wide area wireless data, radio paging, extended cordless telephone)
    - Throughout a state or region (cellular radio, wide area wireless data, radio paging, satellite based wireless)
    - Throughout a large country or continent (cellular radio, paging, satellite-based wireless)
    - Throughout the world?
- Communications by many different modes for many different applications
    - Two-way voice
    - Data

- Messaging
- Video?

Thus, it is clear why wireless personal communications today is not one technology, not one system, and not one service but encompasses many technologies, systems, and services optimized for different applications and diverse scopes.

2.2.2 Evolution of Technologies, Systems and Services

Technologies and systems [2], [4], [5], [6], [7], [8], [9], [10], that are currently providing, or are proposed to provide, wireless communications services can be grouped into about seven relatively distinct groups. All of the technologies and systems are evolving as technology advances and perceived needs change. Some trends are becoming evident in the evolutions. In this section, different groups and evolutionary trends are explored along with factors that influence the characteristics of members of the groups. The grouping is generally with respect to scale of mobility and communications modes.

2.2.2.1 Cordless Telephones

Cordless telephones [4], [5], [7] generally can be categorized as providing low-mobility applying both to the range and user's speed, low-power, and two-way tether-less voice communications. Cordless telephones using analog technologies appeared in the late 1970s, and have experienced spectacular growth. They have evolved to digital technologies in the forms of CT-2 and DECT.

PHS can be considered either as a quite advanced digital cordless telephone similar to DECT or as somewhat limited low-tier PCS technology.

Cordless telephones were originally aimed at providing economical, tether-less voice communications inside residences, i.e., at using a short wireless link to replace the cord between a telephone base unit and its handset. The most significant considerations in design compromises made for these technologies are to minimize total cost, while maximizing the talk time away from the battery charger. For digital cordless phones intended to be carried away from home in a pocket, e.g., CT-2 or DECT, handset weight and size are also major factors. These considerations drive designs toward minimizing complexity and power used for signal processing and for transmitting. The compromises accepted in cordless telephone design in order to meet the cost, weight, and talk-time objectives

- Few users per megahertz
- Few users per base unit
- Large number of base units per unit area; one or more base units per wireline access line
- Short transmission range

There is no added network complexity since a base unit looks to a telephone network like a wireline telephone. These issues are also discussed in [4], [5].

Major characteristics of the DCT technologies, CT-2 and DECT, can be found in [3], [5]. Even though there are significant differences between these

technologies, e.g., multiple access technology (FDMA or TDMA/FDMA), and channel bit rate. The similarities and their implications are

- 32-kb/s ADPCM Digital Speech Encoding
- Average Transmitter Power _ 10 mW
- Low-Complexity Radio Signal Processing
- Low Transmission Delay, e.g.,<50 ms, and for CT-2<10-ms Round Trip
- Simple Frequency-Shift Modulation and Non-coherent Detection
- Dynamic Channel Allocation
- Time Division Duplex

In short, the main features of the cordless telephones are

- **Strengths**: good circuit quality; long talk time; small lightweight battery; low-cost sets and service.
- **Limitations**: limited range; limited usage regions.
- **Evolutionary trends**: phone points in public places; wireless PBX in business.
- **Remaining limitations and issues**: limited usage regions and coverage holes; limited or no handoff; limited range.

2.2.2.2 Cellular Mobile Radio Systems

Cellular mobile radio systems are being acknowledged as high-tier PCS. These systems generally can be categorized as providing high-mobility (referred to vehicular speeds, and also to widespread coverage [4], [5], [6]), wide-ranging, two-way tetherless voice communications.

Cellular radio integrates wireless access with large-scale networks having sophisticated intelligence to manage mobility of users.

Cellular radio was designed to provide voice service to wide-ranging vehicles on streets and highways [4], [5], [7], [11], and generally uses transmitter power on the order of 100 times that of cordless telephones.

Cellular radio at 800 MHz has evolved to digital radio technologies [4], [5], [7] in the forms of the deployed systems standards

- Global Standard for Mobile (GSM) in Europe
- Japanese or personal digital cellular (JDC or PDC) in Japan
- U.S. TDMA digital cellular known as USDC or IS-54

Table 2.1 – Cellular Mobile Radio System's Feature Comparison

	High-Power Systems				**Low-Power Systems**			
	Digital Cellular (High-Tier PCS)				**Low-Tier PCS**		**Digital Cordless**	
System	IS-54	IS-95 (DS)	GSM	DCS-1800	WACS/PACS	Hand-Phone	DECT	CT-2
Multiple Access	TDMA/ FDMA	CDMA/ FDMA	TDMA/ FDMA	TDMA/ FDMA	TDMA/ FDMA	TDMA/ FDMA	TDMA/ FDMA	FDMA
Frequency band, MHz						1895-1907	1880-1990	864-868
Uplink, MHz	869-894	869-894	935-960	1710-1785				
Downlink, MHz	824-849	824-849	890-915	1805-1880				
Portable transmit	600 mW/ 200 mW	600 mW	1 W/ 125 mW	1 W/ 125 mW	200 mW/ 25 mW	85 mW/ 10 mW	250 mW/ 10 mW	10 mW/ 5 mW

power, max/avg.								
Modulation	π/4 DQPSK	BPSK/ QPSK	GMSK	GMSK	π/4 QPSK	π/4 DQPSK	GFSK	GFSK
RF channel spacing						300	1728	100
Downlink, KHz	30	1250	200	200	300			
Uplink, KHz	30	1250	200	200	300			
Speech coding	VSELF	QCELP	RPE-LTP	RPE-LTP	ADPCM	ADPCM	ADPCM	ADPCM
Speech rate, Kb/s	7.95	8	13	13	32/16/8	32	32	32
Speech channel/ RF channel	3		8	8	8/16/32	4	12	1
Channel Bit rate, Kb/s						384	1152	72
Uplink, Kb/s	48.6		270.833	270.833	384			
Downlinks, Kb/s	48.6		270.833	270.733	384			
Channel coding	½ rate conversation	½ rate forward	½ rate conversation	½ rate conversation	CRC	CRC	CRC	

		1/3rate reverse						
Frame, ms	40	20	4.615	4.615	2.5	5	10	2

The most significant consideration in the design compromises made for the PCS systems was the *high cost of base stations*. Because of the *need to cover highways* running through low-population-density regions between cities, the relatively high transmitter power requirement was retained to provide maximum range from high antenna locations. Compromises that were accepted while maximizing the two just cited parameters are

- High transmitter power consumption
- High user-set complexity
- High signal-processing power consumption
- Low circuit quality
- High network complexity

Some of the characteristics of digital-cellular or high-tier PCS technologies are listed in Table 2.1 for IS-54, IS-95, and GSM at 900 MHz, andDCS-1800, which is GSM at 1800 MHz. These technologies, IS-54/IS-136 are also sometimes known as DAMPS (i.e., Digital AMPS). Additional information can be found in [4], [5], [7]. As with the digital cordless technologies, there are significant differences among these cellular technologies, e.g., modulation type, multiple access technology, and channel bit rate. There are also many similarities

- Low Bit-Rate Speech Coding _13 kb/s with some _8 kb/s
- Some implementations make use of Speech Inactivity

- High Transmission Delay; _200-ms Round Trip
- High-Complexity Signal Processing
- Fixed Channel Allocation
- Frequency Division Duplex (FDD)
- Mobile/Portable Set Power Control

The general outline of the characteristic of the Cellular Mobile Radio Systems are concluded as

- **Strength**: widespread service availability.
- **Limitations**: limited talk time; large heavy batteries; high-cost sets and service; marginal circuit quality; holes in coverage and poor in-building coverage; limited data capabilities; complex technologies.
- **Evolutionary trends**: micro cells to increase capacity and in-building coverage and to reduce battery drain; satellite systems to extend coverage.
- **Remaining limitations and issues**: limited talk time and large battery; marginal circuit quality; complex technologies.

2.2.2.3 Wide Area Wireless Data Systems

Existing wide area data systems generally can be categorized as providing high mobility, wide-ranging, low-data-rate digital data communications to both vehicles and pedestrians [4], [5].

These systems have not experienced the rapid growth that the two-way voice technologies have, even though they have been deployed in many cities for a

few years and have established a base of customers in several countries. Examples of these packet data systems are

Table 2.2 – Comparison between the different Wide Area Wireless Data Systems

	CDPD	**RAM Mobile**	**ARDIS**	**MDN**
Data rate, kb/s	19.2	8(19.2)	4.8(19.2)	76
Modulation	GMSK BT = 0.5	GMSK	GMSK	GMSK
Frequency, MHz	800	900	800	915
Channel spacing, kHz	30	12.5	25	160
Status	1994 services	Full service	Full service	In service
Access means	Used AMPS channels	Slotted Aloha CSMA		FHSS (ISM)
Transmit power, W			40	1

A large micro cell network of small inexpensive base stations has been installed in the lower San Francisco Bay Area by Metricom, and public packet-data service was offered during early 1994.

Most of the small (shoe-box size) base stations are mounted on street light poles. Reliable data rates are about 75 kb/s. The technology is based on slow frequency-hopped spread spectrum in the 902–928 MHz. Transmitter power is 1W maximum, and power control is used to minimize interference and maximize battery life time. However, like all wireless data services, it has failed to grow as rapidly or to attract as many subscribers as was originally expected.

In short, Wide Area Wireless Data Systems, possesses

- **Strength**: digital messages.
- **Limitations**: no voice, limited data rate; high cost.
- **Evolutionary trends**: micro cells to increase capacity and reduce cost; share facilities with voice systems to reduce cost.
- **Remaining limitations and issues**: no voice; limited capacity.

2.2.2.4 High Speed Wireless Local Area Networks

Wireless local-area data networks can be categorized as providing low-mobility high-data-rate data communications within a confined region. Coverage range from a wireless data terminal is short, tens to hundreds of feet. WLANs have been evolving for a few years, but overall the situation is chaotic, with many different products being offered by many different vendors [4, 8]. There is no stable definition of the needs or design objectives for WLANs, with data rates ranging from hundreds of kb/s to more than 10 Mb/s, and with several products providing one or two Mb/s wireless link rates. The best description of the WLAN evolutionary process is: having severe birth pains. An IEEE standards committee, 802.11, has been attempting to put some order into this topic, but their success has been somewhat limited. Users of WLANs are not nearly as numerous as the users of more voice-oriented wireless systems. Optimism remains high that “eventually” they will find the “right” technology to make WLANs “take off”—but the world still waits. Success rate is limited.

The high speed WANs have been seen to

- **Strength**: high data rate.
- **Limitations**: insufficient capacity for voice, limited coverage; no standards; chaos.
- **Evolutionary trends**: hard to discern from all of the churning.

2.2.2.5 Paging/Messaging Systems

Radio paging began many years ago as a one-bit messaging system. The one bit was: someone wants to communicate with you. More generally, paging can be

categorized as one-way messaging over wide areas. It is optimized to take advantage of asymmetry. High transmitter power, and high antennas at the fixed base stations permit low-complexity, very low-power-consumption, pocket paging receivers that provide long usage time from small batteries. This combination provides the large radio-link margins needed to penetrate walls of buildings without burdening the user set battery. Another evolutionary paging route is two-way paging. This is an ambiguous concept, however, since the requirement for two-way communications destroys the asymmetrical link advantage so well exploited by paging. Two-way paging puts a transmitter in the user's set and brings along with it all of the design compromises that must be faced in such a two-way radio system. Thus, the word paging is not appropriate to describe a system that provides two-way communications. Two-way paging is unrealistic.

In short, the paging systems has

- **Strengths**: widespread coverage; long battery life; small lightweight sets and batteries; economical.
- **Limitations**: one-way message only; limited capacity.
- **Evolutionary desire**: two-way messaging and/or voice; capacity.
- **Limitations and issues**: two-way link cannot exploit the advantages of one-way link asymmetry.

2.2.2.6 Satellite Based Mobile Systems

Satellite-based mobile systems are the essence of wide-area coverage, expensive base station systems.

They generally can be categorized as providing two-way (or one-way) limited quality voice and/or very limited data or messaging to very wide-ranging vehicles (or fixed locations). These systems can provide very widespread, often global.

The strength of satellite systems is their ability to provide large regional or global coverage.

A satellite system's weakness is also its large coverage area. It is very difficult to provide from Earth orbit the small coverage cells that are necessary for providing high overall systems capacity from frequency reuse. This fact, coupled with the high cost of the orbital base stations, results in low capacity along with the wide overall coverage but also in expensive service. Thus, satellite systems are not likely to compete favorably with terrestrial systems in populated areas or even along well-traveled highways.

Proposed satellite systems range from

- Low-Earth-orbit systems (LEOS) having tens to hundreds of satellites through
- Intermediate- or medium-height systems (MEOS) to
- Geostationary or geosynchronous orbit systems (GEOS) having fewer than ten satellites

LEOS require more, but less expensive, satellites to cover the Earth, but they can more easily produce smaller coverage areas and, thus, provide higher capacity within a given spectrum allocation. Also, their transmission delay is significantly less, providing higher quality voice links. On the other hand, GEOS require only a few, somewhat more expensive, satellites and are likely to provide

lower capacity within a given spectrum allocation and suffer severe transmission-delay impairment on the order of 0.5 s. Of course, MEOS fall in between these extremes.

2.2.3 Wireless Standards

Wireless communication is a very speedy progressing discipline. Many new facilities are being introduced in this area, every now and then. The subsequent part of the chapter covers the different standards that have been a vital part of this field.

2.2.3.1 802.11 Wireless LANs (Wi-Fi)

802.11, known as **Wi-Fi**, define standards for wireless LANs (WLANs) and were approved in Jul'97. WLANs provide half- Duplex (not simultaneous bidirectional) connections that are shared, not switched. IEEE 802.11a and 802.11b (Standardized in Sept'99) and 802.11g (standardized in mid-2003) define different physical layer standards for WLANs, and the 802.11 standard offers no provisions for interoperability between these physical layers. Microsoft certification applies to both 802.11a and 802.11b. The **Wi-Fi** Alliance [13], previously known as WECA, promotes the standard, tests products for interoperability, and awards the "**Wi-Fi**" mark to those that past. **Wi-Fi** Alliance certified over 500 products by November '02. Security is one of the biggest issues with the wireless LANs. The [12] has ready to lend hand information.

2.2.3.1.1 802.11a

802.11a operates at 5 GHz and provides data rates up to 54 Mbps using OFDM modulation. 802.11a supports a maximum of 24 unique connections per access

point, far more than the three connections supported by 802.11b and 802.11g. Compared to 802.11b, 802.11a offers higher (2X-5X) throughput, more available frequencies, avoiding multi-path echoes, but shorter range (60-100 feet). Actual throughput is often only 1-2 Mbps.

2.2.3.1.2 802.11b

802.11b operates at 2.4 GHz and provides data rates up to 11 Mbps over links of 150-300 feet using DSSS modulation. Actual throughput is often only 8-10 Mbps. 802.11b supports a maximum of three unique connections per access point. 802.11 Planet has a helpful paper comparing 802.11a vs. 802.11b [14].

2.2.3.1.3 802.11e

802.11e describes error correction and bandwidth management to be used in 802.11a and 802.11b. 802.11e includes a mandatory EDCA operating mode and an optional Polled Access mode. EDCA will define eight levels of access priority, a feature that is important for properly handling voice over WLAN. The IEEE is expected to ratify this standard around mid-2004.

2.2.3.1.4 802.11g

802.11g is an extension to 802.11b to provide data rates up to 54 Mbps while operating at 2.4 GHz like 802.11b but using OFDM modulation like 802.11a. Products are expected to have RF interference problems similar to 802.11b. Like 802.11b, 802.11g supports a maximum of three unique connections per access point. The IEEE approved the specification in June ‘03, and the first products claiming compatibility with the draft standard shipped in Jan’03. 802.11 Planet has a helpful tutorial comparing 802.11a with 802.11g [15].

2.2.3.1.5 802.11i

A standard approved in June'04 that provides security enhancements based on WPA, TKIP, and AES. AES is the new U.S. Government data encryption standard and is far more secure than WPA, the previous 802.11 security mechanism. 802.11i incorporates key management and authentication, and may eventually replace WEP and WPA for WLAN security.

2.2.3.1.6 802.11n

802.11n is a standard in development for WLANs having 100 Mbps throughput.

2.2.3.2 Wireless Personal Area Networking (WPAN)

The IEEE Wireless Overview Web site is helpful to explain what is happening in this area [16].

2.2.3.2.1 802.15.1 - Bluetooth

A standard defining wireless networking with a 1 Mbps data rate that operates at 2.4 GHz over a range of up to 10m. Bluetooth is intended for short-range links between computers, PDA, mobile phones, printers, digital cameras, keyboards, and other PC peripherals. The 1 Mbps data rate is a serious limitation that prevents this technology from acting as a USB replacement [17], [18].

2.2.3.2.2 802.15.3 and 802.15.3a

Project group planning high rate WPANs with 10-500 Mbps data rates. UWB is a key technology being considered here.

2.2.3.2.3 802.15.4 - ZigBee

This addresses the low cost and low power needs that remote monitoring, control and sensory network applications have, including the ability to run for years on

standard batteries. These products operate with rates up to 250 Kbps at 2.4 GHz globally, 915 MHz in the Americas, and 868 MHz in Europe [19].

2.2.3.2.4 802.16 - WiMax (Worldwide Interoperability for Microwave Access)

This IEEE standard defines broadband wireless for the metropolitan area to address the “last mile” problem of providing connections to individual homes and offices. The initial version operates in the 10-66 GHz frequency band with LOS towers to fixed locations.

The 802.16a extension does not require LOS transmission and allows use of lower 2-11 GHz frequencies for both fixed and portable applications. 802.16a claims up to a 30-mile range and 75 Mbps data transfer that can support thousands of users. 802.16a provides selectable channel bandwidths from 1.25-20 MHz with up to 16 logical sub-channels.

2.3 GSM Technology

Named after the organization that created the system standards (Group Special Mobile) [20]. GSM uses combined TDMA and FDMA with frequency division duplex for access. Carriers are spaced at 200 kHz and support eight TDMA time slots each. For the uplink the frequency band 890–915 MHz is allocated, whereas the downlink uses the band 935–960 MHz. Sophisticated error-correction coding with varying levels of protection for different outputs of the speech coder is provided. GSM provides slow frequency hopping as a further mechanism to improve the efficiency of the inter-leaver.

Different frequencies are used in neighboring cells to provide orthogonal signaling without the need for tight synchronization of base stations. Furthermore, channel assignment can then be performed in each cell individually. Within a cell, one or more frequencies are shared by users in the time domain.

From an implementation standpoint TDMA systems have the advantage that common radio and signal processing equipment at the base station can be shared by users communicating on the same frequency. A somewhat more subtle advantage of TDMA systems arises from the possibility of monitoring surrounding base stations and frequencies for signal quality to support mobile assisted handovers.

Figure 2.1 – GSM Technology

According to the resolution of the World Radio communication Conference in 1978, the European Telecom Authorities primarily reserved two frequency bands

of twice 25 MHz: 890 MHz to 915 MHz from mobile to the network 935 MHz to 960 MHz from base stations to the mobiles for use by cellular systems

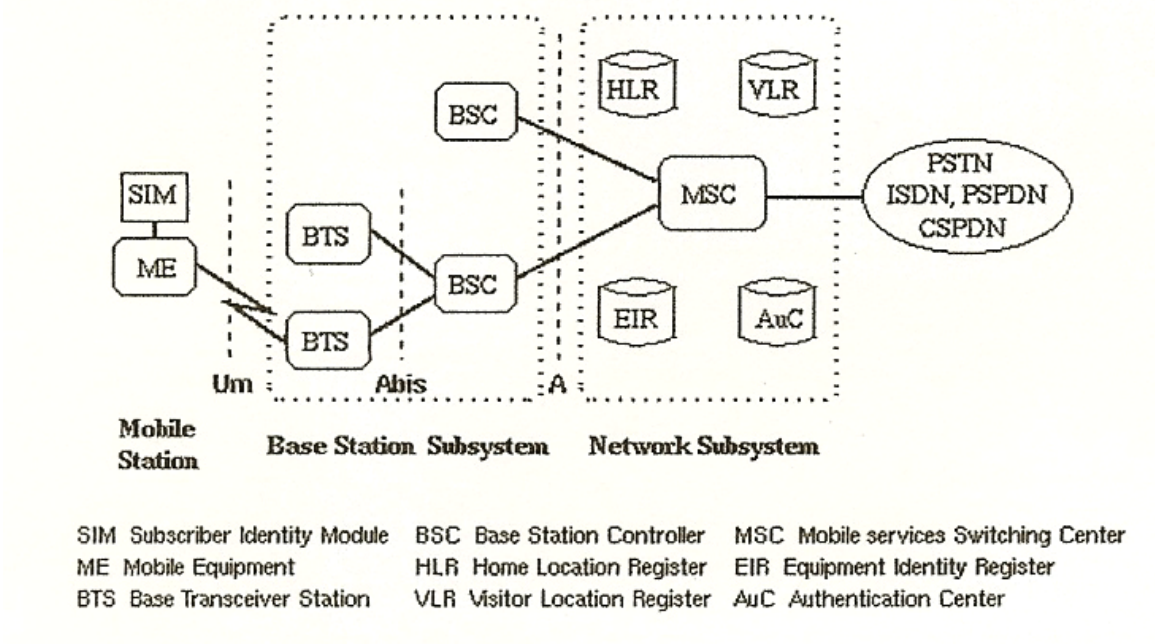

Figure 2.2 – Communication Setup

- **Mobile to Mobile**:
    - Request is sent to the MSC
    - Validated in the VLR
    - Authenticated in the AuC
    - Passed on to PSTN
    - PSTN validates the number
    - Verifies it can be delivered
    - Connects
- **Land to Mobile**:
    - PSTN receives request
    - Sends to home MSC

- Queries VLR
- Incoming call from particular MSISDN
- Responds with IMSI and last known location
- Validates handset is on
- Authenticates receiving SIM
- Instructs handset to ring

- **Mobile to Mobile on the same network**:
  - Call request
  - MSC validates in VLR
  - Authentication
  - MSC advises VLR incoming call with MSISDN
  - VLR responds with IMSI and last known location
  - Validates handset is on
  - Authenticates receiving SIM
  - Network instructs handset to ring

2.4 Wireless LAN Concept

A proliferation of high-performance portable computers combined with end-user need for communication is fueling a dramatic growth in wireless LAN technology. Users expect to have the ability to operate their portable computer globally while remaining connected to communications networks and service providers. Wireless LANs and cellular networks, connected to high-speed networks, are being developed to provide this functionality.

Before delving deeper in to issues relating to the design of wireless LANs, it is instructive to consider some scenarios of user mobility.

- A simple model of user mobility is one where a computer is physically moved while retaining network connectivity at either end. For example, from one room to another as in hospital where computer is a hand-held device displaying patient charts and nurse using the computer moves between wards or floors while accessing patient information.
- Another model situation is where a group of people set up an ad-hoc LAN to share information as shown

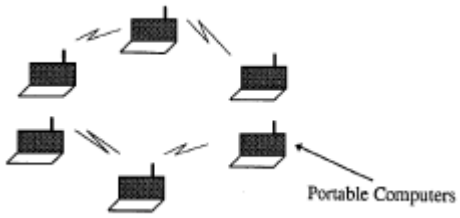

Figure 2.3 – Ad-hoc LAN

- A more complex model is one where several computers in constant communication are in motion and continue to be networked. For example, consider the problem of having robots in space collaborating to retrieve a satellite.

A great deal of research has focused on physical and MAC layer protocols.

2.4.1 Physical Layer Design

Two media are used for transmission over wireless LANs, infrared and radio frequency.

RF LANs are typically implemented in the industrial, scientific, and medical frequency bands 902–928MHz, 2400–2483.5MHz and5725–5850MHz. These frequencies do not require a license allowing the LAN product to be portable, i.e., a LAN can be moved without having to worry about licensing.

IR receiver design is simple and inexpensive, because IR receivers only detect the only the amplitude of the signal and not the frequency or phase. Thus, a minimal of filtering is required to reject interference.

Unfortunately, however, IR shares the electromagnetic spectrum with the sun and incandescent or fluorescent light. These sources of modulated infrared energy reduce the signal-to-noise ratio of IR signals and, if present in extreme intensity, can make the IR LANs inoperable.

There are two approaches to building IR LANs

- The transmitted signal can be focused and aimed. In this case the IR system can be used outdoors and has an area of coverage of a few kilometers.
- The transmitted signal can be bounced off the ceiling or radiated omni-directionally. In either case, the range of the IR source is 10–20 m.

2.4.2 MAC Layer Protocol

MAC protocol design for wireless LANs poses new challenges because of the in-building operating environment for these systems. Unlike wired LANs, wireless LANs operate in strong multi-path fading channels where channel characteristics can change in very short distances resulting in unreliable communication and

unfair channel access due to capture. Another feature of the wireless LAN environment is that carrier sensing takes a long time in comparison to wired LANs; it typically takes between 30 and 50 μs [21], which is a significant portion of the packet transmission time. This results in inefficiencies if the CSMA family of protocols is used without any modifications.

