Improving Application layer IP QoS using fuzzy logic

RESEARCH THESIS

By

SHAFQAT NAZIR

2003-NUST-MS-PhD-CSE-223

Department of Computer Engineering
College of Electrical & Mechanical Engineering,
National University of Sciences and Technology
Rawalpindi
Session 2003-2004

Improving Application layer IP QoS using fuzzy logic

RESEARCH THESIS

Supervised By:
Col. Dr. Shoab A. Khan

Submitted By
SHAFQAT NAZIR

2003-NUST-MS-PhD-CSE-223

In partial fulfillment of the requirements for the award of

MS Software Engineering

Department of Computer Engineering
College of Electrical & Mechanical Engineering,
National University of Sciences and Technology
Rawalpindi
Session 2003-2004

ABSTRACT

Mostly, IP QoS solutions are developed from the entire network point of view i.e. developers tried to cover all the aspects of a huge network like scalability etc. Unfortunately, due to the cost and the complexity of such solutions, still the best-effort service is the main working solution. Our aim is to develop a performance diagnostic solution that continuously observes the performance of the network. Whenever the performance degrades to some threshold level, system automatically tries to resolve the performance bottleneck by using QoS principles and mechanisms. For performance measurement we used passive measurement using polling as data extraction technique.

There are two major problems in this regard;

- How we can measure the available capacity because due to sharing of common bandwidth it is constantly varying?
- How can we reduce the number of measurement samples needed to construct the network history?

To address first problems our solution uses fuzzy logic, because due to the complexity of solution it is difficult to establish precise values. Hence, we generalized some fuzzy rules and based upon this fuzzy rule base we tried to attain peak performance.

To address the second problem, adaptive sampling is used, so when certain pre-determined conditions are met, such as an increase in drop rate, corresponding actions are taken such as a decrease in sampling interval and vice versa.

DEDICATION

I dedicate all the successful efforts to my parents,

especially my father,

Major (retd.) Muhammad Nazir Tubassam (Late),

who always encouraged and supported me to excel in the educational career.

MAY ALLAH ALMIGHTY BLESS THE DEPARTED SOUL WITH ETERNAL PEACE..

ACKNOWLEDGEMENTS

First and foremost, for this entire happening, I am grateful to almighty ALLAH, who blesses with Immense-Peace and guided me through all this process.

I wish to express my thanks to the Brig. Dr. Muhammad Younus Javed, HOD, DCE, who proved to be the agent for stream lining the whole process. Considerable thanks to my supervisor, Lt Col. Dr. Shoab A Khan, for providing me with the opportunity to conduct research with him, for his advice, encouragement and suggestions for improvement. I have learned a lot from him in the past. During the past years, Dr Younus Javed and Dr Shoab A khan have served as both an excellent mentors and instructors, being both very helpful and attentive whenever I have needed advice.

Thanks also to Lt Col. Sajid Nazir and Mr. Muhammad Kaleem for their comments and suggestions. They have been an invaluable source of practical advice and guidance.

Last but not least, I am grateful to my parents who encouraged me to get admission in this course. I cannot express enough gratitude for the sacrifices they have made throughout my life. Early death of my father gave me a set back. I am also grateful to other family members and friends for being the part of source of motivation.

In addition, I would like to thank the entire faculty of the Computer Engineering Department for their guidance and support throughout my studies at The College of Electrical & Mechnical Engineering (CEME).

SHAFQAT NAZIR

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

FOREWORD

IP networks give no transmission time bounds. Packet source sends its packets and does not know exactly when its packets will reach the destination. Time taken from source node to destination node depends entirely up on the state of network. IP networks are best-effort networks i.e. they try their best to deliver the packets to destination but they don’t give any hard guaranties. During the early days, network provisioning was driven by the simple rule that add more bandwidth whenever there is congestion. Sharing of available bandwidth at that time was left to the random laws of statistical multiplexing. As long as the available bandwidth along any one link was significantly larger than the average traffic load, all the end-users were generally satisfied.

Now, times have changed. As more and more corporate IP intranets and commercial IP backbones are facing demands for more predictability in their network's end-to-end behavior due to factors ranging from the explosive growth of PCs running network-based multimedia applications to corporations attempting the migration of their mission-critical applications from proprietary to IP-based networks, network administrators need to constantly monitor the protocol and application usage that make extensive use of the network bandwidth and hence considerably affect overall performance.

Within this context the ability to observe and manage the dynamic functioning of the network becomes critical. One can't manage what he can't see, and network performance measuring tools provide that visibility. Using this visibility one can analyze the existing Quality of Service (QoS) and based on this analysis desired QoS can be applied.

Hence, there is a need to develop some sort of a simple QoS solution using existing best-effort service for end-user, the person for whose satisfaction the entire QoS paradigm has been created. Although using existing infrastructure, the solution should enhance performance by making adjustments according to the network state.

Suppose that an end-user QoS solution is able to diagnose performance bottlenecks by managing and scheduling its traffic according to QoS principles and mechanisms. And, if every node, somehow, applies some sort of same paradigm, the performance of the core network will gradually improve.