Other differences arise because of the mobility of users in wireless LAN environments. To provide a building with wireless LAN coverage, the region to be covered is divided into cells. Each cell is one wireless LAN, and adjacent cells use different frequencies to minimize interference. Within each cell there is an access point called a MSS or base station that is connected to some wired network. The mobile users are called MH. The MSS performs the functions of channel allocation and providing connectivity to the existing wired networks. The two problems that arise in this type of architecture, which are not present in wired LANs are

- The number of nodes with in a cell changes dynamically as users move between cells. How can the channel access protocol dynamically adapt to such changes efficiently?
- When a user moves between cells, the user has to make its presence known to the other nodes in the cell. How can this be done without using up too much bandwidth? The protocol used to solve this problem is called a handoff protocol and works as the switching station collects signal strength information for each mobile host within each cell. Note that if a mobile host is

near a cell boundary, the MSS node in its current cell as well as in the neighboring cell can hear its transmissions and determine signal strengths. If the mobile host is currently under the coverage of MSSM1 but its signal strength at MSSM2 becomes larger, the switching station initiates a handoff whereby the MH is considered as part of M2's or network.

2.4.2.1 Reservation -TDMA (R-TDMA)

This approach is a combination of TDMA and some contention protocol [22]. The MSS divides the channel into slots (as in TDMA), which are grouped into frames. When a node wants to transmit, it needs to reserve a slot that it can use in every consecutive frame as long as it has data to transmit. When it has completed transmission, other nodes with data to transmit may contend for that free slot. There are four steps to the functioning of this protocol.

- At the end of each frame the MSS transmits a feedback packet that informs nodes of the current reservation of slots.
- During a frame, all nodes wishing to acquire a slot transmit with a probability ρ during a free slot. If a node is successful it is so informed by the next feedback packet. If more than one node transmits during a free slot, there is a collision and the nodes try again during the next frame.
- A node with a reserved slot transmits data during its slot. This is the contention-free transmission.

- The MSS sends acknowledgements for all data packets received correctly.

The R-TDMA protocol exhibits several nice properties. First and foremost, it makes very efficient use of the bandwidth, and average latency is half the frame size. Another big benefit is the ability to implement power conserving measures in the portable computer. Since each node knows when to transmit, it can move into a power-saving mode for a fixed amount of time, thus increasing battery life. This feature is generally not available in CSMA-based protocols. Furthermore, it is easy to implement priorities because of the centralized control of scheduling. One significant drawback of this protocol is that it is expensive to implement [23].

2.4.2.2 Distributed Foundation Wireless MAC (DFWMAC)

The CSMA/CD protocol has been used with great success in the Ethernet. Unfortunately, the same protocol is not very efficient in a wireless domain because of the problems associated with cell interference, the relatively large amount of time taken to sense the channel [24] and the hidden terminal problem [25, 26]. The current proposal is based on a CSMA/collision avoidance (CA) protocol with a four-way handshake.

The basic operation of the protocol is simple. All MH nodes that have packets to transmit compete for the channel by sending RTS messages using non-persistent CSMA. After a station succeeds in transmitting a RTS, the MSS sends CTS to the MH. The MH transmits its data and then receives an ACK. The only possibility of collision that exists is in the RTS phase of the protocol and inefficiencies occur in the protocol, because of the RTS and CTS stages.

Note that unlike R-TDMA it is harder to implement power saving functions. Furthermore, latency is dependent on system load making it harder to implement real-time guarantees. Priorities are also not implemented. On the positive side, the hardware for this protocol is very inexpensive.

2.4.2.3 Randomly Addressed Polling (RAP)

In this scheme, when a MSS is ready to collect uplink packets it transmits a READY message. At this point, all nodes with packets to send attempt to grab the channel as

- Each MH with a packet to transmit generates a random number between 0 and *P*.
- All active MH nodes simultaneously and orthogonally transmit their random numbers. We assume that all of these numbers are received correctly by the MSS.
- Steps a and b are repeated *L* times.
- At the end of *L* stages, the MSS determines a stage (say, *k*) where the total number of distinct random numbers was the largest. The MSS polls each distinct each random number in this stage in increasing order. All nodes that had generated the polled random number transmit packets to the MSS.
- Since more than one node may have generated the same random number, collisions are possible. The MSS sends ACK or NACK after each such transmission. Unsuccessful nodes try again during the next iteration of the protocol.

The protocol is discussed in detail in [21] and a modified protocol called GRAP (for Group RAP) is discussed in [27].

2.4.3 Network Layer Issues

An important goal of wireless LANs is to allow users to move about freely while still maintaining all of their connections. This means that the network must route all packets destined for the mobile user to the MSS of its current cell in a transparent manner. Two issues need to be addressed in this context.

- How can users be addressed?
- How can active connections for these mobile users be maintained?

Ioanidis, Duchamp, and Maguire [28] propose a solution called the IPIP (IP-within-IP) protocol. Here each MH has a unique IP address called its home address. To deliver a packet to a remote MH, the source MSS first broadcasts an ARP request to all other MSS nodes to locate the MH. Eventually some MSS responds. The source MSS then encapsulates each packet from the source MH within another packet containing the IP address of the MSS in whose cell the MH is located. The destination MSS extracts the packet and delivers it to the MH. If the MH has moved away in the interim, the new MSS locates the new location of the MH and performs the same operation. This approach suffers from several problems as discussed in [29].

Specifically, the method is not scaleable to a network spanning areas larger than a campus because of

- IP addresses have a prefix identifying the campus sub-network where the node lives; when the MH moves out of the campus, its IP address no longer represents this information.
- The MSS nodes serve the function of routers in the mobile network and, therefore, have the responsibility of tracking all of the MH nodes globally causing a lot of overhead in terms of message passing and packet forwarding.

2.4.3.1 Alternative View of Mobile Networks

The approaches just described are based on the belief that mobile networks are merely an extension of wired networks. Other authors [30] disagree with this assumption because there are fundamental differences between the mobile domain and the fixed wired network domain. Two examples follow

- The available bandwidth at the wireless link is small; thus, end-to-end packet re-transmission for TCP-like protocols is a bad idea. This leads to the conclusion that transmission within the mobile network must be connection oriented. Such a solution, using VC, is proposed in [31].
- The bandwidth available for MH with open connections changes dynamically since the number of other users present in each cell varies randomly. This is a feature not present in fixed high-speed networks where, once a connection is set up, its bandwidth does not vary much. Since bandwidth changes are an artifact of mobility and are dynamic, it is necessary to deal

with the consequences locally to both, i.e., shield fixed network hosts from the idiosyncrasies of mobility as well as to respond to changing bandwidth quickly.

Some other differences are discussed in [30].

2.4.3.2 A Proposed Architecture

Keeping these issues in mind, a more appropriate architecture has been proposed in Ghai and Singh [31], and Singh [30]. Mobile networks are considered to be different and separate from wired networks.

Within a mobile network is a three-layer hierarchy. At the bottom layer are the MHs. At the next level are the MSS nodes. Finally, several MSS nodes are controlled by a SH node. The SH nodes are responsible for flow control for all MH connections within their domain; they are also responsible for tracking MH nodes and forwarding packets as MH nodes roam. In addition, the SH nodes serve as a *gateway* to the wired networks. Thus, any connection setup from a MH to a fixed host is broken in two, one from the MH to the SH and another from the SH to the fixed host. The MSS nodes in this design are simply connection endpoints for MH nodes. Thus, they are simple devices that implement the MAC protocols. Some of the benefits of this design are as follows

- Because of the large coverage of the SH the MH remains in the domain of one SH much longer. This makes it easy to handle the consequences of dynamic bandwidth changes locally. For instance, when a MH moves into a crowded cell, the bandwidth available to it is reduced. If it had an open FTP connection, the

SH simply buffers undelivered packets, until they can be delivered. There is no need to inform the other endpoint of this connection of the reduced bandwidth.

- When a MH node sets up a connection with a service provider in the fixed network, it negotiates some QOS parameters such as bandwidth, delay bounds, etc. When the MH roams into a crowded cell, these QOS parameters can no longer be met because the available bandwidth is smaller. If the traditional view is adopted, then these QOS parameters will have to be renegotiated each time the bandwidth changes. This is a very expensive proposition because of the large number of control messages that will have to be exchanged. In the approach of Singh [30], the service provider will never know about the bandwidth changes since it deals only with the SH that is accessed via the wired network. The SH bears the responsibility of handling bandwidth changes by either buffering packets until the bandwidth available to the MH increases or it could discard a fraction of real-time packets to ensure delivery of most of the packets within their deadlines. The SH could also instruct the MSS to allocate a larger amount of bandwidth to the MH when the number of buffered packets becomes large. Thus, the service provider in the fixed network is shielded from the mobility of the user.

2.4.3.3 Networking Issues

It is important for the network to provide connection-oriented service in the mobile environment because bandwidth is at a premium in wireless networks, and it is, therefore, inadvisable to have end-to-end retransmission of packets.

Every connection set up with one or more MH nodes as a connection endpoint is routed through the SH nodes and each connection is given a unique VC number. The SH node keeps track of all MH nodes that lie within its domain.

When a packet needs to be delivered to a MH node, the SH first buffers the packet and then sends it to the MSS at the current location of the MH or to the predicted location if the MH is currently between cells. The MSS buffers all of these packets for the MH and transmits them to the MH if it is in its cell. The MSS discards packets after transmission or if the SH asks it to discard the packets. Packets are delivered in the correct order to the MH by having the MH transmit the expected sequence number during the initial handshake. The MH sends ACKs to the SH for packets received. SH discards all the acknowledged packets. When a MH moves from the domain of SH1 into the domain of SH2 while having open connections, SH1 continues to forward packets to SH2 until either the connections are closed or until SH2 sets up its own connections with the other endpoints for each of MH's open connections. Detailed protocol is presented in [31].

The SH nodes are all connected over the fixed network. Therefore, it is necessary to route packets between SH nodes using the protocol provided over the fixed networks. The VIP protocol appears to be best suited to this purpose.

Let us assume that every MH has a globally unique VIP address. The SHs have both a VIP as well as a fixed IP address. When a MH moves into the domain of a SH, the IP address affixed to this MH is the IP address of the SH. This ensures that all packets sent to the MH are routed through the correct SH node. SH keeps a list of all VIP addresses of MH nodes within its domain and a list of open VCs for each MH. It uses this information to route the arriving packets along the appropriate VC to the MH.

2.4.4 Transport Layer Design

The transport layer provides services to higher layers, which include connectionless services like UDP or connection-oriented services like TCP. Recently variations of the TCP protocol have been proposed that work well in the wireless domain. These proposals are based on the traditional view that wireless networks are merely extensions of fixed networks. One such proposal is called I-TCP [23] for indirect TCP. The motivation behind this work stems from the following observation. In TCP the sender times out and begins retransmission after a timeout period of several hundred milliseconds. If the other endpoint of the connection is a MH, it is possible that the MH is connected for a period of several seconds. This results in the TCP sender timing out and transmitting the same data several times over, causing the effective throughput of the connection to degrade rapidly. To alleviate this problem, the implementation of I-TCP separates a TCP connection into two pieces—one from the fixed host to another fixed host that is near the MH and another from this host to the MH. The host closer to the MH is aware of mobility and has a larger timeout period. It serves as a type of

gateway for the TCP connection because it sends ACKs back to the sender before receiving ACKs from the MH. The performance of I-TCP is far superior to traditional TCP for the mobile networks studied.

A problem that is unique to the mobile domain occurs because of the unpredictable movement of MH.

Consider the following example. Say 9 MH nodes have opened 11-kb/s connections in a cell where the available bandwidth is 100 kb/s. Let us say that a tenth mobile host M10, also with an open 11-kb=s connection, wanders in. The total requested bandwidth is now 110 kb=s while the available bandwidth is only 100 kb=s. What is to be done?

One approach would be to deny service to M10. However, this seems an unfair policy. A different approach is to penalize all connections equally so that each connection has 10-kb/s bandwidth allocated. To reduce the bandwidth for each connection from 11 kb/s to 10 kb/s, two approaches may be adopted

- Throttle back the sender for each connection by sending control messages.
- Discard 1-kb=s data for each connection at the SH. This approach is only feasible for applications that are tolerant of data loss.

The first approach encounters a high overhead in terms of control messages and requires the sender to be capable of changing the data rate dynamically. This may not always be possible; for instance, consider a teleconference consisting of several participants where each mobile participant is subject to dynamically

changing bandwidth. In order to implement this approach, the data will have to be compressed at different ratios for each participant, and this compression ratio may have to be changed dynamically as each participant roams. This is clearly an unreasonable solution to the problem. The second approach requires the SH to discard 1-kb/s of data for each connection. The question is, how should this data be discarded? That is, should the 1 kb of discarded data be consecutive or uniformly spread out over the data streams every 1 s? The way in which the data is discarded has an effect on the final perception of the service by the mobile user. If the service is audio, for example, a random uniform loss is preferred to a clustered loss. If the data is compressed video, the problem is even more serious because most random losses will cause the encoded stream to become unreadable resulting in almost a 100% loss of video at the user. A solution to this problem is proposed in Seal and Singh [32], where a new sub layer is added to the transport layer called the LPTSL. This layer determines how data is to be discarded based on special transport layer markers put by application calls at the sender and based on negotiated loss functions that are part of the QOS negotiations between the SH and service provider. LPTSL does not need to know encoding details of the data stream.

This scheme is currently being implemented at the University of South Carolina by the author and his research group.

2.5 Summary

The emergence of mobile networks has brought new challenges to traditional communication network design. Recent advancements in mobile networks have

shown its great interest to all researchers all over the globe. New services are being added to the mobile networks to make its acceptance more by its users.

Wireless personal communications has become the talk of the town. Perhaps the most obvious elements in all of the wireless personal communications activity are the desire for mobility in communications and the companion desire to be free from tethers, i.e., from physical connections for communications networks. These desires are clear from the very rapid growth of mobile technologies that provide primarily two-way voice services. For example, cellular mobile has experienced rapid growth.

GSM technologies and systems are evolving as technology advances and perceived needs change. Some trends are becoming evident in the evolutions. A proliferation of high-performance portable computers combined with end-user need for communication is fueling a dramatic growth in wireless LAN technology. Users expect to have the ability to operate their portable computer globally while remaining connected to communications networks and service providers. Wireless LANs and cellular networks, connected to high-speed networks, are being developed to provide this functionality.

3

Existing Mobile Scenario

3.1 Introduction

The emergence of mobile networks has brought many new challenges to the traditional network design. Recent advancements in mobile networks have captured many researchers, all over the globe, to show their significant attention to this vast and un-explored field. New services are being added to the mobile network communication to make its acceptance more and more by its users. Although, a lot of research, as well as the practical work, has been done in the discipline of wireless. However, no such work, as the integration of the wireless protocol with GSM based mobile networks, has ever been conducted.

We got an opportunity to vast our horizon in context of mobile networks, by taking up this venture. Here, in this project we are not mostly concerned with the wireless protocol in depth, rather we presuppose to get its functionality as our requirement for project.

3.2 VOIP Introduction

VOIP refers to technologies that allow telephone-like voice communications carried within IP data packets, rather than dedicated voice signal lines. Audio is digitized, compressed, and then filled into multiple IP packets. These data packets travel through a packet-switched network such as the Internet and arrive at their destination where they are decompressed using a compatible Codec (audio coder/decoder) and converted back to analogue audio.

3.2.1 How VOIP works?

Effectively, VOIP allows people to conduct normal telephone conversations without using telephone lines.

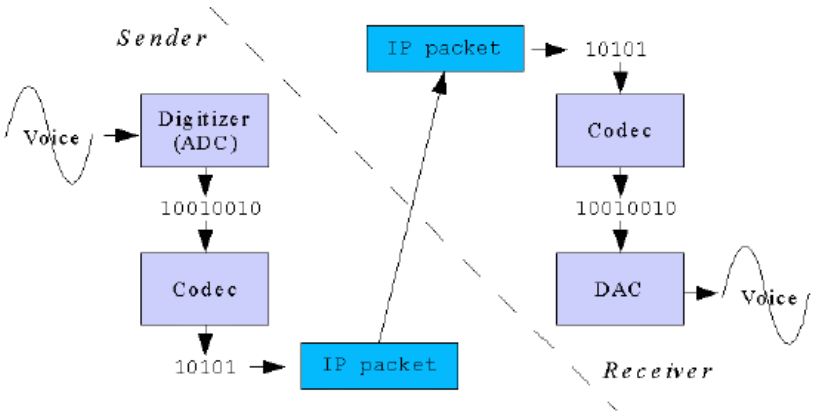

Figure 3.1 – VOIP Architecture

While there are many different implementations of this technology, the fundamental motivations behind VOIP are

- **Low cost:** An international or overseas VOIP call may cost a mere fraction of an equivalent telephone call using long-distance carrier. The costs are reduced to those required to sustain network connectivity at each endpoint, while the data travels freely over global IP networks alongside all the other Internet traffic.
- **Convenience:** While telephone or cellular services require subscription and often have physical limitations with respect to location, VOIP allows service anywhere there is Internet access.
- **Flexibility:** Despite the advances in telephone technology, the user is quite limited by the equipment and features offered. VOIP, on the other hand, allows a myriad of service types and features with the only requirement being adherence to a defined protocol or standard.

3.2.2 Existing VOIP Systems

VOIP systems have existed for quite some time, but have only recently become main stream as audio quality has increased and can now rival that of POTS. This is equally due to newly developed compression algorithms and increased bandwidth availability. Certain properties are vital to the quality and speed of any system. The properties are bandwidth (compression, decompression, etc.), routing, inherent Internet attributes (NAT, firewalls) and security. Our software system uses each of these things.

Previously implemented systems surmounted many key issues central to today's Internet (ie: NAT, bandwidth) in an ad-hoc manner. There are also methods that are common to all VOIP systems because of their proven reliability. Here are just a few of the systems out there.

3.2.2.1 Speak Freely

Speak Freely was designed by John Walker as a freely available chat application and an alternative to conventional long distance communication. It uses GSM and CELP Codecs for voice compression and DES, Blowfish and IDEA ciphers for encryption. It is written in C and available on many different platforms. Speak Freely can operate using different protocols, and communication with different applications is therefore possible. Its development was discontinued in August 1st, 2003 due to outdated technology and NAT/firewall issues.

3.2.2.2 TeamSpeak

TeamSpeak is a freely available VOIP system designed to provide interactive audio for online games, and minimizes latency by using Codecs such as CELP

and GSM, which have high compression ratios. TeamSpeak uses servers in a centralized configuration to overcome the obstacles imposed by NAT.

3.2.2.3 Skype

Skype is a decentralized VOIP system developed by Kazaa that employs a pre-existing peer-to-peer network to route conversations. Skype circumvents NAT by using non-fire-walled hosts as 3rd parties. It is free to users and provides good sound quality and encryption.

3.2.2.4 HawkVoiceDI

HawkVoiceDI, created by Hawk Software, is a voice network API that is used in online games. It was designed as an open source alternative to proprietary APIs. HawkVoiceDI is programmer-friendly and offers a choice of eight Codecs for compression. It uses MD5 and Blowfish for stream encryption.

3.2.2.5 ITU Standards

The ITU defines international protocols and standards that govern many widely used communication networks. Time constraints prevented this project from employing the H.323 standard for packet-based multimedia communications.

3.2.2.6 Strengths of Existing Systems

Current technology can produce high quality voice transfer rivaling that of analogue telephony. This is partly due to improved compression rates and the advancing technology associated with higher bandwidth, which improves audio quality by decreasing latency.

Information security has also evolved. Implementations such as Blowfish theoretically offer strong encryption, although awareness and implementation has been inadequate.

3.2.2.7 Weaknesses

Two NAT users are usually unable to connect to each other in a decentralized manner. The conventional technique for circumventing NAT uses a non-fire-walled 3rd party to connect the two fire-walled users. Doing so, however, incurs additional latency and Internet traffic. Although these constraints are unavoidable, a good solution will need to take into account this balance find a well-designed answer to VOIP. With VOIP, the specific shortcomings are with respect to NAT incompatibility, and bandwidth versus audio quality trade-offs.

3.2.3 Audio Codec Research

VOIP systems use compression to minimize bandwidth and latency. The Codec used determines the compression ratio, and frame size.

3.2.3.1 μ -Law

μ-Law is perhaps the most established voice communications standard, and is used internationally to encode audio telecommunications. μ-Law's speed derives from its using a simple logarithmic look-up table that encodes only the 13 most significant bits of a 16-bit sample, yielding a 2:1 compression ratio. μ -Law's European counterpart is called A-Law.

3.2.3.2 ADPCM

ADPCM is one of the most widely used Codecs. It has been heavily integrated into ITU's standards and is available on almost any platform that can handle

voice communications. ADPCM uses a table look-up system and encodes only the differences between consecutive samples. Its compression ratio is 4 to 1.

3.2.3.3 GSM

Global System for Mobile Communication was specifically developed for wireless digital voice telephony and can be found on most platforms. GSM employs LPC and therefore shares many of its characteristics such as a high compression ratio. GSM has a 10 to 1 compression ratio.

3.2.3.4 LPC

LPC is among the most effective compressions currently available for audio communication. However, it is computationally intensive and the compression quality degrades at high frequencies. LPC predicts the likelihood of a waveform's progression and measures differences between the predicted and actual waveform. Its compression ratio is 12 to 1 and a modification by the US Military called LPC-10 has a compression ratio of 26 to 1.

3.2.3.5 Codec Comparison

Codecs are evaluated by comparing the optimal tradeoff between compression speed and ratio.

Table 3.1 – Codec Comparison

	μ - Law	**ADPCM**	**GSM 6.10**	**LPC**
Data Rate	64 Kbps	32 Kbps	13.2 Kbps	10.7 KBps
Compression Factor	2 : 1	4 : 1	10 : 1	12 : 1
Relative compression rating	1	2	3	4
Overall rating	3	1	2	4

ADPCM was used for this VOIP system because it offers good compression while requiring little computational time. This algorithm will be discussed in

further detail later in this report. GSM 6.10 is also attractive due to its high compression ratio. However, GSM and LPC are computationally expensive and would be better suited for dedicated audio devices.

3.2.4 Network packet Latency

Packet latency is a measure of how long it takes a packet to get from one point to another over a network. In designing our protocol, we had to carefully consider packet latency because this defines a *lower limit* on the expected audio delays. Since one of our primary goals is to achieve low audio latency, we had to investigate actual network conditions in order to design our software to achieve the minimum latency possible.

While the relatively static path of routers between two IP hosts defines the minimum path latency, the latency for any particular packet can vary substantially due to congestion at routers or traffic prioritization. For this project, the packet latency testing attempted to gain some insight into typical network conditions that was to be dealt by the VOIP software. The aim were

- Examine the relationship between packet size and latency. Theory predicts that latency varies linearly with packet size due to the constant bit-rate of transmit/receive operations. This immediately suggests that our protocol should use small packets since these are delivered more quickly than larger packets. However, since actual Internet router behavior is much less predictable, we wanted to see whether the linear

relationship held or whether there was in fact some "ideal packet sizes" that consistently resulted in lower latencies.

- Examine how latency can change over time. Over a period of hours or even minutes, latency due to router congestion can change quite dramatically. Much of this has to do with peak-hour usage and other factors that are far beyond our control. Nevertheless, observing these changes can help us introduce adequate provisions into the protocol.
- Examine latency ranges. We wish to determine the overall ranges in latency for a link.

While it is common for network utilities such as 'ping' to use ICMP packets for latency measurements, the UDP packet latency was to be tested since the software uses UDP and not ICMP. This is particularly important because routers can treat UDP packets very differently from other protocols. For this purpose several versions of a C program, called udpstat.c, were written, that can gather statistics on UDP packets. The software does automated tests according to programmed parameters, and writes raw output files which we can then graph or otherwise analyze.

VOIP is a standard for the communication in internet where the speech data is to be transferred to long distances.

3.3 SIP Overview

There are many applications of the Internet that require the creation and management of a session, where a session is considered an exchange of data

between an association of participants. The implementation of these applications is complicated by the practices of participants: users may move between endpoints, they may be addressable by multiple names, and they may communicate in several different media - sometimes simultaneously. Numerous protocols have been authored that carry various forms of real-time multimedia session data such as voice, video, or text messages. SIP works in concert with these protocols by enabling Internet endpoints (called *user agents*) to discover one another and to agree on a characterization of a session they would like to share. For locating prospective session participants, and for other functions, SIP enables creation of an infrastructure of network hosts (called *proxy servers*) to which user agents can send registrations, invitations to sessions, and other requests. SIP is an agile, general-purpose tool for creating, modifying, and terminating sessions that works independently of underlying transport protocols and without dependency on the type of session that is being established.

SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports *personal mobility* [33, p.44] - users can maintain a single externally visible identifier regardless of their network location. SIP supports five facets of establishing and terminating multimedia communications

- **User location:** determination of the end system to be used for communication
- **User availability:** determination of the willingness of the called party to engage in communications
- **User capabilities:** determination of the media and media parameters to be used
- **Session setup:** “ringing”, establishment of session parameters at both called and calling party
- **Session management:** including transfer and termination of sessions, modifying session parameters, and invoking services

SIP is not a vertically integrated communications system. SIP is rather a component that can be used with other IETF protocols to build a complete multimedia architecture. Typically, these architectures will include protocols such as the RTP [34] for transporting real-time data and providing QOS feedback, the real-time streaming protocol for controlling delivery of streaming media, the MEGACO protocol for controlling gateways to the PSTN, and the SDP [35] for describing multimedia sessions. Therefore, SIP should be used in conjunction with other protocols in order to provide complete services to the users. However, the basic functionality and operation of SIP does not depend on any of these protocols. SIP does not provide services. SIP rather provides primitives that can be used to implement different services. For example, SIP can locate a user and deliver an opaque object to his current location. If this primitive is used to deliver a session description written in SDP, for instance, the endpoints can agree on

the parameters of a session. If the same primitive is used to deliver a photo of the caller as well as the session description, “caller ID” service can be easily implemented. As this example shows, a single primitive is typically used to provide several different services. SIP does not offer conference control services such as floor control or voting and does not prescribe how a conference is to be managed. SIP can be used to initiate a session that uses some other conference control protocol. Since SIP messages and the sessions they establish can pass through entirely different networks, SIP cannot, and does not, provide any kind of network resource reservation capabilities. The nature of the services provided make security particularly important. To that end, SIP provides a suite of security services, which include denial-of-service prevention, authentication (both user to user and proxy to user), integrity protection, and encryption and privacy services. SIP works with both IPv4 and IPv6.

3.3.1 Overview of Operation

This section introduces the basic operations of SIP using simple examples. This section is tutorial in nature and does not contain any normative statements.

The first example shows the basic functions of SIP: location of an end point, signal of a desire to communicate, negotiation of session parameters to establish the session, and teardown of the session once established. Figure 3.2 shows a typical example of a SIP message exchange between two users, Alice and Bob. (Each message is labeled with the letter “F” and a number for reference by the text.) In this example, Alice uses a SIP application on her PC (referred to as a soft phone) to call Bob on his SIP phone over the Internet. Also shown are two

SIP proxy servers that act on behalf of Alice and Bob to facilitate the session establishment. This typical arrangement is often referred to as the “SIP trapezoid” as shown by the geometric shape of the dashed lines in Figure 3.2. Alice “calls” Bob using his SIP identity, a type of URI called a SIS

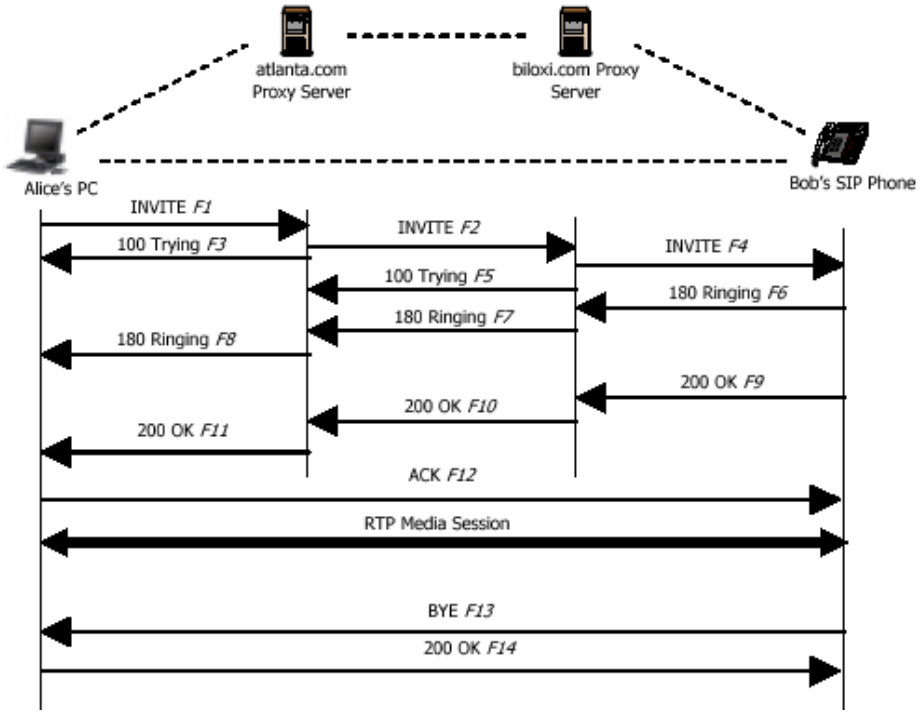

Figure 3.2 – SIP Operation

In this case, bob’s SIP URI is sip:bob@biloxi.com, where biloxi.com is the domain of Bob’s SIP service provider. Alice also has a SIP URI of sip:alice@atlanta.com. Alice might have typed in Bob’s URI or perhaps clicked on a hyperlink or an entry in an address book. SIP also provides a secure URI, called a SIPS URI. An example would be sips:bob@biloxi.com. A call made to a SIPS URI guarantees that secure, encrypted transport is used to carry all SIP messages from the caller to the domain of the callee. From there, the request is sent securely to the callee, but with security mechanisms that depend on the

policy of the domain of the callee. SIP is based on an HTTP-like request/response transaction model. Each transaction consists of a request that invokes a particular *method*, or function, on the server and at least one response. In this example, the transaction begins with Alice's soft-phone sending an INVITE request addressed to Bob's SIP URI. INVITE is an example of a SIP method that specifies the action that the requestor (Alice) wants the server (Bob) to take. The INVITE request contains a number of header fields. Header fields are named attributes that provide additional information about a message. The ones present in an INVITE include a unique identifier for the call, the destination address, Alice's address, and information about the type of session that Alice wishes to establish with Bob. The INVITE might look like

```
INVITE sip:bob@biloxi.com SIP/2.0
Via:SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From:Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

Figure 3.3 – SIP Invite

The first line of the text-encoded message contains the method name (INVITE). The lines that follow are a list of header fields. This example contains a minimum required set. The header fields are briefly described in the following paragraphs

Via contains the address (pc33.atlanta.com) at which Alice is expecting to receive responses to this request. It also contains a branch parameter that contains an identifier for this transaction.

To contains a display name (Bob) and a SIP or SIPS URI (sip:bob@biloxi.com) towards which the request was originally directed.

From also contains a display name (Alice) and a SIP or SIPS URI (sip:alice@atlanta.com) that indicate the originator of the request. This header field also has a tag parameter containing a pseudorandom string (1928301774) that was added to the URI by the soft-phone. It is used for identification purposes.

Call-ID contains a globally unique identifier for this call, generated by the combination of a pseudorandom string and the soft-phone's IP address. The combination of the To tag, From tag, and Call-ID completely define a peer-to-peer SIP relationship between Alice and Bob and is referred to as a *dialog*.

CSeq or Command Sequence contains an integer and a method name. The CSeq number is incremented for each new request within a dialog and is a traditional sequence number.

Contact contains a SIP or SIPS URI that represents a direct route to contact Alice, usually composed of a username at FQDN. While an FQDN is preferred, many end systems do not have registered domain names, so IP addresses are permitted. While the Via header field tells other elements where to send the response, the Contact header field tells other elements where to send future requests.

Max-Forwards serves to limit the number of hops a request can make on the way to its destination. It consists of an integer that is decremented by one at each hop.

Content-Type contains a description of the message body. Content-Length contains a byte count of the message body.

The details of the session, type of media, codec, sampling rate, etc. are not described using SIP. Rather, the body of a SIP message contains a description of the session, encoded in some other protocol format. One such format is the SDP [33]. This SDP message is carried by the SIP message in a way that is analogous to a document attachment being carried by an email message, or a web page being carried in an HTTP message. Since the soft-phone does not know the location of Bob or the SIP server in the biloxi.com domain, the soft-phone sends the INVITE to the SIP server that serves Alice's domain, atlanta.com. The address of the atlanta.com SIP server could have been configured in Alice's soft-phone, or it could have been discovered by DHCP, for example. The atlanta.com SIP server is a type of SIP server known as a proxy server. A proxy server receives SIP requests and forwards them on behalf of the requestor. In this example, the proxy server receives the INVITE request and sends a 100 (Trying) response back to Alice's soft-phone. The 100 response indicates that the INVITE has been received and that the proxy is working on her behalf to route the INVITE to the destination. Responses in SIP use a three-digit code followed by a descriptive phrase. This response contains the same To, From, Call-ID, CSeq and branch parameter in the Via as the INVITE, which

allows Alice's soft phone to correlate this response to the sent INVITE. The atlanta.com proxy server locates the proxy server at biloxi.com, possibly by performing a particular type of DNS lookup to find the SIP server that serves the biloxi.com domain. This is described in [36]. As a result, it obtains the IP address of the biloxi.com proxy server and forwards, or proxies, the INVITE request there. Before forwarding the request, the atlanta.com proxy server adds an additional Via header field value that contains its own address (the INVITE already contains Alice's address in the first Via). The biloxi.com proxy server receives the INVITE and responds with a 100 response back to the atlanta.com proxy server to indicate that it has received the INVITE and is processing the request. The proxy server consults a database, generically called a location service, which contains the current IP address of Bob. The biloxi.com proxy server adds another Via header field value with its own address to the INVITE and proxies it to Bob's SIP phone. Bob's SIP phone receives the INVITE and alerts Bob to the incoming call from Alice so that Bob can decide whether to answer the call, that is, Bob's phone rings. Bob's SIP phone indicates this in a 180 (Ringing) response, which is routed back through the two proxies in the reverse direction. Each proxy uses the Via header field to determine where to send the response and removes its own address from the top. As a result, although DNS and location service lookups were required to route the initial INVITE, the 180 response can be returned to the caller without lookups or without state being maintained in the proxies. This also has the desirable property that each proxy that sees the INVITE will also see all responses to the INVITE. When Alice's soft phone receives the 180 response, it

passes this information to Alice, perhaps using an audio ring back tone or by displaying a message on Alice's screen. In this example, Bob decides to answer the call. When he picks up the handset, his SIP phone sends a 200 (OK) response to indicate that the call has been answered. The 200 contains a message body with the SDP media description of the type of session that Bob is willing to establish with Alice. As a result, there is a two-phase exchange of SDP messages: Alice sent one to Bob, and Bob sent one back to Alice. This two-phase exchange provides basic negotiation capabilities and is based on a simple offer/answer model of SDP exchange. If Bob did not wish to answer the call or was busy on another call, an error response would have been sent instead of the 200, which would have resulted in no media session being established. The 200 might seem to be

SIP/2.0 200 OK

Via: SIP/2.0/UDP 192.2.1.1:5060;branch=4b43c2ff8.1

Via: SIP/2.0/UDP 192.1.1.1:5060;branch=77ef4c2312983.1

Via: SIP/2.0/UDP 192.1.3.3:5060

To: Bob <sip:bob@biloxi.com>;tag=a6c85cf

From: Alice <sip:alice@atlanta.com>;tag=1928301774

Call-ID: a84b4c76e66710@192.1.3.3

CSeq: 314159 INVITE

Contact: <sip:bob@192.0.2.4>

Content-Type: application/sdp

Content-Length: 131

(Bob's SDP not shown)

Figure 3.4 – SIP OK (200) Response

The first line of the response contains the response code (200) and the reason phrase (OK). The remaining lines contain header fields. The Via, To, From, Call-ID, and CSeq header fields are copied from the INVITE request. Bob's SIP phone has added a tag parameter to the To header field. This tag will be incorporated by both endpoints into the dialog and will be included in all future requests and responses in this call. The Contact header field contains a URI at which Bob can be directly reached at his SIP phone. The Content-Type and Content-Length refer to the message body that contains Bob's SDP media information. In addition to DNS and location service lookups shown in this example, proxy servers can make flexible "routing decisions" to decide where to send a request. For example, if Bob's SIP phone returned a 486 (Busy Here) response, the biloxi.com proxy server could proxy the INVITE to Bob's voicemail server. A proxy server can also send an INVITE to a number of locations at the same time. This type of parallel search is known as *forking*. In this case, the 200 is routed back through the two proxies and is received by Alice's soft-phone, which then stops the ring-back tone and indicates that the call has been answered. Finally, Alice's soft-phone sends an ACK to Bob's SIP phone to confirm the reception of the final response (200). In this example, the ACK is sent directly from Alice's soft-phone to Bob's SIP phone, bypassing the two proxies. This occurs because the endpoints have learned each other's address from the Contact header fields through the INVITE/200 (OK) exchange, which was not known when the initial INVITE was sent. The lookups performed by the two proxies are no longer needed, so the proxies drop out of the call flow. This

completes the INVITE/200/ACK three-way handshake used to establish SIP sessions. Alice and Bob's media session has now begun, and they send media packets using the format to which they agreed in the exchange of SDP. In general, the end-to-end media packets take a different path from the SIP signaling messages. During the session, either Alice or Bob may decide to change the characteristics of the media session. This is accomplished by sending a re-INVITE containing a new media description. This re-INVITE references the existing dialog so that the other party knows that it is to modify an existing session instead of establishing a new session. The other party sends a 200 to accept the change. The requestor responds to the 200 with an ACK. If the other party does not accept the change, he sends an error response such as 406 (Not Acceptable), which also receives an ACK. However, the failure of the re-INVITE does not cause the existing call to fail - the session continues using the previously negotiated characteristics. At the end of the call, Bob disconnects (hangs up) first and generates a BYE message. This BYE is routed directly to Alice's soft phone, again bypassing the proxies. Alice confirms receipt of the BYE with a 200 response, which terminates the session and the BYE transaction. No ACK is sent - an ACK is only sent in response to a response to an INVITE request. The reasons for this special handling for INVITE will be discussed later, but relate to the reliability mechanisms in SIP, the length of time it can take for a ringing phone to be answered, and forking. For this reason, request handling in SIP is often classified as either INVITE or non-INVITE, referring to all other methods besides INVITE. In some cases, it may be useful for proxies in the SIP

signaling path to see all the messaging between the endpoints for the duration of the session. For example, if the biloxi.com proxy server wished to remain in the SIP messaging path beyond the initial INVITE, it would add to the INVITE a required routing header field known as Record-Route that contained a URI resolving to the hostname or IP address of the proxy. This information would be received by both Bob's SIP phone and (due to the Record-Route header field being passed back in the 200 Alice's soft-phone and stored for the duration of the dialog. The biloxi.com 521proxy server would then receive and proxy the ACK, BYE, and 200 to the BYE. Each proxy can independently decide to receive subsequent messaging, and that messaging will go through all proxies that elect to receive it. This capability is frequently used for proxies that are providing mid-call features. Registration is another common operation in SIP. Registration is one way that the biloxi.com server can learn the current location of Bob. Upon initialization, and at periodic intervals, Bob's SIP phone sends REGISTER messages to a server in the biloxi.com domain known as a SIP registrar. The REGISTER messages associate Bob's SIP or SIPS URI (sip:bob@biloxi.com) with the machine into which he is currently logged. The registrar writes this association, also called a binding, to a database, called the *location service*, where it can be used by the proxy in the biloxi.com domain. Often, a registrar server for a domain is co-located with the proxy for that domain. It is an important concept that the distinction between types of SIP servers is logical, not physical.

Bob is not limited to registering from a single device. For example, both his SIP phone at home and the one in the office could send registrations. This

information is stored together in the location service and allows a proxy to perform various types of searches to locate Bob. Similarly, more than one user can be registered on a single device at the same time. The location service is just an abstract concept. It generally contains information that allows a proxy to input a URI and receive a set of zero or more URIs that tells the proxy where to send the request. Registrations are one way to create this information, but not the only way. Arbitrary mapping functions can be configured at the discretion of the administrator. Finally, it is important to note that in SIP, registration is used for routing incoming SIP requests and has no role in authorizing outgoing requests. Authorization and authentication are handled in SIP either on a request-by-request basis with a challenge/response mechanism, or by using a lower layer scheme Additional operations in SIP, such as querying for the capabilities of a SIP server or client using OPTIONS, or canceling a pending request using CANCEL.

3.4 Summary

VOIP are the technologies that allow voice communications carried within IP data packets. Audio is digitized, compressed, and then filled into multiple IP packets. These data packets travel through a packet-switched network such as the Internet and arrive at their destination where they are decompressed using a compatible and converted back to the audio signal.

SIP is an application-layer protocol that can establish, modify, and terminate sessions. SIP can also invite participants to already existing sessions, such as multicast conferences. There are many applications of the Internet that require

the creation and management of a session, where a session is considered an exchange of data between an association of participants. The implementation of these applications is complicated by the practices of participants: users may move between endpoints, they may be addressable by multiple names, and they may communicate in several different media - sometimes simultaneously. SIP works in concert with different protocols by enabling user agents to discover one another and to agree on a characterization of a session they would like to share. For locating prospective session participants, and for other functions, SIP enables creation of an infrastructure of network hosts (called proxy servers) to which user agents can send registrations, invitations to sessions, and other requests.

4

Project Infrastructure

4.1 Introduction

This part of the thesis covers the layout of our project. What all things were available and what all components were designed and how they were linked to each other.

4.2 Project Layout

The basic layout of the project is shown in figure 4.1. As it can be seen, from the figure, the layout comprises of mainly five components. The GSM Network, internet, NAT, Communication Servers and the PDAs are already there in the mobile communication sector.

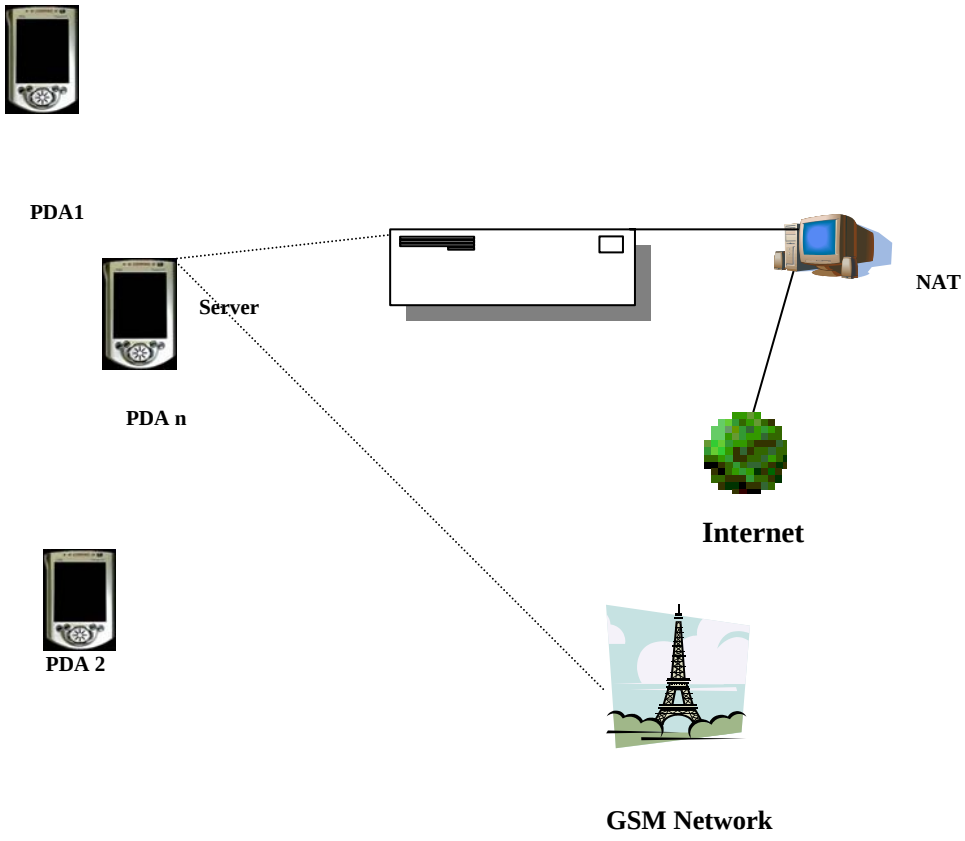

Figure 4.1 – Project layout

Our primary task was to make two PDAs communicate over a wireless LAN setup.

4.2.1 PDA's

Since PDA's has built in Wireless LAN which follows the standard 802.11b. Now we can use the already provided functionality of Wireless LAN, existing in the PDAs. We have to use this functionality and transfer voice data over it. So, by using infrastructure mode of wireless LAN as well as we can use peer to peer communication within wireless LAN range.

4.2.2 Access Point

Access points are used for communication of voice over the internet. All the mobile nodes use the access point to connect itself to NAT Server.

4.2.3 NAT Server

NAT Server is being used for the purpose if multiple computers are connected to it, and all the connected nodes are sharing the same IP address .The same server is using firewall. All the local network generated traffic for internet is passed through this server and the same case is for incoming traffic.

For this communication the paradigm used is Client-Server. A centralized system is being provided for all the mobile nodes. Here it can be understood that mobile nodes can be a PC with wireless LAN cards or Laptops with wireless LAN facility .Although actual aim is to implement it on the PDA's. If the person is available in wireless LAN range, the communication is peer-to-peer.

The NAT Server contains the SIP Proxy, required for SIP based clients. This proxy works in the same way simple proxy server of e.g. Yahoo, user

authentication, new user etc. the complete functionality of SIP has been discussed in the previous chapter. Though, the development of SIP Proxy is not a part of our project. We have used it in a way to test our own SIP Clients.

4.2.4 GSM Network

Another communication media is through GSM. If person is unable to connect through the internet he would be able to connect through GSM.

4.3 Infrastructure Introduction

From previous chapters we have understood the basic functionality of Wireless LAN, SIP based VOIP and the GSM networks.

The project is being divided into three major phases

- Wireless LAN interface development phase
- GSM Programming Phase
- Interface between Wireless LAN and GSM Network phase.

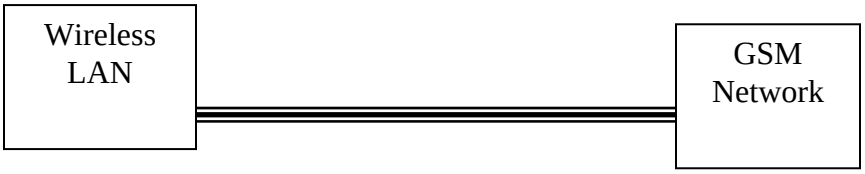

Figure 4.2 – Interface between wireless LAN & GSM network

The following headings would cover the above mentioned points in detail.

4.3.1 Wireless LAN Interface Development Phase

The following figure puts an emphasis on the fact that the wireless LAN Infrastructure comprises of three major phases.

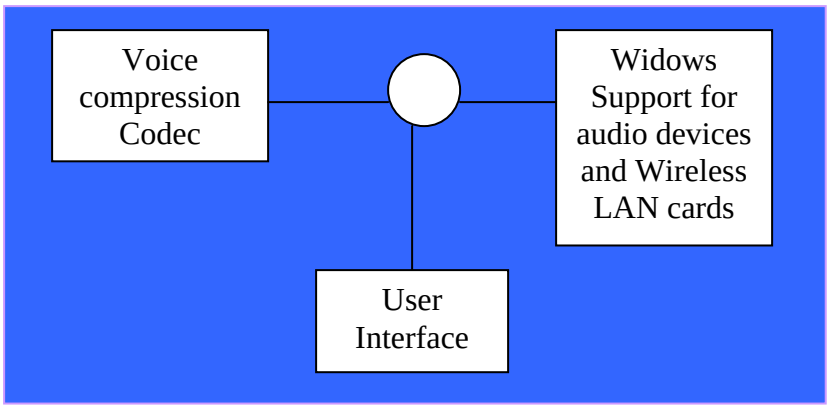

Figure 4.3 – Wireless LAN infrastructure

The major phases are

- Voice Compression Codec
- Windows support for audio devices and wireless LAN cards
- User interface

4.3.1.1 Voice Compression Codec

For peer-to-peer communication GSM 6.10 voice compression codec is used. And for communication over the internet G.711 voice compression codec is used. As basic functionality of these Codecs is provided in the previous chapter.

4.3.1.2 Windows Support for Audio Devices & Wireless LAN Cards

This is basic driver support for wireless LAN cards and audio devices like speaker and microphone is being provided by using DLL's of windows for our project.

4.3.1.3 User Interface

User interface for getting IP address, name and cell number as required in common mobile hand sets. In case of communication over internet user enters

his address in terms of syntax like kashif@mcs.com.So, two different versions are available. Since the project is under development phase.

4.3.2 GSM Programming Phase

For GSM programming standard SDK is available called GNOKII. We have used its version gnokii-0.6.2.Complete details of it would be provided in the next chapter. As basic introduction is enough at this time.

5

Wireless LAN Implementation Phase

5.1 Introduction

In this chapter the complete functionality of the wireless LAN implementation phase would be included.

5.2 Communication over Local LAN

Bottom-up software approach was being used for its development. The following software design was made and work was done. Let from software engineering perspective if various components are depending upon each other and due to this dependency provide some functionality we can put them in one package.

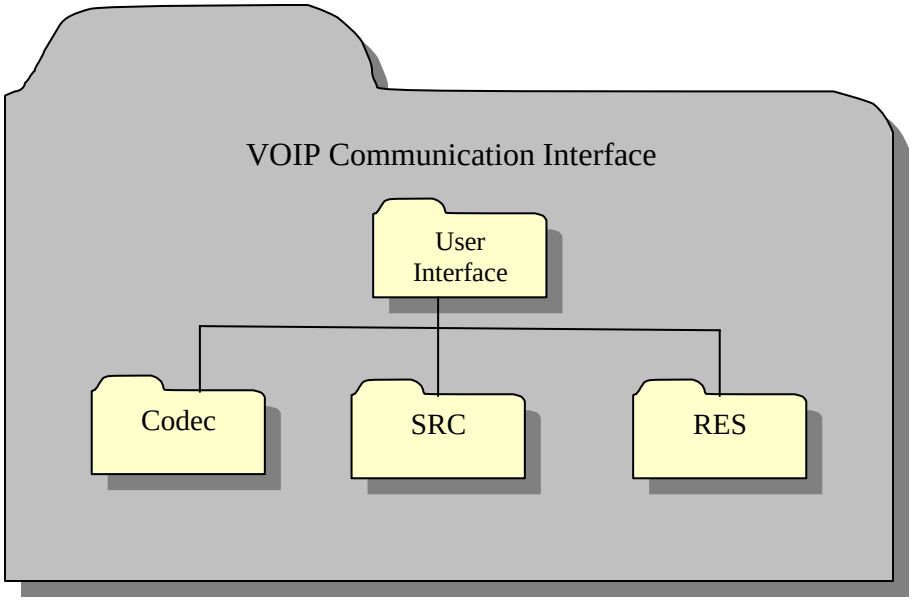

Figure 5.1 - VOIP Communication Interface Package

5.2.1 Package User Interface

This package contains all the different GUI components, which are required to facilitate the user to communicate with each other. These are the group of several *.cpp (c++ files) and *.h (header files) that takes required information from the user.

5.2.2 Package Codec

This package contains the codec components that make it possible for voice packets to travel over the IP networks.

5.2.3 Package SRC

This package contains components that make it possible to use windows media devices like speaker and microphone etc.

5.2.4 Package RES

This package contains components that are displayed on GUI like pictures for add button, delete button and icons etc.

5.3 Call State Model

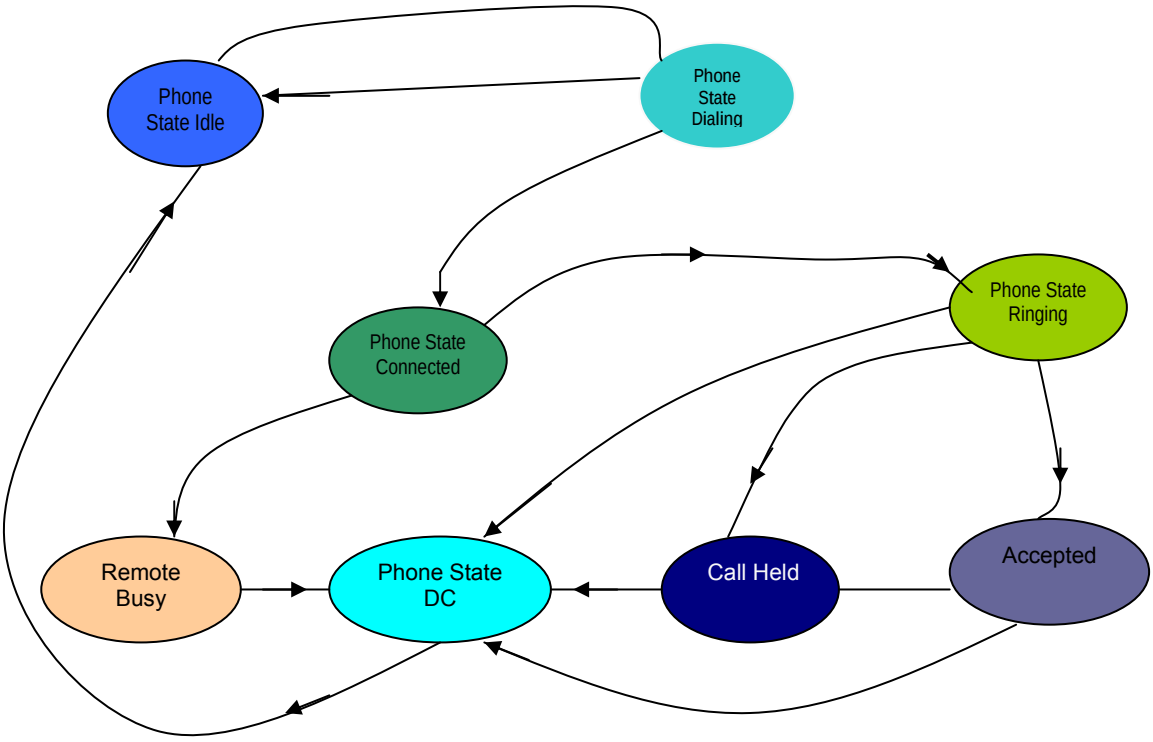

Figure 5.2 – Call State Model

At any instant of time user can be in any of the following state, has been shown in state model diagram, where the arrow heads are showing the state towards which transition is taking place and the oval shapes show the different states.

5.4 Network Programming

It has been done using the windows sockets. In order to fall into the details of network programming, we need to come to know of some important concepts.

5.4.1 Socket

A socket is a communication endpoint — an object through which a Windows Sockets application sends or receives packets of data across a network. A socket has a type and is associated with a running process, and it may have a name. Currently, sockets generally exchange data only with other sockets in the same “communication domain,” which uses the Internet Protocol Suite. Both kinds of sockets are bi-directional: they are data flows that can be communicated in both directions simultaneously (full-duplex). Two socket types are

- Stream Sockets
- Datagram Sockets

5.4.1.1 Stream Sockets

Stream sockets provide for a data flow without record boundaries — a stream of bytes. Streams are guaranteed to be delivered and to be correctly sequenced and unduplicated.

5.4.1.2 Datagram Sockets

Datagram sockets support a bi-directional data flow that is not guaranteed to be sequenced or unduplicated. Datagram also are not guaranteed to be reliable;

they can fail to arrive. Datagram data may arrive out of order and possibly duplicated, but record boundaries in the data are preserved, as long as the records are smaller than the receiver's internal size limit. You are responsible for managing sequencing and reliability. (Reliability tends to be good on LANs but less so on WANs, such as the Internet).

5.5 Our Network Programming Model

For our system we have chosen the Datagram sockets for a number of reasons. One of the main reasons being we have mainly concentrated on voice and video communication. Even if the Datagram sockets are not reliable, the error chances are very low.

5.6 The Protocol Architecture

The architecture of the protocol, that has been used in our project's infrastructure comprises of

- The Data Link Layer
- The Network Layer
- The Transport Layer
- The Session Layer
- The Application Layer

5.6.1 The Data Link Layer

The main task of our data link layer is to take the data from the higher layer and sent it to the sockets. This layer ensures that the data is transmitted successfully over the network and generates error messages in case of any transmission or receiving errors. The UDP protocol is used to ensure reliable data transfer between the two nodes.

5.6.2 The Network Layer

The network layer is concerned with controlling the operation of the subnet. A key design issue is determining how the packets are routed from source to the destination.

5.6.3 The Transport Layer

The basic function of the transport layer is to accept data from the session layer, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore all this must be done efficiently, and in a way that isolates the upper layers from the inevitable changes in the hardware technology.

The transport layer also determines what type of service to provide the session layer, and ultimately, the users of the network. The transport connection used in our protocol stack is connection less. This service has been opted as the wireless networks require small bursts of data to be transmitted and irrespective of the forthcoming and preceding packets. In contrast the connection oriented service offer pipes for the data transfer that may not be available in the wireless networks.

5.6.4 The Session Layer

The session layer allows users on different machines to establish sessions between them. A session allows ordinary data transport, as does the transport, as does the transport layer, but it also provides enhanced services useful in some applications. A session might be used to allow a user to log into a remote timesharing system or to transfer a file between two machines.

The main service offered by our session layer is the establishing of a voice link between two machines using the standard call-connect procedures. A caller may initiate a call with another node by sending a call request packet. The called machine replies to the request as by accepting or rejecting the call request. Once a call is accepted the voice data packet, create process initiates unless the caller or the called machine ends the call.

5.6.5 The Application Layer

The Application Layer of our protocol stack may be treated as the top layer user interface that allows user's interaction with our network.

5.7 Communication over the Internet

In previous all we have done is for peer to peer communication. Now can we direct over voice over the internet? .But to send voice over the internet this requires little bit complex architecture for voice call.

5.7.1 SIP Clients Implementation

For SIP Clients, the list of libraries include

- SIPCALLLIB-Call Processing Library
- SIPMEDIALIB-Media Processing Library
- SIPPORTLIB-OS abstraction Layer and portability library

Figure 5.3 - SIP Clients

The specifications for SIP based VOIP Phone are

- Call Transfer
- Automatic Echo Cancellation
- Caller ID
- SIP URL calling
- Most recently used numbers list
- Volume and record level adjustments
- STUN support
- sipXtapi User Agent SDK Key Features
- TAPI-like API
- Multiple simultaneous calls
- Hold/Unhold, Mute/Unmute
- Volume and Gain control
- Selectable audio input device, speaker device, and ringer device
- Logical separation between ringer and speaker
- Silent Call Forwarding and Rejection
- Tone Generation (DTMF, busy, ring back, etc.)
- Play audio from file
- Bind to specific network interfaces
- RFC3261 compliant SIP stack
- RFC2833 Out of band DTMF tones
- RTP suppression on mute
- Configurable Ports (UDP, TCP, and RTP)

- Multiple SIP identities (line appearances)
- Line authentication
- DNS SRV timeout control
- SIP Proxy control
- G711 Codec
- RFC-2543

5.7.2 SIPCALLLIB-Call Processing Library

The sipcallLib project provides an abstract call processing layer on top of the sipXmediaLib, media processing library, and sipXtackLib, SIP protocol stack . This layer provides a call model on top of the basic SIP constructs and manages life cycles and resources. Additionally, sipXcallLib enforces a safe threading model.

The heart of the call processing library includes call manager classes, call classes, connection classes, and media interface. However, a number of different APIs/methods exist to manipulate the call manager.

5.7.3 SIPMEDIALIB-Media Processing Library

The sipXmediaLib includes all of the audio processing .For example, the library contains audio bridges, audio splitters, echo suppression, tone generation (e.g. DTMF), streaming support, RTCP, G711 codecs, etc.

The highest level object in the sipXMediaLib is the MpFlowGraph. The flow graph assembles a number of media resources in a defined order of media flow where each resource has zero or more inputs and zero or more outputs. The resources are process in the order implied by the connection topology. The following

illustrates the logical connections of the MpCallFlowGraph. Resources may be added dynamically to the flow graph and new resource types can be derived. A connection is a logical construct containing the resources related to a single remote media source. There may be zero or more connections in a MpCallFlowGraph.

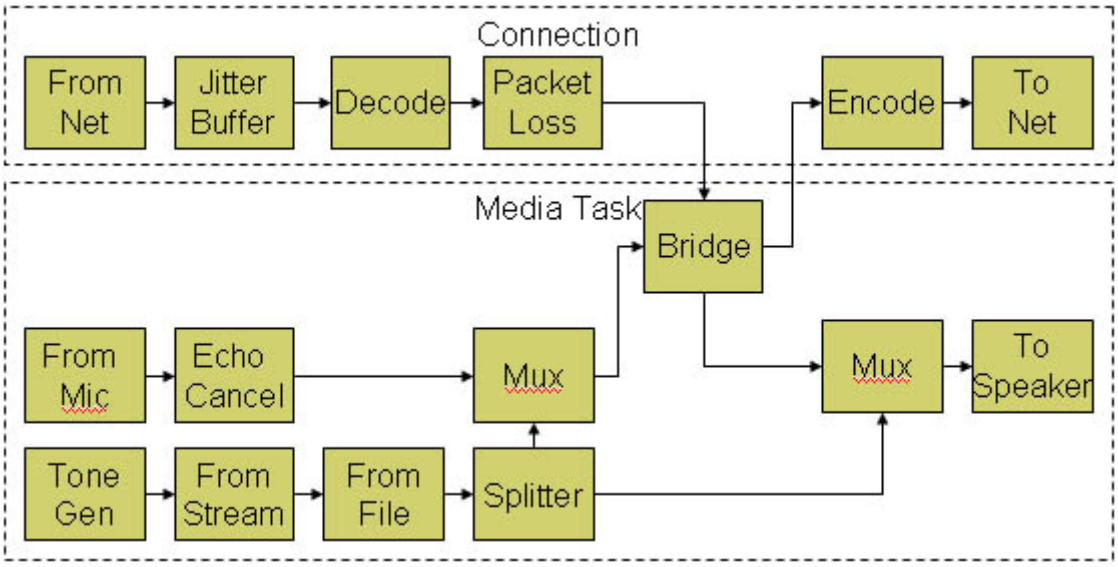

Figure 5.4 - MpCallFlowGraph

5.7.4 SIPPORTLIB-OS abstraction Layer and Portability Library

This library provides a set of classes that provide an operating system abstraction from a majority of OS provided functions.. The library currently provides classes that encapsulate functions and operations for

- Threads
- Locks and Mutexes
- Semaphores
- Messages and Queues

- Timers
- Time and Date
- Sockets
- File and Directory
- Operating System Processes
- Dynamic loading of shared libraries and symbols

5.8 Summary

The chapter covered the complete functionality of the wireless LAN implementation phase. That encompasses the different libraries that are developed or used.

6

Voice Communication

6.1 Introduction

In the previous chapter the Internet Protocol was explained. This was done in a general way, without paying much attention to Voice over IP. Since we now know the most important features of the protocol, we can bring other components of VOIP into the picture.

In this chapter we will take a closer look at some aspects of digitized voice communication. The chapter starts with a discussion about grabbing and reconstruction of voice signals. Next, the requirements for a reasonably good form of voice communication are given. We will then take a closer look at communication patterns and finally we will see what the impact of all these things is on VOIP.

6.2 Grabbing and Reconstruction

Before you can send voice information over a packet network, you must first digitize the voice signal. After the transmission, the receiver of this digitized signal has to convert it back to an analogue signal, which can be used to generate speaker output. The first stage is also called `grabbing' of the voice signal and the second stage is called reconstruction. In general, these stages are also referred to as analogue-to-digital (A/D) conversion and digital-to-analogue (D/A) conversion, respectively.

As for terminology, it is useful to know that digitizing an audio signal is often referred to as pulse code modulation (PCM).

Nowadays, digitization and reconstruction of voice signals can be done by any PC soundcard, so this is not the most difficult step in creating VOIP applications. For completeness, however, I will give a brief description of the processes.

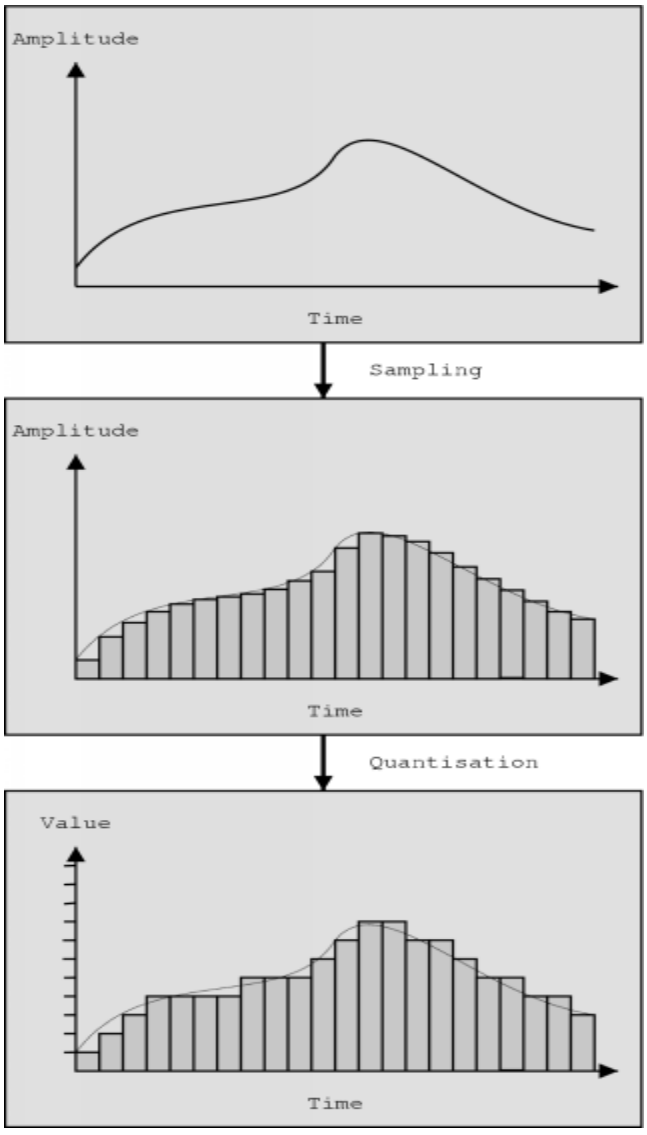

Figure 6.1- Sampling and quantization

6.2.1 Sampling and Quantization

A continuous signal on a certain time interval has an infinite number of values with infinite precision. To be able to digitally store an approximation of the signal,

it is first sampled and then quantized. When you sample a signal, you take infinite precision measures at regular intervals. The rate at which the samples are taken is called the sampling rate.

The next step is to quantize the sampled signal. This means that the infinite precision values are converted to values which can be stored digitally.

In general, the purpose of quantization is to represent a sample by an N-bit value. With uniform quantization, the range of possible values is divided into 2N equally sized segments and with each segment; an N-bit value is associated. The width of such a segment is known as the step size. This representation results in clipping if the sampled value exceeds the range covered by the segments. [10]

With non-uniform quantization, this step size is not constant. A common case of non-uniform quantization is logarithmic quantization. Here, it is not the original input value that is quantized, but in fact the log value of the sample. For audio signals this is particularly useful since humans tend to be more sensitive to changes at lower amplitudes than at high ones [23]. Another non-uniform quantization method is adaptive quantization [10]. With such methods, the quantization step size is dynamically adapted in response to changes in the signal amplitude. PCM techniques which use adaptive quantization are referred to as adaptive PCM (APCM). The sampling and (uniform) quantization steps are depicted in figure 6.1. An important thing to note is that both steps introduce a certain amount of error. It is clear that a higher sampling rate and a smaller quantization step size will reduce the amount of error in the digitized signal.

6.2.2 Reconstruction

Signal reconstruction does the opposite of the digitization step. An inverse quantization is applied and from those samples a continuous signal is recreated. How much the reconstructed signal resembles the original signal depends on the sampling rate, the quantization method and the reconstruction algorithm used. The theory of signal reconstruction is quite extensive and goes beyond the scope of this thesis. A good introduction can be found in [11].

6.2.3 Mixing Audio Signals

When using VOIP in virtual environments, there is another thing that we must take into account. Each participant will send its own digitized voice signal which will be received by a number of other participants. If two or more persons are talking at the same time, their signals will have to be mixed somehow.

Luckily this is very simple: physics teaches us that for sound waves, the principle of superposition applies. This principle states that when two waves overlap, the amplitude of the combined wave at a specific time can be obtained simply by adding the amplitudes of the two individual waves at that time [38]. Practically speaking this means that we merely have to take the sum of the digitized versions of the signals.

6.3 Communication Requirements

Nowadays everybody is used to telephone quality voice which typically has very few noticeable errors and low delay. Also, when using the telephone system there is no such thing as variation in delay. With pocket-sized voice however, each packet will typically arrive with a slightly different amount of delay, resulting

in jitter. There is also no guarantee about delay caused by the network and in general, some packets will contain errors on arrival or will not even arrive at all.

In this section, we will see what the requirements are for decent voice communication. With `decent communication' a form of conversation is meant which does not cause irritation with the participants.

6.3.1 Error Tolerance

In contrast to data communication, where even the smallest error can cause nasty results, voice communication is much more tolerant to the presence of errors. An occasional error will not seriously disturb the conversation as long as the error does not affect a relatively large portion of the signal.

6.3.2 Delay Requirements

When you are using data communication, it does not really matter how much delay there is between the sending of a packet and its arrival. With voice communication however, the overall delay is extremely important. The time that passes between one person saying something and another person hearing what was said, should be as low as possible.

Studies show [10] that when the delay exceeds 800 ms, a normal telephone conversation becomes very hard to do. They also show that a delay of 200 to 800 ms is tolerable for short portions of the communication. However, in general a delay below 200 ms has got to be attained to hold a pleasant conversation.

6.3.3 Tolerance for Jitter

If each block containing a part of a digitized voice signal would be played immediately on arrival, the quality of the communication would be rather low.

Since each block typically arrives with a slightly different delay, sometimes a block would be played before the previous one was finished and sometimes there would be a small gap between the end of one block and the next. Since this jitter is not something that occurs for short periods, but continuously, this will be very annoying to the participants of the conversation.

6.4 Communication Patterns

In a conversation between two persons, it very unlikely that both are always talking at the same time. Usually, when one person is speaking, the other one listens, possibly giving short affirmations. The same principle applies when a group of people is holding a discussion: when one person is talking, the other ones listen.

It is because of this pattern that a normal telephone call wastes a large amount of bandwidth. When someone is not speaking, the bandwidth stays assigned without being used. With pocketsize voice this bandwidth could be used by other calls or applications.

Several speech models are presented in [10]. Such models could be used to predict arrival patterns of packets containing voice data. These predictions in turn could be used to create a network design with a more effective utilization of resources. Although these models are obviously important, I feel that they are beyond the scope of this document [10].

6.5 Impact on VOIP

We have just seen some aspects of voice communication. In this section the importance of these aspects for VOIP is described.

6.5.1 Sampling Rate and Quantization

At the start of the chapter we saw what sampling and quantization is. It was also mentioned that a higher sampling rate and a smaller quantization step implied a better representation of the original signal. But this also means that more digitized information will have to be transmitted and more bandwidth is required. So we have to determine how much information is necessary to hold a telephone quality conversation.

The Nyquist theorem is important in this matter. It states that the minimum sampling frequency should be twice the maximum frequency of the analogue signal [26]. This is in fact quite logical: to be able to capture N cycles, you have to take measures at least 2N points along the signal. Otherwise it would be impossible to capture the maxima and minima of the signal.

The speech signals that humans produce can contain frequencies of even beyond 12 kHz [10]. However, in the telephone system, only frequencies below 4000 Hz are transmitted and this still allows high-quality communication. Using this information together with the Nyquist theorem suggests that a sampling rate of 8000 Hz is adequate for the digitization of speech.

To cover the range of amplitudes that a voice signal can produce, at least twelve bits are needed when a uniform quantization scheme is used [23]. However, a uniform scheme which reduces each sample into an eight bit value is usually good enough to attain telephone quality conversations.

As for logarithmic quantization, there are two schemes that are also worth mentioning at this point. In the United States and Japan, m-law encoding is the

standard for transmission over networks. It reduces a thirteen bit uniformly quantized signal into an eight bit value. In Europe, A-law is used. This scheme does the same but starts with a twelve bit uniformly quantized value. [23]

From this information we can calculate the required bandwidth for a telephone quality conversation. Above was explained that we needed a sampling rate of 8000 Hz and that an eight bit value is usually used to store a sample. So we will be transmitting 8000 eight bit values each second. This requires a bandwidth of at least 64000 bps or 64 kbps.

6.5.2 Packet Length

With Voice over IP, packets can get corrupted or even lost. To reduce the amount of lost information, a packet should contain only a very small amount of the voice signal. This way, if a packet gets lost, only a tiny fraction of the conversation will be missing, which is very unlikely to disturb the conversation.

It is important to note that this argument is made from the point of view of the conversation. We will see later on that from the transmission's point of view, an argument can be made in favor of larger packets. Somehow, a compromise will have to be made.

6.5.3 Buffering

A simple technique to avoid jitter in the playback of a voice signal is to introduce an amount of buffering, as figure 6.2 illustrates. Instead of playing the voice data of an incoming packet as soon as the packet arrives, a small amount of delay is introduced. Because this is done for all packets, there is a higher probability that when one packet has been played, the next one is immediately available.

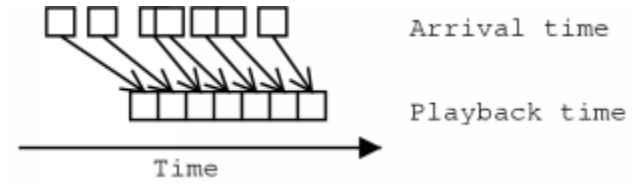

Figure 6.2 - Buffering to avoid jitter

In practice, normally only a small amount of buffering is needed. In the applications I developed, I have used jitter calculations to determine the amount of buffering needed. The amount of jitter is usually not very high and accordingly only a small amount of buffering is done. This method appears to have good results.

6.5.4 Delay

A large delay is disastrous for a conversation. The total delay can be categorized into two types [42]. The first type is fixed delay. This is the total delay which is always present due to buffering, link capacity etc. The second type is variable delay. This is the delay component which is caused by packet queuing in routers, congestions etc.

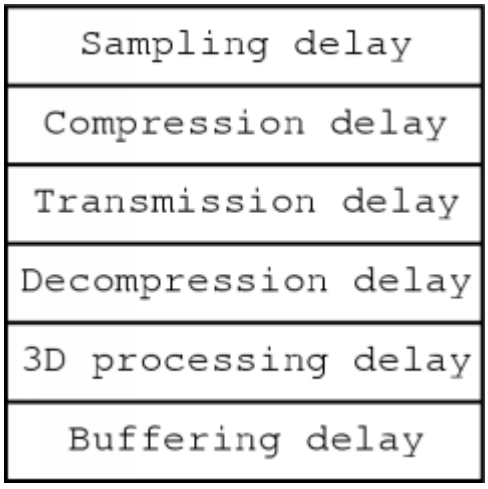

Figure 6.3 - Types of delay

The components shown in figure 6.3 determine the amount of fixed delay for a VOIP system in virtual environments. For `normal' VOIP, the 3D processing component can simply be left out. The **sampling delay** is the delay introduced by the sampling of the voice signal. For example, if the sampling interval is one second, the total delay will be at least one second long since the digitized voice signal cannot be processed before the data is collected. This means that from the point of view of the delay, the sampling interval should be kept as small as possible. Again, from the point of view of the transmission, an argument will be made in favor of larger sampling intervals. The **compression** and **decompression delays** are introduced by compression and decompression algorithms respectively. The **transmission delay** is the delay which is present due to link capacities. The **3D processing delay** is the delay caused by the algorithms which generate the three dimensional sound for use with virtual environments. Finally, the **buffering delay** is the delay which is artificially introduced to compensate for jitter. This delay could be set manually or determined automatically, like it is done in my own VOIP applications.

6.5.5 Silence Suppression

We saw earlier that in a conversation, there is usually only one person speaking at a time. In pocketsize voice, this gives us the opportunity to save bandwidth because packets containing only silence do not need to be sent. However, before we can discard packets, we must first be able to determine whether they contain silence or not. One way this could be done is by calculating the amount of energy of the voice signal in a packet. Packets which do not contain a sufficient amount

of energy are assumed to hold only silence and can be discarded. A simpler technique which I have used is to check a packet for samples with a value above a certain threshold. If no such samples exist, the packet is considered to hold silence and can be discarded. This method has proven to be simple and effective. Silence suppression does have a minor side effect. Because the `silent' packets are discarded there is absolutely no sound at all at the receiving side, not even background noise. This is truly a deadly silence and it might even seem that the connection has gone. A solution is to artificially introduce some background noise at the receiver side.

7

The Internet Protocol

7.1 Introduction

Before we can really talk about Voice over IP, it is necessary to explain what IP is. The abbreviation IP stands for Internet Protocol. Version four is currently most in use and it is common to use the term `IPv4' to indicate this version of the protocol. When no version number is mentioned, usually the discussion is about version four. This is also the case in this thesis.

The Internet Protocol is covered in this chapter. It begins with a discussion about network software architecture, followed by a description of the workings of IP. We will also see some characteristics of IP networks and I will describe the most used protocols which run on top of IP. Afterwards, some reasons will be given for the use of IP for voice communication. Finally, the chapter contains an overview of IPv6, the new version of the Internet Protocol. The information in this chapter was mostly obtained from [34], [9] and [41]. The official specification of IPv4 can be found in [19].

7.2 Network Software Architecture

Nowadays, network software is usually very structured. This section is about the way this software is organized. It also contains a discussion about the OSI reference model, which is a good example of this structured design, and about the TCP/IP reference model, in which - as the name suggests - IP plays a very important role.

7.2.1 Layered Design

To facilitate the design of network software, usually the approach of a `layered design' is used. In this approach, each layer provides a certain functionality,

which can be used by the layer directly above. There are several advantages to this approach.

First of all, the software is much easier to design. Trying to implement the desired functionality all at once will be very difficult and will probably result in many flaws in the program. Furthermore, these flaws will be difficult to track. By dividing the software in layers, you only have to worry about implementing some functionality for each layer. This does not mean that is will be an easy task, but by using a structured approach you will be able to tackle it more efficiently. Another advantage is the adaptability. If you want to make some changes to the software, for example to correct a flaw or to improve an algorithm, you will only have to change the relevant layers if the interface with the layer above stays the same. Closely related to this is portability. If the layers are well designed, only a few of them will have to be changed to be able to use the software with other networking hardware or on another operating system. Finally, since many layers will probably be implemented as part of the operating system itself, the end-user applications do not have to contain those layers. This way, the size of those applications can be reduced. To make communication between two hosts possible, they have to be connected to some kind of physical medium. All data will be sent over this medium, but only the lowest layer will have direct access to it. Conceptually, however, two layers on different machines but at the same level can be thought to communicate directly. The rules and conventions that are used in this communication are contained in the protocol for that level. The whole set of protocols is often referred to as the protocol stack. Figure 7.1 illustrates all

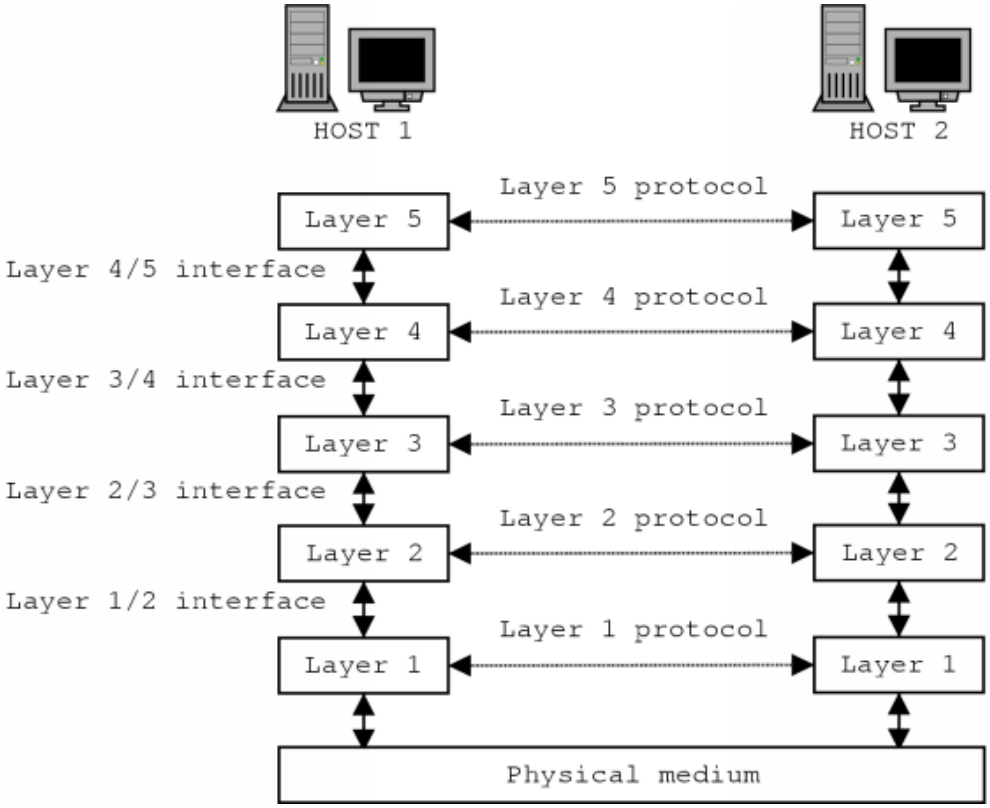

Figure 7.1- Example of layered design

When a layer wants to transmit some data to its corresponding layer at another host, it uses the functionality of the layer below to do this. That layer adds some control information, usually in the form of a header, to the data and uses the layer below to transmit the data. The whole process keeps repeating itself until the data is finally sent over the physical medium. When the data reaches the receiver, the first layer processes the control information and passes the data to the layer above. At each layer, this process then repeats itself.

7.2.2 OSI Reference Model

The Open Systems Interconnection (OSI) reference model is a model with seven layers which was developed by the International Standards Organization (ISO). The model only specifies what each layer should do, without going into any detail

about, for example, the protocols that should be used. In actual implementations it turns out that some of the layers are almost empty and others are too elaborate. However, conceptually the model is quite nice and it is a good example of layered design.

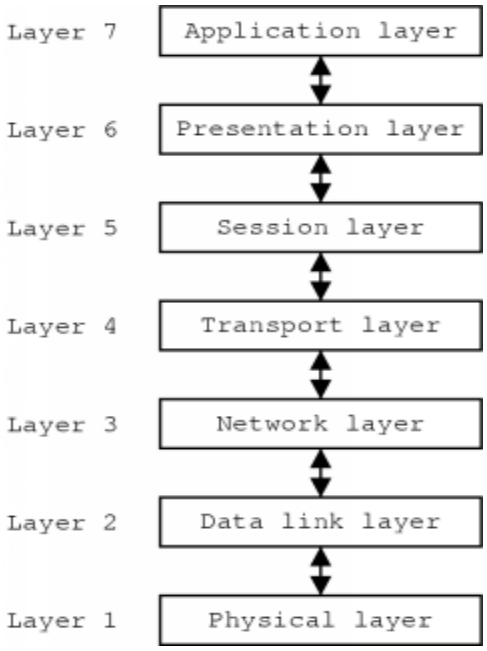

Figure 7.2 - OSI seven layer model

7.2.2.1 The Physical Layer

The physical layer is the lowest layer in the model and this is the only one which has immediate access to the communication medium. It is responsible for the transfer of bits from the source to a destination which is connected to the same medium.

7.2.2.2 The Data Link Layer

The data link layer uses the facilities of the physical layer to create a more reliable communication channel. This layer makes it possible to send blocks of data, called frames, reliably from one host to an adjacent one.

7.2.2.3 The Network Layer

So far, the layers have only been concerned with transporting information between hosts connected to the same medium. The network layer's function is to make it possible to send packets to a host that does have a connection to the sender, but is not connected to the same physical medium.

This means that between the different physical media, there have to be devices which transfer data from one medium to another. These devices are usually called routers or gateways. The use of such devices makes some extra work for the network layer necessary.

First of all, it is possible that between a certain source and destination there exist several possible routes. The network layer then has to determine which one to choose. These routes can be determined in advance but it is also possible that the network layer dynamically adjusts the routing information to achieve better performance.

Second, since the flow between adjacent networks can get very large, it is possible that a router cannot cope with all that traffic. The router then becomes a bottleneck for the data flow. The network layer tries to control such congestions.

7.2.2.4 The Transport Layer

The previous layer made it possible to actually send data from source to destination. In that layer communication is done by exchanging packets. The transport layer makes it possible to consider the data as a stream of bytes, and not in terms of packets. The layer itself will divide the data in smaller units and hand it over to the network layer. If some packets get lost, the layer handles this

and the receiver will still receive the correct stream of bytes. To be able to keep track of which data has already been sent and which not, the transport layer uses a connection-oriented approach.

The transport layer will also have flow control mechanisms, to prevent the flooding of a slow receiver, and congestion prevention mechanisms. Note that the network layer also has congestion control functionality. However, the best way to handle congestions is to prevent them from happening in the first place. This is what the transport layer does.

This layer is the first true end-to-end layer. The physical and data link layers were only able to communicate with an immediate neighbor. The network layer actively had to transport the packets step by step from source to destination. In this layer however, the underlying topology is transparent to its user.

7.2.2.5 The Session Layer

The session layer makes it possible to establish sessions between two hosts. A session extends the capabilities of the transport layer with some extra services.

An example of such an extra service is synchronization. During a transfer there would be certain synchronization points. If the data transfer would be interrupted due to an error, the transfer could be restarted from the last synchronization point rather than starting the transfer all over again.

7.2.2.6 The Presentation Layer

The presentation layer takes the type of information which is being transferred into consideration. This layer could, for example, make the necessary

transformations if one computer is sending ASCII characters and the other one is sending Unicode characters.

7.2.2.7 The Application Layer

Finally, the highest layer in the model is the application layer. This is the layer in which most end-user networking applications reside. To communicate, such programs mostly use their own protocols. Examples of such applications are applications for file transfer and applications which represent a virtual terminal.

7.2.3 TCP/IP Reference Model

The Internet Protocol is a protocol which is used in the TCP/IP model. The TCP/IP model was originally designed for use on the ARPANET, a military network in the late 1960s. It is, in fact, this network which grew out to become the Internet as we know it today.

Because of its military background, there were two major requirements for the model. The first was robustness. The US Department of Defense (DoD) wanted to make sure that communication was still possible even if some routers or lines went down. The second requirement was interoperability. Since there were different types of hardware involved, for example copper wires and satellites, the DoD wanted a set of protocols which could not only handle these types of hardware separately, but which would also make it possible to connect them.

Compared to the OSI model there is a big difference in the way that the model came to existence. The OSI model was first carefully designed, and later protocols were designed to fit the model. This makes the OSI model a very general one. The TCP/IP model, however, originated in the opposite way. First

the protocols were designed to meet the requirements of the DoD. Later, these protocols were described and it is this description which is the reference model. This means that the TCP/IP model does not really fit anything else but TCP/IP networks. Another point about TCP/IP is that the layered design is not followed very strictly. There are some violations to this principle in the model.

Despite of these arguments, the TCP/IP model has become very popular and very widely used. In contrast to the OSI model which has seven layers, the TCP/IP model only has four, as figure 7.3 shows. Here is a description of these layers.

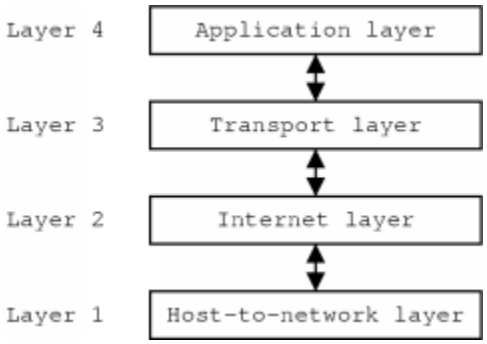

Figure 7.3 - TCP/IP four layer model

7.2.3.1 The Host-to-Network Layer

The host-to-network layer is the lowest layer of the model. Sometimes it is also called the link layer or the network interface layer. There is in fact little to be said about this layer. The only requirement which is given by the model is that this layer should be able to transmit and receive the IP datagrams of the layer above over the network. The layer has somewhat the same function as the physical and data link layers in the OSI model. This means that this layer usually is only able to send data to hosts which are connected to the same medium.

7.2.3.2 The Internet Layer

The internet layer corresponds to the network layer in the OSI reference model. Its job is to bring packets from source to destination, across different types of networks if necessary. There are, however, no guarantees that the packets will arrive or that their order will be preserved. The service that this layer offers is therefore called a best-effort service. There is no notion of a connection in this layer. The packets which are exchanged are called Internet Protocol datagram or IP datagrams and the protocol which is used is called the Internet Protocol or IP. The datagrams consist of a header and the actual data. The header will be described later on.

Like in the OSI network layer, intermediate devices called routers, are needed to make transmission of data across different types of networks possible. The IP datagrams can then be sent from source to destination, on a hop-by-hop basis. Again, like in the OSI network layer, this also means that routing algorithms and congestion control are important aspects of the internet layer.

7.2.3.3 The Transport Layer

To make sure that multiple applications can use the network facilities at once, some extra naming mechanism is needed. The internet layer does contain a naming mechanism to identify different hosts, but there still has to be some way to differentiate between the processes which are using the network. This is done in the transport layer by the use of a port number. This layer has somewhat the same functionality as the transport layer in the OSI model. Here also, the transport layer is the first real end-to-end layer.

The TCP/IP model has two major transport layer protocols. One of them is the Transmission Control Protocol (TCP). This protocol transforms the connectionless unreliable packet based service of the internet layer into a connection-oriented reliable byte stream. It is a very important protocol since it makes reliable communication possible. This is why its name is also in the name of the reference model.

The other protocol is the User Datagram Protocol (UDP). This is a protocol for applications which do not need the service offered by TCP or want to use a protocol of their own. The User Datagram Protocol is merely a small extension to IP. It is also an unreliable packet based connectionless protocol and the only real extensions to IP itself are the presence of a port number and an optional checksum of the data.

7.2.3.4 The Application Layer

Like in the OSI model, the application layer contains the protocols of networking applications. Among these are virtual terminal applications (TELNET protocol), file transfer utilities (FTP protocol) and electronic mail (SMTP protocol).

7.3 How IP works

Let us now take a closer look at the Internet protocol itself and how it makes communication between two hosts possible. First I will give a description of the IP packet format. Next, the addressing mechanism used by IP is discussed. We will then take a closer look at how packets are routed from source to destination. Finally, an explanation is given of multicasting, a technique which allows us to save bandwidth when the same data has to be sent to multiple destinations. This

is, of course, a very interesting feature when using VOIP in virtual environments, since there will typically be many receivers for each talking participant.

7.3.1 Packet format

Any packet sent by the IP layer consists of an IP header, followed by the actual data. The format of the IP header is shown in figure 7.4. The most significant bit is the one at the left, numbered zero. The least significant bit is the one at the right, numbered thirty-one. Transmission is done in network byte order, also called big endian format. This means that in each 32-bit word the most significant byte is sent first and the least significant byte is sent last.

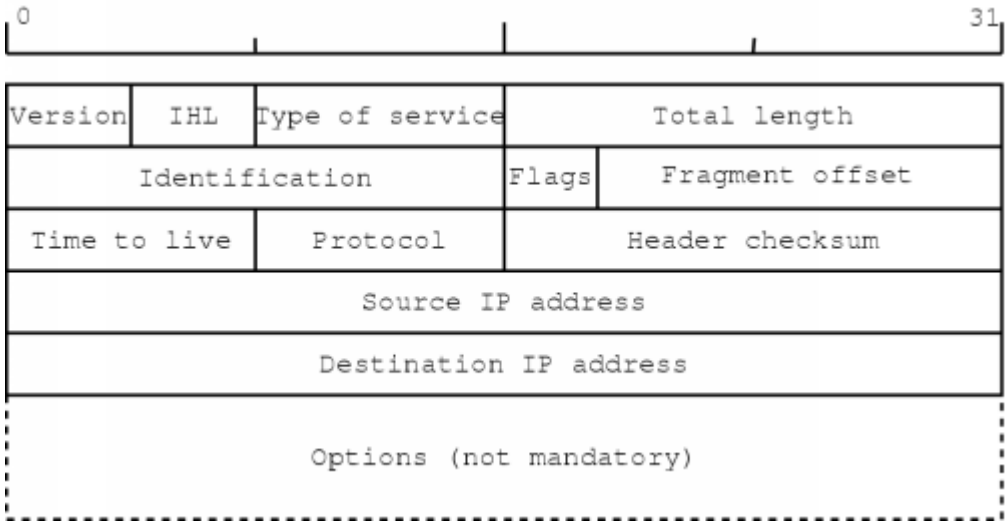

Figure 7.4 - IP header format

The **version** field should contain the value `four' for the current version of the Internet Protocol. This field can be used to let different versions coexist, something which will make the transition to a new version much easier.

The **IHL** field contains the `Internet Header Length'. This specifies the length of the header in 32-bit words. Since it is a 4-bit value, the maximum length of the header will be sixty bytes. Also, since the mandatory part of the header consists

of five words, the smallest legal value is five. The specification in 32-bit words also has as a consequence that the header must end on a 32-bit boundary, so it is possible that some padding is required if options are present.

The next field is the **Type of service** (TOS) field. This field was meant to supply a quality of service (QoS) mechanism, but in practice it is rarely used. However, since voice data has real-time aspects, it may be necessary to pay attention to it if we want to keep the end-to-end delay in the communication low.

An overview of the TOS field is depicted in figure 7.5. The contains a three-bit precedence field which specifies the priority of the packet. A value of zero indicates a normal priority and a value of seven indicates the highest priority. Following the precedence field, there are three bits which stand for delay, throughput and reliability. Only one of the bits can be set to one. The last two bits in the field are currently unused and should be zero.

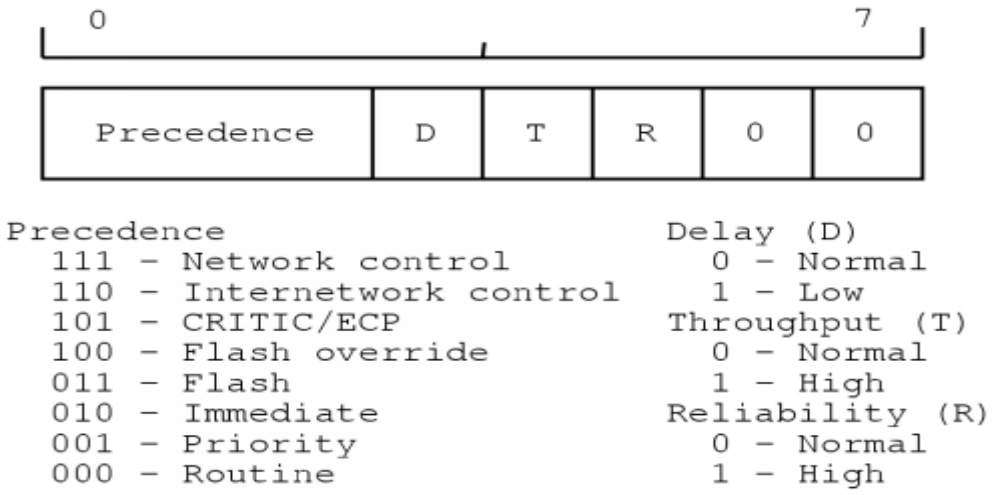

Figure 7.5 - The TOS field

The size of the IP datagram is specified in the **Total length** field. It is a 16-bit field, so the maximum size is 65535 bytes. Most networks cannot handle this size

so usually it is much less. All hosts are, however, required to be able to send and receive datagrams with a length of 576 bytes or less.

During the transmission of a packet it is possible that it has to traverse different kinds of networks. Each network has its own Maximum Transfer Unit (MTU) which specifies the maximum frame size it can handle, including the link layer header and trailer (if present). This means that there is always a possibility that the datagram, as it passes over the different networks, cannot be transmitted over a certain network. It then has to be fragmented and each piece has to be sent separately.

The **identification** field is an aid in reconstructing fragmented datagrams. Each datagram fragment will have the same value in this field. When sending IP datagrams, a host typically increments this field for each datagram sent.

Next, there are 3 **flag** bits, of which the first one is reserved and should be zero. The next one stands for `don't fragment' (DF) and the last one stands for `more fragments' (MF). If a datagram cannot be transmitted across a network because it is too large and the DF bit is set, an error will be sent back to the sender. All but the last the fragment of the original datagram will have the MF bit set.

Using the **fragment offset** field, the internet layer can reassemble fragmented datagrams. This 13-bit value specifies the offset of the fragment in the original datagram. The offset is given in units of 64-bit words.

The **time to live** (TTL) field is used to limit the lifetime of a datagram. In theory the value specifies the number of seconds the datagram is allowed to exist. There is also the requirement that each router must decrement the value by at

least one. If the packet stays a long time in the queue of the router, the TTL value should be decreased with the number of seconds the datagram spent in queue. When the counter is zero, the datagram must be discarded. In practice, the value is just decremented at each router, which makes the field a hop counter.

The **protocol** field is used to specify to which protocol the data in the datagram belongs. This can be a transport layer protocol, but it can also be one of the control protocols of the internet layer.

The **header checksum** is used to check the validity of the datagram. Note that the checksum is only for the header, so higher level protocols will have to use their own checksums if they want to make sure their data is valid.

Finally, the minimal header contains the **source IP address** and the **destination IP address**. These addresses must be included in each datagram since the internet layer operates in a connectionless way. Each datagram is sent separately and therefore each datagram must contain not only its destination but also its source, in case an error has to be reported. The format of the addresses is described further on.

The **options** section can be used to record the route a datagram follows, possibly with timestamps. Another option is source routing, where you can specify the route a datagram should follow.

7.3.2 Addressing

Every host on an interconnection of networks - or internet - which uses IP, should have a unique IP address. An IP address is a 32-bit value and the complete

address space is divided into five classes, named class A to class E. The way these classes are represented is shown figure 7.6.

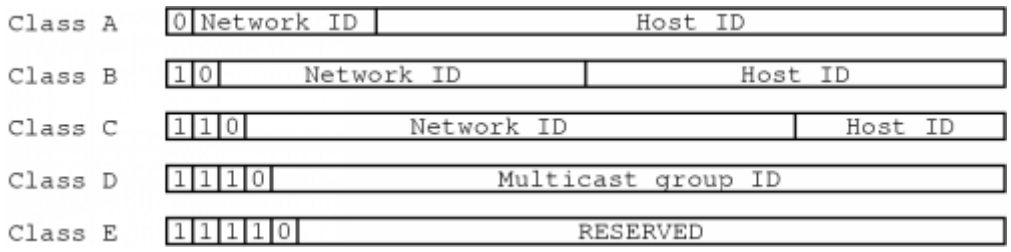

Figure 7.6 - Classes of IP addresses

The way an address is usually written, is in its dotted decimal form. To obtain this the 32-bit value is split in four 8-bit values. These four values are then written in decimal form, separated by dots.

The first three classes contain the addresses which can be assigned to hosts. Not all possibilities are allowed though; there are some reserved addresses. First of all, a host ID with value zero does not specify a host, but the network on which hosts with the specified network ID are located.

If the host ID is the highest possible value for its class (all one bits in binary format), the address is a broadcast address for a certain network. This means that if you send IP datagrams to that address, they are delivered to all hosts on that network.

When the network ID of an address is zero, it specifies the local network. This type of address is only used in initialization procedures, when the local network ID is not known.

Other reserved addresses are 0.0.0.0 and 255.255.255.255. The first of these specifies the local host on the local network. It is also only used in initialization

procedures. The second address is the so-called limited broadcast address. This specifies a broadcast to all hosts on the local network.

Of the remaining two classes, only class D is actually used. Class E was meant for future use. Class D specifies a multicast address. Multicasting allows data to be sent to a group of hosts. This means that when you send an IP datagram to a multicast address, the datagram is sent to all hosts in the corresponding multicast group. Multicasting is explained in more detail later.

7.3.3 Routing

The internet layer uses the link layer to actually transmit its data. The link layer, however, can only deliver this data to hosts which are connected to the same medium. To be able to send this data across several networks, routers are used. These devices connect to several networks and make sure that incoming IP datagrams are forwarded to the appropriate network. We will now take a closer look at how this process works. Note that only the basic mechanisms of routing are explained here.

When the internet layer of the sending host has to transmit a datagram to a certain destination, it first examines the destination IP address. This is necessary because the internet layer has to tell the link layer to which machine the data has to be sent. If the destination IP address is on the same network, the machine which will receive the datagram will simply be the destination for the transmission.

If the address does not specify a host on the local network, the internet layer examines its routing table. The entries of such a routing table can be seen as

pairs of a destination address and a router address. The destination address can be an address of a host or of a network.

The internet layer then starts looking for a router to send the datagram to. To do this, it compares the destination address of the datagram with the destination addresses in the routing table. If no complete match can be found, it checks if a matching network entry can be found. If not, it uses a default entry. If an entry was found, the internet layer takes the corresponding router address and tells the link layer to send the datagram to that address.

For example, consider a host with IP address 199.198.1.10 who wants to send a packet to 199.198.2.100. This destination host is not on the same network, so the internet layer of the sender will consult its routing table. Suppose that the table looks like

Table 7.1 – Routing Table

Destination	Gateway
199.198.5.10	199.198.1.251
199.198.2.0	199.198.1.252
default	199.198.1.253

The internet layer first looks in the table for a complete match for address 199.198.2.100. It finds no such match, so it will check for a matching network address. This time, it does find a matching entry: the second one describes the network on which the destination host is present. The internet layer then takes

the corresponding gateway entry - address 199.198.1.252 - and sends the packet to that router (gateway).

When the datagram reaches the router, it is passed on from the link layer to the internet layer. The internet layer then follows almost the same procedure to search for a destination machine to forward the datagram to. The only difference is that the router will usually be connected to several networks and this means that the appropriate interface to transmit the data also has to be chosen. The whole procedure is repeated until the datagram reaches its final destination.

To make sure good routes are chosen, many routers communicate with each other. They exchange their routing information and based upon this information each router updates its routing table to contain the best known route for each destination. The type of information and the way it is exchanged are determined by the routing protocol which is used. Examples of routing protocols are the Open Shortest Path First (OSPF) protocol and the Border Gateway Protocol (BGP).

7.3.4 Multicasting

Basically, there are three transmission modes that can be used when sending an IP datagram. They are called unicast, multicast and broadcast. Unicasting simply means sending a datagram from a source to one destination. The term broadcasting is used when you want to send a datagram to all hosts on a specific network. When you want to send a datagram to an arbitrary set of hosts, it is called multicasting.

A simple way to implement multicasting would be to unicast a copy of the datagram to each destination. This method obviously wastes a lot of resources. A better way would be to transmit one datagram which is copied only at points where it needs to follow different routes to reach its destinations. This is the way it is done on IP networks.

To be able to receive datagrams directed to a certain multicast address, a host must first join the multicast group associated with that address. Similarly, when it no longer wants to receive those datagrams, it leaves the multicast group. This group management is done according to the Internet Group Management Protocol (IGMP), which is formally specified in [18].

In general, the protocol works as follows. Each host maintains a list of multicast groups from which it wants to receive datagrams. Multicast routers periodically broadcast IGMP queries on the networks to which they are connected. The hosts then send IGMP replies, containing the groups in which they are interested.

Once these replies have been gathered using IGMP, multicast routers exchange this data with each other and use all this information to build their routing tables. When they receive a multicast datagram, they can then determine to which hosts and multicast routers the datagram should be sent.

7.4 Characteristics of IP Networks

When datagrams have to travel across several networks, they will also need to pass through a number of routers. Each router has to examine all incoming packets and this will introduce a certain delay in the communication. Studies even show that the time it takes for a packet to reach its destination is much

more affected by the number of hops the packet makes than the actual geographical distance covered [41].

When a router gets too heavily loaded, some packets will have to be discarded. This packet loss is usually bursty. This means that for a short period of time several consecutive packets will be lost.

Routers communicate with each other to dynamically adapt their routing tables to the current state of the network. This means that datagrams going to the same destination can sometimes follow different routes. Although it turns out that routes do not change very often during a transmission, it does happen. Such a change can cause datagrams to arrive out of order.

Besides packet loss and out-of-order arrival of packets, it can also happen that a datagram gets duplicated during its transmission. This will cause two or more identical datagrams to arrive at the destination, possibly with some delay between them.

Finally, another important feature of IP networks is the fact that when a source sends datagrams to a certain destination, the amount of time to reach the destination will differ for each datagram. This is usually called inter arrival delay, inter arrival jitter or simply jitter.

7.5 Higher Level Protocols

The two most common transport level protocols in the TCP/IP architecture are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Each of these protocols offers a specific kind of service which applications can use to communicate across networks.

7.5.1 TCP

Currently, TCP is undoubtedly the most used protocol of the two. This protocol transforms the unreliable packet-based service of the internet layer into a reliable byte stream. The protocol is designed for communication between two hosts, so it only supports unicasting.

To offer this kind of service, the TCP module has to do a lot of work. First of all, a connection has to be set up, and this has to be done in such a way that it is more or less safe: the module must make sure that connections cannot be established accidentally - for example because of duplicate packets.

The incoming stream of bytes then has to be split up at the side of the sender and the stream has to be reconstructed at the side of the receiver. Care must be taken to discard duplicate datagrams and to correct their arrival order if necessary. There must also be some kind of mechanism to cope with lost packets.

All this is handled quite effectively. To establish a connection the TCP module uses a handshake mechanism, called a three-way handshake. Duplicate and out-of-order datagrams are handled by using sequence numbers. Finally, lost packets are handled by an acknowledgement mechanism: all bytes of the stream have to be acknowledged by the destination. If the source did not receive an acknowledgement after a certain amount of time, it sends the necessary data again. The protocol also specifies flow control mechanisms, which prevent the swamping of a slower receiver, and congestion control mechanisms, which try to avoid congestions.

Note that the exact way in which the TCP module works is a lot more complicated than this explanation makes it seem [36].

7.5.2 UDP

Applications which do not require the functionality that TCP provides, can use UDP. To transmit data, the UDP module simply passes a UDP header followed by that data to the internet layer which then sends the datagram on its way. This means that just like IP itself, UDP is a best-effort service. No guarantees about delivery are given, datagrams can get reordered and datagrams can be duplicated. The exact specification of UDP can be found in [39].

The UDP header is shown in figure 7.7. The header contains the source and destination ports, which identify the sending and receiving applications. Next, it contains the number of data bytes which must be sent and finally the header contains space for an optional checksum.

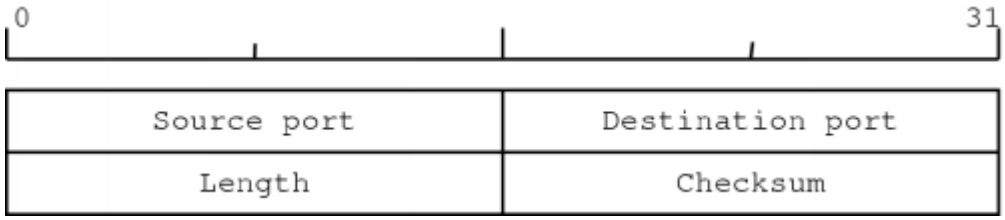

Figure 7.7 - UDP header

Since the service which UDP offers is almost identical to the service of IP itself, it is possible for applications to send UDP datagrams to a multicast address and to receive UDP datagrams from a multicast group.

7.6 Why use IP?

Delivering speech information in packets has some advantages to the classical telephone system. When you make a `normal' telephone call, a path is set up

between you and the destination of the call. You will then have a fixed amount of bandwidth you can use during the whole call.

The major advantage of that approach is that you will have some guarantees about the QoS, since you are certain to have a specific amount of bandwidth available. But this way, a lot of bandwidth is also wasted, because during a conversation there are a lot of silent intervals for each person.

Using VOIP, those silent intervals can be detected. The VOIP application can examine each packet and detect whether it contains speech information or only silence. If the latter is the case, the packet can simply be discarded.

Another advantage is the possibility of compression. With the compression methods available today, it is possible to reduce the requirement of 64 kbps for uncompressed telephone-quality voice communication to amounts which are far lower. However, a high compression ratio often means that the voice signal will be of lesser quality. We will go deeper into the domain of compression in one of the next chapters.

So packetised voice has certain advantages to the classical telephone system. But IP is not the only packet-based protocol. Why exactly should IP be used? This protocol was designed mostly for data transport, and it has only limited QoS support. The main reason IP is so important is because of its omnipresence. The TCP/IP architecture has proved to be very popular and nowadays it is very widely used. This fact gives IP a great advantage over other protocols. Alternatives for packetised voice include Voice over Frame Relay (VoFR) and Voice over ATM

(VoATM). Both allow better support for real-time traffic than an average IP network. However, these technologies are not used as widely as IP.

7.7 IPv6

With the growth of the Internet - on which IP is used - it has become clear that the current version of the Internet Protocol has some shortcomings. For this reason a new version of the protocol has been devised, now called IP version six, or just IPv6.

This section contains a brief description of the protocol, which was introduced in [20] and later redefined in [21]. The latter reference is the source of the information in this section.

7.7.1 Reasons

Because of the enormous growth of the Internet, there will soon be a shortage of IP addresses. The current version uses 32-bit values, which can provide enough IP addresses in theory. However, because of the subdivision in classes and the way addresses are allocated within those classes, in practice there are far less addresses available. This lack of addresses was one of the most important reasons for the development of a new version.

Other reasons were the need for better QoS support and better support for security. Also, it turned out that some features of IPv4 were hardly ever used and bandwidth and processing time could be saved by redesigning the protocol. Finally, because the routing tables in routers kept growing, the reduction of their sizes was also an important reason for the design of an improved protocol version.

7.7.2 Description

Let us now take a closer look at this new protocol. Firstly, the format of the IPv6 header was described. Next, it will be compared to IPv4.

7.7.2.1 Header

The IPv6 header is shown in figure 7.8. In this version, the header has the fixed size of forty bytes.

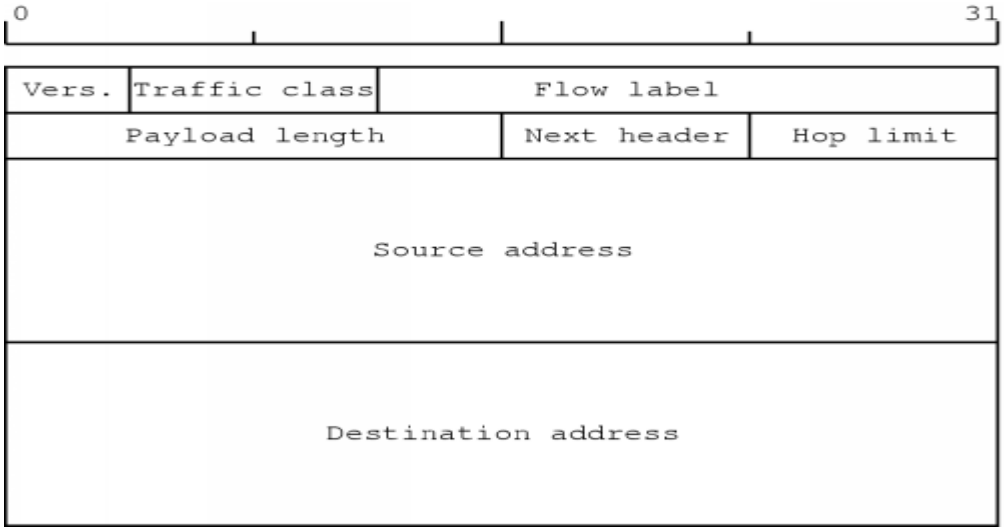

Figure 7.8 - IPv6 header

The **version** field contains the value six. This way, the version of the protocol can be detected and IPv4 and IPv6 can coexist. This will make the transition to the new version easier.

The **traffic class** has somewhat the same function as the TOS field in the IPv4 header. Using this field, one could specify the type of traffic this datagram belongs to. This could then allow appropriate handling of the datagram.

A flow is defined as a sequence of datagrams which are sent from a certain host to a receiver or - in case multicasting is used - to a group of receivers, and for

which the sender desires special handling by the routers along the way. The **flow label** field can then be used as an identifier for such flows.

The number of data bytes following the header is specified by the **payload length** field. This is a 16-bit wide field, so the maximum number of data bytes in a datagram is 65535. However, it is possible to create larger datagrams than this field allows. How this can be done is explained further on.

The **next header** field specifies of what type the header following the IPv6 header is. In the simplest case, this is a header from a higher level protocol. But it can also be one of the extension headers which IPv6 defines. It is because of these extension headers the IPv6 header is somewhat simpler than the header of IPv4. Some fields in the IPv4 header and the different options are now used through extension headers.

Several extension headers are defined. Fragmentation, security, authentication, source routing and many more are all made possible through these extension headers. For a complete description you should consult [21].

Earlier, I mentioned that the payload length of 65535 can be exceeded. Well, this can be done using an so-called `hop-by-hop' extension header. This header has an option called `Jumbo Payload' and allows lengths greater than 65535 to be specified. Such datagrams are often called `jumbograms'.

The **hop limit** field is a replacement for the TTL field in the IPv4 header. This field limits the lifetime of a datagram by requiring that the value in the hop limit field must be decremented by one by each node that forwards the packet.

Finally, the header contains the **source address** and the **destination address** for the datagram, which are 128-bit values.

7.7.2.2 Important changes from IPv4

First of all, there is the larger address space. The 128-bit values should be enough to continue for quite some time. On the entire planet, these addresses would allow for 7×10^{23} addresses per square meter [9].

Furthermore, because of the way multicast addresses are represented, the scalability of multicast routing should be improved. Also, a new type of transmission, called `anycasting', is available. This type of transmission is used to send a datagram to anyone of a group of receivers.

The header format is simpler than it was the case with IPv4. The IPv6 header has only eight fields, whereas the IPv4 header had at least twelve fields. This allows for faster processing of datagrams. The extension headers give the protocol great flexibility, certainly compared to the limited IPv4 options field.

The concept of a flow is also new to this version. This makes it possible for a certain stream of data to receive special treatment. This feature could prove to be useful for real-time services for example.

Finally, the added support for authentication and security are definitely an important improvement over version four.

8

Transmission of Voice Signals

8.1 Introduction

We now know how to send voice information without wasting a lot of bandwidth so we can move on to issues relating to the actual transmission of the speech data. The Internet Protocol only offers a best-effort service without any QoS guarantees. For decent voice communication it is, however, necessary to have certain guarantees since too much delay or too many lost packets will seriously affect the quality of the conversation.

This chapter discusses how we can transmit voice information while preserving the communication quality. First we will talk about some general requirements. Next, we will see what protocols can be used to transmit the speech data. Also, some resource reservation methods will be discussed since this can help to improve the transmission quality. Finally, we will discuss the transmission delay.

8.2 Requirements

When transmitting packets containing voice data, there must be some mechanism to preserve synchronization within the speech signal. The consecutive packets should be played at the right time, in the right order. This type of synchronization is called intra-media synchronization.

We have seen in chapter three that for real-time voice communication, the overall delay has to be kept as low as possible. Since an IP network in general only offers a best-effort service, there is no guarantee that the delay will meet the requirements. Similarly, the amount of lost packets can be quite high, for example during periods of congestion. To be able to deliver telephone quality

speech, there will have to be some quality of service (QoS) mechanism which offers guarantees about these things.

The speech data which has to be sent is typically generated at regular small intervals. It is possible that a receiving end cannot cope with this data flow, so somehow the sender should know whether the receiver can handle the incoming stream or not. A method that does this is often called a flow control method.

Also, due to the fact that data is sent at a regular basis, it is not unlikely that a link becomes overloaded and congestion occurs. In turn, congestion causes the loss of packets and an increase in delay which are not desirable features for voice communication. The transmission component should be able to detect an arising congestion and take appropriate actions. The mechanism to prevent and control congestions is called congestion control.

The appropriate action for flow and congestion control is to decrease the amount of data sent. Typically, this is done in cooperation with the compression module: when the data rate has to be lowered, the compression module is signaled to increase the amount of compression. This will usually result in a degradation of speech quality, but it is still better than having a lot of lost packets and a large delay.

8.3 Transmission Protocols

If an application wants to transmit data, it uses a certain protocol to perform this task. Recall that in the TCP/IP architecture, TCP and UDP are the protocols which an application can use.

First, we will see why the bare TCP and UDP services are not sufficient. Then, a description of the Real-time Transport Protocol (RTP) is given. This is a widely used protocol for real-time data like speech and video.

8.3.1 Why not TCP or UDP?

When we are thinking about VOIP applications which should offer a telephone-like service, TCP could seem a good candidate to transmit the speech data. It offers a service in which the connection can be seen as a reliable byte stream. To use TCP, a connection is set up, data is exchanged and the connection is torn down again. This procedure immediately reminds one of the ways a telephone call is made.

When it comes to synchronization, the reliable byte stream service seems like a very good starting point: all data arrives nicely in the exact same order as it was transmitted. Also, data is guaranteed to arrive correctly, which is also good for voice communication. The protocol also has built-in flow and congestion control mechanisms which offer good protection against overloading the network.

There are however, several disadvantages to the use of TCP. One of the basic problems is that to offer this reliable byte stream service, the protocol relies heavily on the retransmission of lost or corrupted packets. While this offers a reliable service in which order is preserved, the waiting for retransmitted packets adds extra delay to the communication. Usually, it is better to have an occasional lost or corrupted packet than having a large amount of delay.

A related issue is that one lost or corrupted packet effectively prohibits the application of receiving any packets which come after it, since TCP preserves the

order of the packets. The application has to output speech data at regular intervals, so if one packet stays lost for a sufficient amount of time, this will block the playback of other packets, even when they have already arrived.

The flow and congestion control features might seem very useful, but an application has very little control over these things. TCP can easily decide by itself to decrease the rate at which data is sent, and this again would increase the overall delay.

The key point to be made here is that TCP has a lot of features and a lot of complexity which are not very useful for VOIP. We could easily get an equal or better performance using far less elaborate protocols.

When speech data has to be distributed to several users at the same time, TCP has another major disadvantage. While IP offers the efficient distribution of data using multicasting, TCP has no support for this. If data has to distribute to several destinations using TCP, it has to be done using separate TCP connections. This, of course, wastes a lot of bandwidth.

When we eliminate TCP, the only basic protocol we can use is UDP. This protocol has almost no complexity at all. It is simply a minor extension to IP, so it offers only a best-effort service.

The protocol has the advantage of not having to wait for retransmissions of lost packets. Also, since it is only a small extension to IP, it can make use of the IP multicasting features and save bandwidth when data has to be sent to multiple destinations. As good as all this may seem the disadvantage is that the UDP

provides no mechanism for synchronization whatsoever and there are no means for flow or congestion control.

A solution to these problems is to extend UDP somewhat: we can add extra information to the speech data and use UDP to distribute this control and speech information. This is in fact how the Real-time Transport Protocol (RTP) works in the TCP/IP architecture.

8.3.2 Real-time Transport Protocol (RTP)

The Real-time Transport Protocol is formally specified in [30]. There, it is defined as a protocol which provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. So this protocol can also be used for VOIP applications.

The RTP specification actually defines two separate protocols. The first one is the Real-time Transport Protocol (RTP). The second one is called the RTP Control Protocol (RTCP). The function of RTP is to transfer the real-time data. The control protocol supplies information about the participants in the session. The protocols are defined in such a way that they can be used on a lot of network architectures and not just on TCP/IP networks. However, if RTP is used on a TCP/IP network, it is typically run on top of UDP.

The protocols themselves do not provide mechanisms to ensure timely delivery. They also do not give any QoS guarantees. These things have to be provided by some other mechanism.

Also, out of order delivery is still possible, and flow and congestion control are not directly supported. However, the protocols do deliver the necessary data to

the application to make sure it can put the received packets in the correct order. Also, RTCP provides information about reception quality which the application can use to make local adjustments. For example if congestion is forming, the application could decide to lower the data rate.

In the following sections, an introduction to RTP and RTCP is given. These sections are based on the specifications in [30], which is also the appropriate source to consult for more detailed information about these items.

RTP is used in the VOIP application which is developed. For this purpose, firstly, the RTP library was written, which is described in chapter eight.

8.3.2.1 RTP

A RTP packet consists of a RTP header, followed by the data to send. In the RTP specification this data is referred to as the payload. The header is transmitted in network byte order, just like the IP header. Figure 8.1 shows the RTP header format.

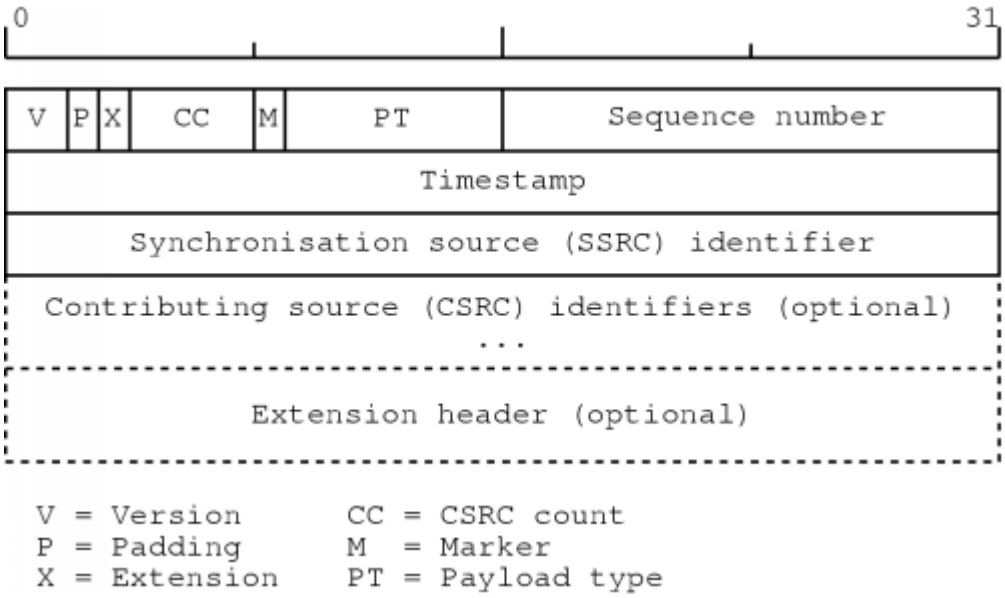

Figure 8.1: The RTP header

The first two bits of the header contain the **version** number. The current version of the protocol is `two'. Next, there is the **padding** bit. If this bit is set, the packet contains some padding bytes which are not part of the payload. The last padding byte then contains the number of padding bytes. For example, padding may be necessary for some encryption algorithms which need the payload to be aligned on a multiple byte boundary. The **extension** bit specifies if the header contains an extension header. Then, there is the **CSRC count** which specifies how many contributing sources are specified in the RTP header.

The **marker** bit can be used by an application to indicate a talkspurt for example. The exact interpretation is not defined in the RTP specification, it is left to the application itself. Next, there is the **payload type**. This defines the type of data the packet contains, so it defines the way in which the application will interpret the payload.

The **sequence number** can be used by an application to place received packets in the correct order. The numbering starts at a random value for security reasons.

The **timestamp** contains the synchronization information for a stream of packets. This value specifies when the first byte of the payload was sampled. For example, for audio, the timestamp is typically incremented with the amount of samples in the packet. Based on this value, the receiving application can then play the audio data at exactly the right time. Just like with the sequence number, the initial value of the timestamp is random. Note that several packets can have the same timestamp value: with digitized video for example, one image will

usually have to be sent in several pieces. These pieces will all have a different sequence number, but their timestamp value will be the same.

The **synchronization source (SSRC) identifier** is the identification number of the sender of the packet. If an application wishes to send different media at the same time, for example audio and video, there have to be separate RTP sessions for each of the media. This way, an application can group the incoming data according to the SSRC value. The identifier is chosen randomly; the chance that two communicating parties accidentally end up having the same SSRC value is extremely small. In the rare case that this should happen, the specification gives the appropriate course of actions to resolve this problem.

Next, there are possibly a number of **contributing source (CSRC) identifiers**. For example, if at some point different audio streams have to be mixed together, the original SSRC identifiers can be put here. The SSRC identifier of this packet then becomes the identifier of the source which forwards the mixed packet.

Finally, the header can contain extra information through the use of an **extension header**. The RTP specification only defines the extension mechanism, not the possible extensions. This is left to the application.

Note that the header does not contain a payload length field. The protocol relies on the underlying protocol to determine the end of the payload. For example, in the TCP/IP architecture, RTP is used on top of UDP, which does contain length information. Using this, an application can determine the size of the whole RTP packet and after its header has been processed, it automatically knows the amount of data in its payload section.

8.3.2.2 RTCP

The RTP protocol is accompanied by a control protocol, RTCP. Each participant of a RTP session periodically sends RTCP packets to all other participants in the session. According to [30], RTCP has four functions

- The primary function is to provide feedback on the quality of data distribution. Such information can be used by the application to perform flow and congestion control functions. The information can also be used for diagnostic purposes.
- RTCP distributes an identifier which can be used to group different streams - audio and video for example - together. Such a mechanism is necessary since RTP itself does not provide this information.
- By periodically sending RTCP packets, each session can observe the number of participants. The RTP data cannot be used for this since it is possible that somebody does not send any data, but does receive data from other participants. For example, this is the case in an on-line lecture.
- An optional function is the distribution of information about a participant. This information could be used in a user-interface for example.

There are several types of RTCP packets which are used to supply this functionality. Sender reports (SR) are used by active senders to distribute

transmission and reception statistics. If a participant is not an active sender, it still distributes reception statistics by sending receiver reports (RR). Information which describes a participant is transmitted in the form of source description (SDES) items. There is also a packet type to allow application specific data (APP). Finally, when a participant is about to leave the session, it sends a goodbye (BYE) packet. The transmission statistics which an active sender distributes include both the number of bytes sent and the number of packets sent. It also includes two timestamps: a Network Time Protocol (NTP) timestamp, which gives the time when this report was created, and a RTP timestamp, which describes the same time, but in the same units and with the same random offset of the timestamps in the RTP packets. This is particularly useful when several RTP packet streams have to be associated with each other. For example, if both video and audio signals are distributed, on playback there has to be synchronization between these two media, called inter-media synchronization. Since their RTP timestamps have no relation whatsoever, there has to be some other way to do this. By giving the relation between each timestamp format and the NTP time, the receiving application can do the necessary calculations to synchronize the streams. A participant to a RTP session distributes reception statistics about each sender in the session. For a specific sender, a reception report includes the following information

- The fraction of lost packets since the last report. An increase of this value can be used as an indication to congestion.
- The total amount of lost packets since the start of the session.

- Amount of inter-arrival jitter, measure in timestamp units. When the jitter increases, this is also a possible indication of congestion.
- Information that can be used by the sender to measure the round-trip propagation time to this receiver. The round-trip propagation time is the time it would take a packet to travel to this receiver and back.

The source description items give general information about a participant, like name and e-mail. But it also includes a so-called canonical name (CNAME). This is a string which identifies the sender of the RTP packets. Unlike the SSRC identifier, this one stays constant for a given participant, independent of the current session and it is normally unique for each participant. Thanks to this identifier it is possible to group different streams coming from the same source.

Since these packets are sent periodically by each participant to all destinations, we have to be careful not to use too much of the available bandwidth for RTCP packets. The RTCP packet interval is calculated from the number of participants and the amount of bandwidth which RTCP packets may occupy. To prevent that each participant would sent its RTCP packets at the same time, this value is multiplied by a random number.

8.3.3 Packet Size

Now that we have a decent protocol which we can use to transmit the digitized speech, we need to address another matter. With RTP, we can transmit packets containing voice information, but what should the size of these packets be?

We have already seen in a previous chapter that packets containing only a small amount of voice data are desirable for two reasons. First, if a packet gets lost, it does not cause a severe distortion in the communication. Second, to reduce the overall delay, the sampling interval should be as low as possible and each piece of the digitized voice signal should be transmitted as soon as possible. This automatically implies small packet sizes.

But we have to keep in mind that when this speech data is transmitted, a part of the bandwidth will be occupied with headers of the underlying protocols. So, the smaller the time interval captured in a packet, the larger is the overhead caused by headers.

Consider the following example. A voice signal is sampled at regular intervals of one millisecond. After each sample interval we will transmit the digitized signal using RTP. This means that every millisecond, at least a RTP, UDP and IP header is actually transmitted over some medium. Their total size is at least forty bytes. Sending forty bytes each millisecond needs a bandwidth of 320 kbps. It is obvious that if we increase the sampling interval, the bandwidth occupied with only header information will decrease. This will result in a larger overall delay, so we must be careful not to make the sampling interval too large. It will also result in larger packet sizes which causes the communication to be more vulnerable to lost packets. Clearly, somehow a compromise will have to be made. Usually, sampling intervals of ten to thirty milliseconds are used, since these are the sampling intervals which a lot of compression techniques use. When bandwidth is scarce, perhaps even larger values are advisable.

With dial-up links, the available bandwidth is very low compared to the available bandwidth on a LAN for example. In this case, we would like to have as much bandwidth available for the actual data as possible. Luckily, there exist methods to greatly reduce the bandwidth occupied by header information on such links.

When you are using a dial-up link, a lot of consecutive packets will go to the same destination application. This means that many IP and UDP header fields stay the same. When RTP is used on top of UDP, a number of fields in the RTP header will also stay the same, while the values of other fields change with a fixed amount for each packet. Using this information, the aggregate header size of forty bytes can be reduced to two to four bytes! This, of course, greatly reduces the bandwidth occupied by header information. The exact way to do all this is specified in [8].

8.4 QoS Mechanisms

We mentioned before that RTP itself offers no way to achieve certain levels of QoS, it relies on external methods to provide this. There are several ways in which this can be done. This section gives an overview of such methods.

8.4.1 Assigning Priorities to Packets

Both IPv4 and IPv6 have a way to specify the priority of a datagram. In the IPv4 header some level of QoS can be specified in the TOS field. The IPv6 header has a similar feature through the use of the traffic class field.

If all routers take such priorities into account, this could help real-time data to be delivered with, for example, low delay. The main advantage of this approach is

that no additional protocols are needed. The only thing that needs to be done is to adjust routers so they can take the priorities of packets into account.

But these mechanisms can only help to give a better service; they cannot give any guarantees whatsoever. For example, if the whole network is filled with high-priority traffic, the quality will still be poor.

So we have to rely on other means if we want to be able to provide guarantees about QoS. In the following sections we will explore two protocols which are designed for this purpose. First, an explanation of version two of the Stream Protocol (ST2) is given. Next, I will describe the Resource Reservation Protocol (RSVP). To give guarantees about QoS, both of these methods rely on the reservation of resources. The way this is done in each of these cases differs greatly. Therefore, a section with a comparison of their basic techniques is presented afterwards.

8.4.2 Stream Protocol Version Two (ST2)

The Stream Protocol version two (ST2) was first specified in [13]. This document was released in October 1990. Five years later, after gaining experience with the protocol, it was revised and redefined in [22]. The main goal of the revision was to simplify the protocol and to clarify some issues. Also, some extensions were added. The basics of ST2 remained the same however. The information in this section was obtained from the two mentioned references.

Within the TCP/IP architecture, ST2 is situated in the internet layer. Its purpose is to provide an end-to-end guaranteed service across an internet. The protocol is

not intended as a replacement for IP, but as an addition. This way, general data transfers can still use IP while real-time data can be transmitted using ST2.

Unlike IP, which is a connectionless protocol, ST2 is connection-oriented. This implies that to transfer data, there are three stages involved. First, the connection has to be set up. During this stage, resources are reserved to be able to provide certain QoS guarantees. When the ST2 connection has been established, the actual data transfer can take place. When all data has been transmitted, the connection has to be released again.

A connection will only allow the flow of data in one direction: from the origin to the destinations. If communication in the other direction has to be possible, a different ST2 connection will have to be made. This way, the connection can be represented by a directed tree, from the origin to the destinations. Using this model, the distribution of data will be done in such a way as to minimize the amount of duplicate packets sent.

To create a connection, an application must first know a number of targets to connect to. For this purpose, the application cannot use ST2 itself, it must use some other means. Probably, some IP based protocol will be used to do this.

When the application knows the destinations and the necessary QoS constraints, it can deliver this information to the ST2 module and ask it to set up a connection to these destinations. Within ST2 the QoS constraints are distributed by means of a flow specification, also called FlowSpec.

Based upon the information in this FlowSpec, the intermediate ST2 supporting routers can make the necessary reservations. If these reservations do not

correspond to the desired QoS, the information in the flow specification is updated to reflect the actual obtained QoS at the current point.

When a destination receives the connection request, this FlowSpec can be investigated by the application and it can decide whether to accept the connection or not. If the connection is accepted, the FlowSpec with the acquired QoS values is propagated back to the origin of the connection request.

If all goes well, the application which attempted the connection, receives a confirmation of each destination. These confirmations also contain a FlowSpec describing the obtained QoS for each target. The application can then decide whether to start sending its data or to abort the connection.

Note that at different stages along the path from origin to destinations, the reservations can differ. The application has to explicitly release any excess reservations.

During the lifetime of the connection, there is still a possibility to add destinations. The procedure is similar as the creation stage and it is started by the origin of the connection. Destinations can also be removed from the connection. This can be done by the origin or the destination itself.

Only the data packets are in fact transmitted using ST2. Other functions are provided through the use of a control protocol: the ST Control Message Protocol (SCMP). These functions include connection creation and adding destinations. All control messages are transmitted reliably, using acknowledgements and retransmissions if necessary.

To be able to detect the failure of network elements, each ST2 capable machine periodically sends a `hello' message to each of its neighbors. If necessary, recovery procedures will be initiated.

Note that ST2 itself does not specify how the reservations should actually be made or how the QoS itself should be provided. It only presents a way to distribute the desired QoS specifications.

8.4.3 Resource Reservation Protocol (RSVP)

Another way to reserve resources is by using the Resource Reservation Protocol (RSVP). This protocol is specified in [29]. The protocol is a part of an Integrated Services model, described in [17]. The term Integrated Services refers to the fact that several kinds of services can be offered, for example both real-time and best-effort services.

Unlike with ST2, RSVP does not provide its own data transmission protocol. This function is still performed by IP. RSVP is merely a control protocol which can be used to help provide QoS guarantees to applications. The protocol can be used with both IPv4 and IPv6.

If a host is going to transmit data which should arrive with a certain QoS, it periodically sends a so-called path message to the destination of the data. This address can be both a unicast and multicast address. The path message contains information about the characteristics of the traffic that will be generated by this sender. Also, the format of its data packets is described. Using this information it is be possible to select packets from this specific sender out of others.

On its way to the destinations, RSVP capable routers store the information in the path message. This way, it can be used when reservations will be made for data coming from the sender of the path message.

When an application receives a path message, it can decide it wants to receive the sender's data with a specific QoS. It can determine the necessary QoS constraints from the information contained in the path message. To request this QoS, it will periodically send a reservation request along the reverse path of the path message. The exact reverse path can be followed because of the saved information in RSVP capable routers in response to the path message.

The reservation request contains two items. First, it contains the QoS which the receiver would like to obtain. Second, it contains a description of the set of data packets which should be received with this QoS.

Inside each router, the necessary reservations can be made to supply the QoS. An important feature of RSVP is that reservations may be merged. After a merge, the router checks if there is a net change for the link upstream, and if so, an appropriate reservation request is sent to the previous router.

Let’s illustrate this with a small example. Consider the situation in figure 8.2. Suppose host A is the sender and is distributing digitized speech as part of an on-line conference. This data is being distributed by sending it to a multicast address to which path messages have also been sent. Host B in response, has issued a reservation request which caused the reservation of 32 kbps along the path from host A to B.

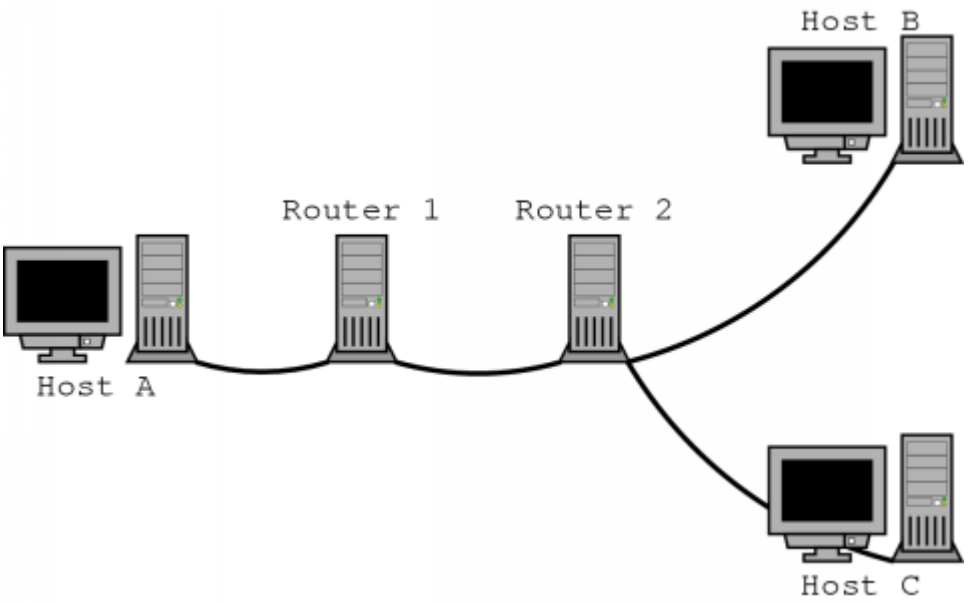

Figure 8.2 - Reservation example

Now host C also joins the conference and wishes to have a guaranteed 16 kbps link with host A. The host creates a reservation request which specifies this and sends it to router 2. There, the router investigates the request and reserves bandwidth of 16 kbps downstream over the link to host C. It also notices that upstream, there is already a reservation of 32 kbps to host A, so no extra reservation request will have to be sent to router 1. If, however, host C would have requested 64 kbps, new reservation would have been forwarded to router 1. Several reservation styles are supported by RSVP. For example, if a host receives data from many sources, it could issue a reservation request which would allocate separate bandwidth for each source. But it is also possible to specify that the allocated bandwidth should be shared by all senders. This is useful in case of an on-line discussion where there will be usually only one speaker at a time. It would then be sufficient to allocate bandwidth to accommodate only one or a few speakers.

Note that the path messages and reservation requests are sent periodically. This is because the RSVP information within a router will time out after a while. To keep the path information and reservations in place, they have to be updated regularly. This is called a soft-state mechanism. When a reservation or path state times out, the associated resources can be released. Resources can also be released explicitly when a sender or receiver quits.

Like ST2, RSVP does not specify how actual QoS guarantees have to be enforced. The protocol is only used to distribute the QoS related information.

8.4.4 ST2 vs RSVP

Both ST2 and RSVP provide a mechanism which can be used to make resource reservations along the path from sender to receivers. These resource reservations are intended to supply QoS guarantees. With both protocols, the resources are allocated for data distribution in one direction only: from sender to receivers. This can be modeled as a directed tree, with the root being the sender. But the approach that these protocols follow differs significantly. So the question arises which one is the most efficient. In [5] a comparison is made between the two protocols. An important difference is from where the reservation requests originate. With ST2, it is the sender which makes the necessary reservations along its distribution path. In contrast, with RSVP the receivers request the reservations. For this reason, ST2 is often said to have a sender initiated reservation style, while RSVP's reservation style is called receiver initiated. For some applications, there will be several senders, but there will usually be only a few of them sending at the same time. Like was mentioned above, with RSVP

this knowledge can be exploited through the use of a specific reservation style. ST2 however, does not have such a feature. Here, reservations will have to be made for each possible sender and this, of course, wastes a lot of bandwidth. When data is distributed to a number of receivers, it is very unlikely that all these receivers will have the same QoS demands. Since ST2 is sender-initiated, the sender will have to request reservations to satisfy the needs of the most demanding receiver. Even branches that lead to less demanding receivers will all have the same reservations. With RSVP, the heterogeneity of receivers is handled much more efficiently. The requests originate from the receivers themselves and if possible, requests are merged. This means that branches of the distribution tree which lead to less demanding receivers will have fewer reservations. A previous example already illustrated this. When a receiver is unable to accommodate data streams from all active senders, it may wish to be able to dynamically select from which sources to receive data. This is called channel selection. With ST2, the only possibility for channel selection is to make a separate reservation for each sender. The actual channel selection will have to be done at the receiver. Recall that a RSVP reservation request contains a description of which packets should receive the associated QoS. This way, when a reservation request is sent, a new set of sources can be selected and filtering can be done inside the network. Network failure detection in ST2 is done by periodically sending messages to neighboring machines which participate in the same stream. If an error is detected, a recovery procedure is started. RSVP completely relies on the soft-state mechanism to automatically adapt to any

failures. Note that both protocols send messages periodically. With RSVP however, the overhead of these messages is reduced by merging reservation requests were possible. When a receiver joins a ST2 session, the reservations for this receiver have to be requested by the sender. This way a message will have to travel all the way from the sender to the new receiver. With the receiver-initiated approach of RSVP, a reservation request is only propagated towards the sender until it can be merged with other reservations. This results in less protocol overhead. However, the receiver may have to wait a while to send its request until it receives a path message.

8.5 Transmission delay

When resource reservation methods are supported in routers, transmission delays can probably be kept low enough to satisfy the overall delay constraint of 200 ms. But at this time, routers which are currently used in general do not have such capabilities. When data is transmitted there is always a minimal amount of delay due to the capacity of the links along which the data travels. But the most significant part of the delay by transmission is usually due to queuing of packets inside routers. This delay is highly variable and depends both on the number of routers along the path and the load of the routers. It is not possible to make a general claim about transmission delay in IP networks, although one-way transmission delays rarely tend to exceed 100 ms [41]. However, it is not inconceivable that this delay can exceed 200 ms

9

A VOIP Framework

9.1 Introduction

By separating the VOIP components from the rest of the application, this task is much simpler. Furthermore, I wanted to create it in such a way that would allow testing of different components, for example different compression techniques. This work led to the C++ framework described in this chapter. First, I will present the general structure of this framework. Afterwards, its implementation is discussed.

9.2 Framework Layout

The structure of the framework is depicted in figure 9.1. The Voice Call class is the one which connects several components. As you can see, these components are pretty much the ones described in this thesis.

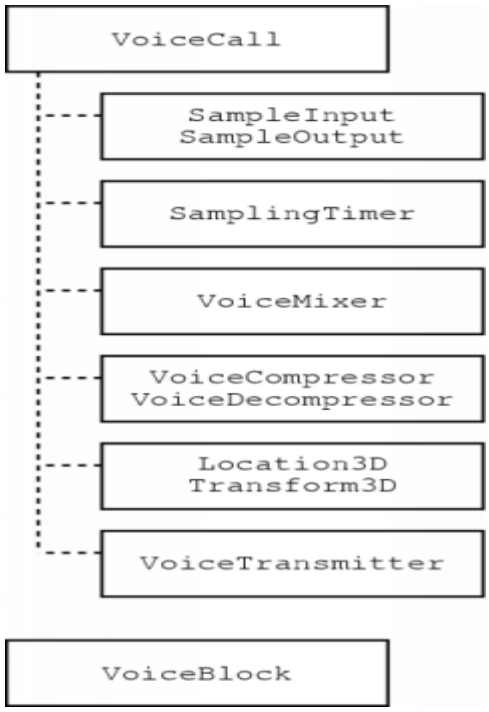

Figure 9.1- VOIP framework layout

The SampleInput and SampleOutput classes are the ones responsible for grabbing and reconstruction of voice signals.

In the framework, the intervals for both capturing the voice signal and playback are assumed to be the same. The SamplingTimer class is responsible for indicating when this interval has elapsed.

With VOIP in virtual environments, there can be several persons speaking at the same time. The VoiceMixer component is responsible for mixing these signals together. This component gives the SampleOutput class a new block of speech data when an interval has elapsed, so it must make sure that everything is in the correct order.

The VoiceCompressor and VoiceDecompressor classes compress and decompress a block of speech data respectively.

For VOIP in virtual environments, the sender of speech data has to include some information about the position of the sender. This is done in the Location3D class. The actual 3D effects are added at the receiver side by the class Transform3D.

Somehow, the speech data will have to be transmitted and received. These tasks are the responsibility of the VoiceTransmitter class.

The VoiceBlock class is used as a container for voice data. An instance of this class is passed between the previously listed components to make VOIP possible. This class has member functions indicating properties of the data, for example the sampling rate, so that each component can process the data correctly.

In this basic framework, only the VoiceCall and VoiceBlock are really implemented, the other classes are abstract. Using inheritance, this allows us to try several different techniques. For example, we can easily try several compression schemes by implementing them in classes inherited from VoiceCompressor and VoiceDecompressor.

9.3 Implementation

In this section I will cover the implementation of the framework. First, I will explain how the VoiceCall class uses the other components to make VOIP possible. Then, the components themselves are discussed.

9.3.1 Main VoiceCall Routine

The VoiceCall class contains several members to set up the different components. When this is done, the application only has to call the `Step' member function to make VOIP possible. The outline of this function is depicted in the pseudocode in Appendix C. In principle, this function should be called continuously to assure that the necessary actions are taken at the exact point when the sampling (and playback) interval has elapsed. However, one could implement the abstract classes in such a way that the application only calls the `Step' function when necessary. To give an example, you could let an implementation of the SamplingTimer send a signal to the application when an interval has elapsed. In turn, the application can then call the `Step' function. The pseudocode should be quite clear, but it may be necessary to explain what the sample offset is for. When voice data is received, it is put in a VoiceBlock instance. This will also hold information about when this data will have to be

played, expressed in terms of samples. This timing information is used by the mixer to insert this data at the correct position in its output stream. When the mixer passes a block of data to the playback routine, the sample number of the first sample in its queue has changed. It is this value that is passed to the transmission component. The current sample offset is needed each time a participant joins in. The timing information in the packets of this participant only provides information about when the data should be played, relative to the start of the participant's transmission. To provide the correct timing information for the mixer, the current sample offset has to be added to the timing information in the packet. There is another use for the current sample offset. After having calculated the timing information for a block of voice data, the transmitter can easily see if it is still possible to play this data: if the sample offset of the block is smaller than the current sample offset, it is useless to process the block any further since its playback time has already passed. The transmitter can then simply discard the data.

9.3.2 Grabbing and Reconstruction

The grabbing and reconstruction routines are rather straightforward. They simply use the operating system's capabilities to either record or playback speech data. Routines for both Linux and MS-Windows platforms have been used. On both platforms, SampleInput and SamplingTimer have been implemented in one class, using multiple inheritances. This way, recording interval can be used as timing information.

9.3.3 Mixing

At the end of each sampling interval, the mixer sends a block of data to the playback routine. For this reason, mixer has been implemented using a linked list of such blocks. Initially, these blocks contain only silence. Each time the mixer receives a block, it adds the data to the blocks in the list. Because of the principle of superposition mentioned in chapter three, the data simply has to be added to the data which is already present.

9.3.4 Compression schemes

Because of the structure of the framework, it was easy to try several compression schemes. The first scheme that was implemented was the simple silence suppression technique which has been described in chapter three. This silence suppression scheme was also used in the other compression methods which implemented:

- a simple delta modulation scheme,
- a DPCM technique, and
- the wavelet coding method described in chapter four.

The simple DPCM technique was observe to produce the most acceptable results. The delta modulation method simply could not reproduce the signal accurately enough. The wavelet coding technique could do this, but to achieve the same quality as the DPCM technique, the required bandwidth was comparable to that of DPCM. A Linear Predictive Coder has also been tested, using a library which provided the LPC coding and decoding functions. The resulting communication quality was very good, but the speech signal did sound

somewhat synthetic. To allow somebody to choose a compression technique at run-time, a wrapper class was created for these compression modules. If all users in a session use that wrapper class as the compression and decompression module, each user can select the compression scheme he prefers.

9.3.5 Localization effects

At the sender side, the sender's position was added to the data. This information was then used by a receiver, together with its own position, to create a localized effect. The technique used was described in chapter seven.

9.3.6 Transmission

The transmission module was implemented using RTP to send the data. The RTP functionality was provided by JRTPLIB, which made the module implementation easier and faster. This module tries to stay synchronized as good as possible. When too many consecutive packets from a certain source have to be discarded, the module resynchronizes with that source. Also, as was mentioned in chapter three, the amount of buffering is determined using jitter information provided by RTCP packets. Buffering is simply done by adding some value to the timing information of a block with speech data. This will cause the mixer to insert the block further on in its linked list and this, in turn, will cause the data to be played back a bit later. The module also filters out any duplicate packets. When the same data is passed to the mixer several times, that piece of the signal will sound much louder because of the principle of superposition. Such a discontinuity is very disturbing when VOIP is used in 3D environments.

10

Gnokii-0.6.2

SDK

10.1 Introduction

In this chapter we will show the implementation of GSM phase for our project. Since GSM can't be customized. And even you will not found any implementation so you can start form somewhere. How Nokia vendors have written software for GSM would be different from Motorola.

10.2 Gnokii-0.6.2 SDK

For GSM Phase Implementation gnokii-0.6.2 SDK is being used. Since, we cannot program the mobile equipment directly. By using it we can get basic functionality of mobile equipment at the PC. So, through following ways we can connect to mobile equipment. This SDK is compatible with NOKIA mobiles.

- Parallel Port
- USB Port
- IrDA

For Nokia 61X0/51X0 models you should use DAU9P cable. Nokia 61X0 models are also capable to make an infrared connection.

For Nokia 7110/62X0/6310 models you need either DAU9P or DLR3P cable. With DLR3P cable you can use AT commands to talk to the phone. These phones are able to make an irda connection.

For modern Nokia phones without external memory (6100, 3100, 3120, 3200, 3220, 6610, 7210, 7250, 68xx, 6200, 6220) use DKU-5 cable. The recent replacement for this cable is named CA-42.

For modern Nokia phones with external memory (6620, 7610, 6630, 3230, 6230, 668x, 9300, 7710, 9500, 3200) use DKU-2 cable.

These models are supported (ie. reported to work)

- nk6100 driver: Nokia 6110, 6130, 6150, 6190
  Nokia 5110, 5130, 5190
  Nokia 3210, 3310, 3330, 3360, 3390, 3410
  Nokia 8210, 8250, 8290, 8850
  Nokia RPM-1
- nk7110 driver: Nokia 7110, 6210, 6250, 7190
- nk6510 driver: Nokia 3510(i), 3510i 3595
  Nokia 5100
  Nokia 6020, 6100, 6170, 6230, 6310(i), 6360, 6510
  Nokia 6610(i), 6650, 6800, 6810, 6820, 6820b, 6610i, 6230
  Nokia 7210, 7250, 7250i, 7600
  Nokia 8310
- nk3110 driver: Nokia 3110, 3810, 8110, 8110i
- nk6160 driver: Nokia 5160, 6160, 6185
- atgen driver: AT mode compatible phones (namely Nokia 62X0, Nokia 7110, Nokia 8210, Nokia 6310, Nokia 6510 (you should expect all newer Nokia phones to work), Ericsson T39, Sony-Ericsson T68i, Siemens S25/SL45i/C55/M55/S55,

Motorola Timeport P7389i (L series), C350, Bosch 908/909 are known to work)

- gnapplet driver: Symbian series60 based phones (namely Nokia 3650, Nokia 3660, Nokia 6600, Nokia 7650, Siemens SX1 are known to work)

10.3 GNOKII Implementation for our Project

Though this phase was excluded from our project. Since our basic aim was to use sim base PDA's. However we have programmed it for use of our project.

11

Conclusions

&

Future Work

11.1 Introduction

In this chapter we will show what we have achieved, what are the results and what are the pros and cons of our project and what future advancements can be made to our project.

11.2 Achievements in Project

Throughout this project report we have emphasized on communication over IP networks using WLAN Infrastructure. We have established communication over WLAN.

Initially we set the following Objectives.

- WIRELESS Communication Phase
- VOIP Programming Phase
- VOIP Codec Development Task
- Application Development Task
- Final Testing

We have achieved the above mentioned objectives. During CODEC development phase we have used two voice compression codec GSM 6.10 and G.711.Now the purpose to use these codecs is due to their following characteristics.

- GSM 6.10 is best ever codec for voice compression.
- GSM 6.10 is best for communication over local LAN where packet loss is not very high due to less traffic and peer-to-peer communication.
- G.711 is best known codec where communication is required over the internet where a chance of loss of packets is very high.

- G.711 is best for low usage of bandwidth since over internet bandwidth is major concern for VoIP.

We have achieved following targets w.r.t Objectives.

Since the project was divided into three major phases. These phases were assigned to three different groups by Dr Shoaib Ahmed Khan. The phases were as

- Wireless LAN Phase
- GSM Phase
- Integration of Wireless LAN and GSM Phase.

So, we were assigned wireless LAN Phase. In this phase we have achieved the following results w.r.t Objectives.

- Peer-to-Peer communication over WLAN achieved.
- We have made only SIP based clients and used already available server called ONDO proxy server to check the validity of our clients and they were working fine.

11.3 Future Work

- One of the future advancements in this project are that each mobile can become adhoc based. There is no need of infrastructure, i.e each mobile node when it enters into the range of another one it hand shakes with it and identify itself by sending its IP address.
- Each mobile can be considered as router for other nodes. If two nodes cannot be connected directly.

- New wireless standards are coming in market that is purposefully to increase the wireless LAN range. That underlying architecture can be useful for this project.

Appendix

A

Appendix A

Abbreviations used in the text

Abbreviation	Abbreviated for
ACK	Acknowledgement
ADPCM	Adaptive Differential Pulse Code Modulation
AES	
API	Application Programming Interface
ARDIS	Advanced Radio Data Information Service
ARP	Address Resolution Protocol
AuC	Authentication Center
BSC	Base Station Controller
BTS	Base Transceiver Station
CDPD	Cellular Digital Packet Data
CELP	Code Excited Linear Prediction
CSMA	Carrier Sense Multiple Access
CT	Cordless Telephone
CTS	Clear To tranSmit
DCT	Digital Cordless Telephone
DECT	Digital European Cordless Telephone
DES	Data Encryption Standard
DHCP	
DNS	Domain Name Service
DSSS	Direct Sequence Spread Spectrum
EDCA	Enhanced Digital Control Access
EIR	Equipment Identity Register
FDMA	Frequency Division Multiple Access
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GSM	Global System for Mobile communication
GUI	Graphical User Interface
HF	High Frequency
HLR	Home Location Register
HTTP	Hyper Text Transfer Protocol
ICMP	Internet Control Message Protocol
IDEA	International Data Encryption Standard
IETF	
IP	Internet Protocol
ITU	International Telecommunications Union
LAN	Local Area Network

LOS	Line Of Sight
LPC	Linear Predictive Coding
LPTSL	Loss Profile Transport Sub Layer
LTP	
MAC	Medium Access Control
MDN	Microcellular Data Network
MD	Message Digest
ME	Mobile Equipment
MEGACO	MEdia GAteway COntrol
MH	Mobile Host
MSC	Mobile services Switching Center
MSS	Mobile Support Station
NACK	Negative Acknowledgement
NAT	Network Address Translation
OFDM	Orthogonal Frequency Division Multiplexing
PBX	
PCS	Personal Communications Service
PDA	Personal Digital Assistants
PHS	Personal Hand Set
PSTN	Public Switched Telephone Network
QCELP	
QOS	Quality Of Service
RF	Radio Frequency
RPE	
RTP	Real-time Transport Protocol
RTS	Ready To tranSmit
SDP	Session Description Protocol
SH	Supervisor Host
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TKIP	
URI	Uniform Resource Identifier
USB	
UWB	Ultra Wide Band
VC	Virtual Circuits
VIP	
VLR	Visitor Location Register
VOIP	Voice Over Internet Protocol
VSELF	
WAN	Wide Area Network
WEP	
WLAN	Wireless Local Area Network

Appendix

B

Appendix B

Windows CE Programming

The Windows CE operating system is based on the Win32 application programming interface (API). The fundamentals of programming for Windows CE closely parallel programming for Windows 98, Windows NT, and Windows 2000. As with the other Windows operating systems, Windows CE is an event-driven programming model. A Windows CE-based program receives messages, interprets the messages, and acts on the messages.

Message Loops and Windows Handles

A Windows CE program has one or more Windows that receive and process messages in a message loop. The Windows can be visual or, for an application that does not require a user interface, invisible. Each window has a window handle (hwnd) associated with a message processor that handles the messages for the window. You can also use the window handle to call any related function.

Like any other Windows-based program, a Windows CE program has two primary functions, a message processor (usually called **WndProc**) and **WndMain**, which provides an entry point to the program. The **WndProc** function processes messages for the Window. In general, an application processes only those messages that are relevant to it, and passes other messages back to the operating system. In addition to being the primary message process for an application, **WinMain** also handles initialization and shutdown.

When developing a program for Windows CE, you must first determine the hardware platform and processor on which your program is going to run. You must also know the hardware configuration for which you are developing the program. The platform and processor will be determined by the SDK you are using. Because Windows CE is a modular operating system, an original equipment manufacturer (OEM) chooses specific modules and components to configure Windows CE devices. You may have to consider, for example, the memory that will be available to your application.

Programming the Pocket PC

Microsoft eMbedded Visual C++ provides development environment for the embedded devices. However, unlike Visual C++, eMbedded Visual C++ provides tools and resources designed especially for creating Windows CE applications. Along with the SDKs, it includes only those language components supported for the Windows CE platforms. The default project settings are configured for the Windows CE platforms, and the debugging tools are designed to test applications on CE device simulators or the devices themselves. Microsoft eMbedded Visual CE includes these major components:

- Project Workspace to organize projects and their components.
- Text editor to write and edit code.
- Resource editors to design and modify resources, such as dialog boxes and menus.
- Compiler.

- Windows CE device emulators to test applications on the local computer.
- Platform Manager to test applications on a Windows CE Device.
- Debugger to control step-by-step execution and breakpoints.
- AppWizard to create working frameworks for various types of applications target platforms.
- Source Browser to examine and edit functions and classes in a program and quickly see relationships between them.
- Class Wizard to automate maintaining class code.

A brief description of all the components is described below:

Project Workspace

The project workspace, in the Workspace window, provides views of the projects related to an application. A project is the configuration and files that produce an executable binary file, normally an .exe or .dll file in Windows. A simple application may have only one project generate a single executable file. A large application, containing multiple components, may have multiple projects generating several .exe and .dll files.

The views provided by the Workspace window are the class, resource, and file views. The class view shows classes in a project. The resource view displays the project resource files grouped by resource types. The file view shows the source, header, resource, and other files in a project.

Text Editor

The eMbedded Visual C++ text editor is an integral part of the IDE. When you work with code or other text, the editor provides the usual clipboard, editing, and text entry functionality common to nearly any text editing window. In addition to these basic editing capabilities, the text editor also provides additional functionality to help you write code.

The editor provides code statement completion, auto listing of object members, and parameter and type information. It can automatically indent blocks of code, color-code language and other programming elements, tab blocks of code, bookmark lines in a file, go directly to, and set or clear breakpoints. It can also go directly to specified addresses, bookmarks, errors, tags, lines, offsets, or references. With the search command, you can choose to search all open documents in addition to the active document, mark lines that contain matches, and search for files containing specified text in one or more folders on local or network-based storage locations.

The active Window determines which editing features are available. One or two functions may be available for a project workspace window. Most or all editing features are available when a code or header file is active.

Resource Editors

The Resource View tab in the project Workspace window shows the resources that you can edit. These are grouped by resource types, including accelerators, dialogs, icons, menus, and strings. When you open a resource, it automatically opens in the appropriate editor for that resource type.

Each project has a resource script (.rc) file that describes the resources shown on the ResourceView tab. It may have *#include* statements that bring in resources from other files. These can include bitmaps and icons specific to a project, or other resources common to all eMbedded Visual C++ programs. The resource editors normally enable you to make any changes you require. They even process .exe and .dll files, so you can use the clipboard to get resources from other Windows applications. You cannot edit the .rc file directly in eMbedded Visual C++. You can edit a resource script directly with a text editor such as Notepad, but we do not recommend it.

Compiler

The eMbedded Visual C++ compiler can process both C source code and C++ source code. It determines the language by the source codes filename extension. A .c extension indicates C source code and a .cpp or .cxx extension indicates C++ source code. The compiler is ANSI compliant for C and C++.

Windows CE Device Emulators

Microsoft eMbedded Visual C++ provides desktop emulation. You can configure the IDE to build for an emulation, such as *Win32 (WCE emulator) Debug.* Then when you run the application from within the eMbedded Visual C++, the IDE opens the emulation on the desktop, downloads the executable files, and starts the application. You can run and test the application on the emulation just as you would on an actual Windows CE device.

Before you distribute a finished product, you will certainly want to test it on the actual Windows CE devices. However, the emulation enables you test code

much more quickly than downloading files after every compilation. It also enables you to test an application when no Windows CE device is connected to your desktop computer.

Platform Manager

The eMbedded Visual C++ Platform Manager manages all communication between the remote devices and tools and the debugger. The Platform Manager commands are built into the menus and dialog boxes of the tools; you implicitly use the Platform Manager every time you connect or disconnect a tool from a device.

Debugger

The eMbedded Visual C++ debugger saves breakpoints on disk. The Debug toolbar buttons insert and remove breakpoints and control single-step execution in your code. The Watch, Variables, Registers, Memory, Call Stack, and Disassembly windows opened by buttons on the Debug toolbar show extensive information about your code. If you position the pointer on a simple variable, the debugger shows you its value. To debug a program, you must build it with the compiler and linker options set to generate debugging information.

AppWizard

The AppWizard is a code generator that creates a working skeleton of a Windows CE application with features that you specify on wizard pages. The code generated by any AppWizard provides an initial framework for you to create an application. In addition to the AppWizards provided with eMbedded Visual C++, you can develop your own with macros.

Source Browser

The Source Browser displays information about the symbols (classes, functions, data, macros, and types) in your program. The Browser relies on browse information (.bsc) files generated by the BSCMAKE utility. Building the browse information file adds additional time during compilation, so in the project settings you can turn on and off the building of these files. In the source browser, you can examine:

- information about all the symbols in any source file
- the source code line in which a symbol is defined
- each source code line where there is a reference to symbol
- the relationship between base classes and derived classes
- the relationships between calling functions and called functions

ClassView

The ClassView shows a tree view of all classes in your project as well as member functions and data members. You can double-click on any element to see the source code for it. ClassView does not use the browse information file required by the Source Browser, but it also does not show the hierarchy information shown by the Source Browser.

Class Wizard

For MFC programs, you can use the ClassWizard to perform many class-related programming tasks.

Windows Diagnostics Tools

Microsoft eMbedded Visual C++ provides remote diagnostic tools for Windows CE devices. You can use Windows CE Remote Spy (Cespy.exe) to view a device's processes, threads, and windows. Use Remote Registry Editor (Ceregedt.exe) to examine and edit the registry on a remote device. Use Remote Heap Walker (Ceheapwk.exe) to view detailed information about heap identifiers and flags for processes that are running on a device. Use Remote Process Viewer (Cepview.exe) to examine processes and threads on a remote device and to kill rogue processes. Use Remote Zooming (Cezoomin.exe) to capture images from a remote devices display and view them on the desktop computer. Use Remote File Viewer to view the hierarchy of folders and files on a remote device.

Building an Application

Microsoft eMbedded Visual C++ provides two ways of building an application. The easiest and most common way is to build within the eMbedded Visual C++ development environment. The other way is to build from the MS-DOS prompt using command-line tools. Building an application involves the preprocessor, the compiler, and the linker.

- The preprocessor prepares source files for the compiler by translating macros, operators, and directives.
- The compiler creates an object file containing machine code, linker directives, sections, external references, and function/data names.

- The linker combines code from the object files created by the compiler and from statically-linked libraries, resolves the name references, and creates an executable file.

The Build Process

The following diagram shows the components of the build process in eMbedded Visual C++ starting with the editor in which you create your source code.

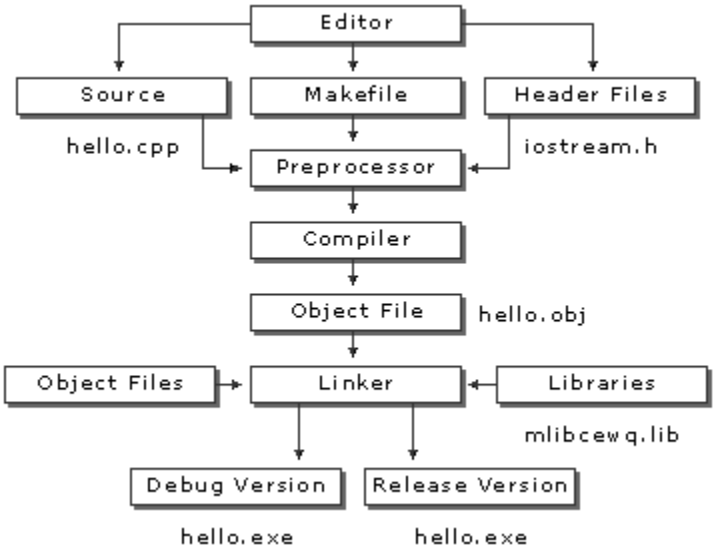

Figure C-1 Build Process

If you build your program outside the IDE, you may use a makefile to invoke the command-line tools. Microsoft eMbedded Visual C++ provides the NMAKE utility for processing makefiles. If you build your program within the IDE, the eMbedded Visual C++ project system uses the project (.vcp) files to store make information. The .vcp files are not compatible with NMAKE. However, if your program uses a makefile rather than a .vcp file, you can still build it in the development environment as an external project.

Testing and Debugging an Application

Microsoft eMbedded Visual C++ provides tools to help test and debug your application. In the eMbedded Visual C++ options, you can either automatically or manually download your programs after building them to a connected device. When you have completed building a project configuration, you can run the program in eMbedded Visual C++ with or without debugging capabilities provided by the Integrated Development Environment (IDE). Running programs without using the debugging capabilities is faster because eMbedded Visual C++ does not have to load the debugger first. With the debugger, however, you can use breakpoints and step through execution, inspect memory and registry values, check variables, observe message traffic, and generally examine closely how your code works.

In addition to the debugging features described previously, eMbedded Visual C++ now supports just-in-time (JIT) debugging. You can attach the JIT debugger either to a running process or to a process that has stopped responding. When JIT debugging is enabled, unhandled exceptions launch the JIT debugger instead of ending the program. You can then view the state of the program before execution stops. For information on setting up the JIT debugger, see Enabling JIT Debugging. For information on attaching the JIT debugger to running processes or to processes that have stopped responding, see Attaching the Debugger to Processes.

Welcome to Microsoft C Run-Time Library for Windows CE .NET

The Microsoft C Run-Time Library for Windows CE supports a subset of the functions that are available in the full Microsoft C Run-Time Library for the desktop. Supporting only a subset of the full library enables Windows CE to run on devices which have more limited resources than desktop computers.

Generally, Windows CE supports the wide-character versions of functions rather than the other versions; this enables Windows CE to support Unicode while keeping the library compact. Many function groups have the naming pattern **strfunction**, **wcsfunction**, **mbsfunction**. **strfunction** is the traditional string function, **wcsfunction** is the version of the function for wide-character strings (such as Unicode strings), and **mbsfunction** is the version of the function for multibyte-character strings. Similarly, many function pairs have the naming pattern **function**, **wfunction**; **wfunction** is usually a wide-character version of the function. The Coredll.lib and Corelibc.lib library files contain the C run-time library functions. The Microsoft Windows CE operating system run-time library does not support American National Standards Institute (ANSI) C or POSIX. All run-time library routines included with this product are compatible with the Win32 API. The C run-time libraries support Windows 95 and Windows NT, but not Win32s. Windows 95 and Windows NT support the Win32 Application Programming Interface (API), but only Windows NT provides full Unicode support. In addition, any Win32 application can use a multibyte character set (MBCS). The description of each run-time routine in this book includes a list of the required and optional include, or header (.H), files for that routine. Required

header files need to be included to obtain the function declaration for the routine or a definition used by another routine called internally. Optional header files are usually included to take advantage of predefined constants, type definitions, or inline macros.

MFC for Windows CE

The Microsoft® Foundation Class (MFC) library for the Windows® CE operating system is both a mature, comprehensive class library and a complete object-oriented application framework designed to help you build applications, components, and controls for Windows CE-based platforms. You can use the Microsoft Foundation Classes for Windows CE to create anything from a simple dialog box-based application to a sophisticated application that employs the full MFC document/view architecture. You also can use MFC for Windows CE to create full-featured Microsoft® ActiveX® controls and ActiveX containers.

This guide, which is a subset of the documentation provided for eMbedded Visual C++, targets application developers who have experience using MFC for desktop applications, and consequently does not delve into the philosophy, structure, or mechanics of MFC. If you do not have previous MFC experience, consider studying material devoted to MFC programming, such as the second edition of Jeff Prosise's *Programming Windows with MFC*, available from Microsoft Press.

The MFC for Windows CE library versions included in this package are designed to be used with the Microsoft® eMbedded Visual C++® 4.0 toolkit. The toolkit contains all the development tools and wizards you need for building MFC for Windows CE applications.

Appendix C

```
VoiceCall::Step()
{
   // Check if the sampling interval has passed
   if (samplingtimer->HasTimeOut())
   {
      // Get a sampled block and start sampling again
      sampleinput->GetSampleBlock(&inputblock);
      sampleinput->StartSampling();
      // Get a block from the mixer and play it
      mixer->GetSampleBlock(&outputblock);
      sampleoutput->Play(&outputblock);
      // Restart the timer
      samplingtimer->RestartTimer();
      // Prepare block for transmission and transmit it
      if (add3Dinfo)
         location3d->Add3DInfo(&inputblock);
      if (compressblock)
         compressor->Compress(&inputblock);
      transmitter->SendBlock(&inputblock);
      // Adjust the current sample offset
      mixer->GetSampleOffset(&sampleoffset);
      transmitter->SetSampleOffset(sampleoffset);
      // Poll for incoming data
```

```
        transmitter->Poll();
        // Add input from the connection to the mixer
        for (each voice source)
        {
           while (this source has data available)
           {
              // Get some data and process it
              transmitter->GetSampleBlock(&block);
              if (decompressblock)
                 decompressor->Decompress(&block);
              if (add3Deffects)
                 transform3d->Create3DEffect(&block);
              // Send processed data to the mixer
              mixer->AddBlock(&block);
           }
        }
     }
  }
```

Bibliography

[1] Cox, D.C., Research toward a wireless digital loop. *Bellcore Exchange,* 2, 2–7, Nov./Dec. 1986.

[2] Cox, D.C., Wireless loops: what are they. *Intl. J. Wireless Inf. Net.,* 3(3), Plenum Press, 1996.

[3] Cox, D.C., Gifford, W.G., and Sherry, H., Low-power digital radio as a ubiquitous subscriber loop. *IEEE Comm. Mag.,* 92–95, Mar. 1991.

[4] Cox, D.C., Wireless network access for personal communications. *IEEE Comm. Mag.,* 96–115, Dec. 1992.

[5] Padgett, J.E.,Hattori, T., and Gunther, C., Overview of wireless personal communications. *IEEE Comm. Mag.,* 28–41, Jan. 1995.

[6] Cox, D.C., Personal communications—A viewpoint. *IEEE Comm.Mag.,* 8–20, Nov. 1990.

[7] Goodman,D.J., Trends in cellular and cordless communications. *IEEEComm.Mag.,* 31–40, Jun. 1991.

[8] Schneideman, R., Spread spectrum gains wireless applications.*Microwaves and RF,* 31–42, May 1992.

[9] Steele, R., Deploying personal communications networks. *IEEE Comm.Mag.,* 12–15, Sep.1990.

[10] Special issue on wireless personal communications. *IEEE Comm. Mag.,* Jan. 1995.

[11] Special issue on advanced mobile phone service (AMPS). *Bell System Tech. J.,* (BSTS). 58, Jan. 1979.

[12] http://grouper.ieee.org/groups/802/11/

[13] http://www.wi-fi.org/

[14] http://www.wi-fiplanet.com/columns/article.php/961181

[15] http://www.wi-fiplanet.com/tutorials/article.php/1009431

[16] http://standards.ieee.org/wireless/overview.html

[17] http://www.bluetooth.com/

[18] http://www.bluetooth.org/

[19] http://www.zigbee.org/

[20] Hodges, M.R.L., The GSM radio interface. *Br. Telecom Tech. J.,* 8(1), 31–43, 1990.

[21] Chen, K.-C. and Lee, C.H., RAP: a novel medium access control protocol for wireless data networks. *Proc. IEEE GLOBECOM'93,* IEEE Press, Piscataway, NJ, 08854. 1713–1717, 1993.

[22] Goodman, D.J., Cellular packet communications. *IEEE Trans. on Comm.,* 38(8), 1272–1280, 1990.

[23] Barke, A. and Badrinath, B.R., I-TCP: indirect TCP for mobile hosts. Tech. Rept. DCS-TR-314, Dept. Computer Science, Rutgers University, Piscataway, NJ, 1994.

[24] Glisic, S.G., 1-Persistent carrier sense multiple access in radio channel with imperfect carrier sensing. *IEEE Trans. on Comm.,* 39(3), 458–464, 1991.

[25] Tobagi, F. and Kleinrock, L., Packet switching in radio channels: Part I carrier sense multiple access models and their throughput delay characteristic. *IEEE Trans. on Comm.,* 23(12), 1400– 1416, 1975a.

[26] Tobagi, F. and Kleinrock, L., Packet switching in radio channels: Part II the hidden terminal problem in CSMA and busy-one solution. *IEEE Trans. on Comm.,* 23(12), 417–1433, 1975b.

[27] Chen, K.-C., Medium access control of wireless LANs for mobile computing. *IEEE Network,* 8(5), 50–63, 1994

[28] Ioanidis, J., Duchamp, D., andMaguire, G.Q., IP-based protocols for mobile internetworking. *Proc. of ACM SIGCOMM'91,* ACM Press, New York, NY, 10036 (Sept.), 235–245, 1991.

[29] Teraoka, F. and Tokoro, M., Host migration transparency in IP networks: the VIP approach. *Proc. of ACM SIGCOMM,* ACM Press, New York, NY, 10036 (Jan.), 45–65, 1993.

[30] Singh, S., Quality of service guarantees in mobile computing. *J. of Computer Comm.*, 19, 359–371, 1996.

[31] Ghai, R. and Singh, S., An architecture and communication protocol for picocellular networks. *IEEE Personal Comm. Mag.,* 1(3), 36–46, 1994.

[32] Seal, K. and Singh, S., Loss profiles: a quality of service measure in mobile computing. *J. Wireless Networks*, 2, 45–61, 1996.

[33] R. Pandya, “Emerging mobile and personal communication systems,” *IEEE Communications Maga-zine*, Vol. 33, pp. 44–52, June 1995.

[34] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: a transport protocol for real-time applications,” Request for Comments 1889, Internet Engineering Task Force, Jan. 1996.

[35] M. Handley and V. Jacobson, “SDP: session description protocol,” Request for Comments 2327, Internet Engineering Task Force, Apr. 1998.

[36] H. Schulzrinne, A. Rao, and R. Lanphier, “Real time streaming protocol (RTSP),” Request for Comments 2326, Internet Engineering Task Force, Apr. 1998.