Keywords

QoS	=	Quality of Service
TR	=	Transmission Rate
DR	=	Drop Rate
RED	=	Random Early Detection
ARQ	=	Automatic Repeat Request
SNMP	=	Simple Network Management Protocol
IS	=	Increase Slight
IE	=	Increase Exponential
NC	=	No Change
DS	=	Decrease Slight
DL	=	Decrease Large
dTR	=	Amount of change in Transmission Rate
WinSock2 API	=	Windows Sockets 2 Application Programming Interface.

CHAPTER 1

Introducing Quality of Service (QoS)

Quality of Service (QoS) is described as “The collective effect of service performance, which determines the degree of satisfaction of a user of the service” [1].

QoS, which is closely related to and is sometimes referred to as Policy-based Networking, lets network managers define service policies that govern how much bandwidth goes to specific applications and end users. QoS makes it possible to implement policies across devices on LAN and IP networks. Converting best-effort IP into QoS capable network could mean huge savings for corporations. It would allow network managers to guarantee that revenue generating traffic gets more bandwidth compared to e-mail and casual web surfing, all with their current infrastructure.

Before delving into the details of QoS, an important question must be answered first: why should we be concerned with QoS at all when most networking problems can be solved by increasing capacity?

The main point is that data is inherently bursty, no matter how high the capacity, congestion will always occur for short periods. Another consideration is that routing protocols don't know about load levels, so congestion will build up on some paths while others have bandwidth to spare. There is also a speed mismatch at the LAN/WAN border. Even with faster WAN circuits, traffic from LAN is likely to congest the wide area link. Finally, bandwidth alone doesn't ensure low and predictable delay as even with huge bandwidth, there is still the danger that real-time applications will get stuck behind large file transfers. Therefore, QoS mechanisms are important because they enable networks to deliver defined levels of service with the existing network infrastructure.

1.1 QoS attributes and methods

We start by defining the characteristics that are controlled to provide a specific QoS and the two generic methods for providing QoS [1].

- Delay: Two aspects of delay have an impact on QoS: end-to-end delay, and delay variation or jitter. Interactive real time applications (e.g., voice communication) are sensitive to end-to-end delay and jitter. Long delays reduce the interactivity of the communication. Non-interactive real time applications (e.g., one way broadcast) are not sensitive to end-to-end delay but are affected by jitter. Jitter is usually accommodated by using a buffer at the receiver where the received packets are stored and then replayed at the appropriate time offset. Non real time applications are not delay sensitive.
- Throughput: Throughput is the amount of bandwidth available to an application. This determines how much traffic an application can get across the network. Throughput is also influenced by the link error rate and losses (often related to buffer capacity).

The fundamental idea in QoS is that traffic can be differentiated and provided different levels of service. The granularity of differentiation can be a small set of classes or individual flows. The amount of traffic allowed into the network is also controlled based on the available resources.

There are two generic methods for providing QoS:

Reservation-based

In this model, resources are reserved explicitly. The network classifies incoming packets and uses the reservations made to provide a differentiated service. Typically, a dynamic resource reservation protocol is used, in conjunction with admission control, to make reservations.

Reservation-less

In this model, no resources are explicitly reserved. Instead, traffic is differentiated into a set of classes, and the network provides services to these classes based on their priority. However, it is necessary to control the amount of traffic in a given class that is allowed into the network, to preserve the quality of service being provided to other packets of the same class.

1.2 IP QoS approaches

Many approaches have been suggested for providing QoS in the Internet Protocol. There are following main approaches for providing QoS:

- Type of Service (ToS) Routing

  Network layer and reservation-less QoS. Earliest QoS mechanism.

- Integrated Services (IntServ)

  Transport layer, flow-oriented and reservation-based QoS mechanism.

- Differentiated Services (DiffServ)

  Transport Layer reservation-less QoS mechanism.

- Multi-Protocol Label Switching (MPLS)

  Link layer QoS mechanism.

1.2.1 TOS Routing

Internet is currently based on the best-effort paradigm, which is highly scalable in nature. However, it cannot provide the hard guarantee that is desired by most of the present-day time-critical bandwidth-intensive applications. Internet Protocol was first specified in RFC – 791 [2]. At that time TOS bits were provided in the IP header, but were not used by most of the routers. The focus then was on routing the packets from source to the final destination and not on providing QoS guarantees of any kind whatsoever. TOS byte format is interpreted in this manner as shown in Figure 1.1.

Bits 0 – 2 are used as precedence control.

Bit 3 represents Delay. 0 represents normal delay while 1 represent low delay.

Bit 4 represent throughput. 0 represents normal throughput and 1 represents high throughput.

Bit 5 represents reliability. 0 represents normal reliability while 1 represents high reliability.

Bits 6 and 7 are unused for the time being. These are set to zero for now.

The use of high throughput/high reliability/low delay links could cost more for the end-users. Early networks, which implemented the TOS routing, include the AUTODIN II, ARPANET, SATNET, and PRNET. These implementations had their own mappings from the type of service to the actual service in the networks.

As specified in Requirements for Internet Hosts for Communication layer [3] and the Requirements for Internet Hosts for Application and Support [4], both these suggest the use of TOS bits by the hosts. This RFC recommends the use of the first three bits in the TOS field for the precedence and all the remaining bits for the purpose of specifying the type of service required. However, the specifics of the bits in the TOS have not been dealt with in [3] and [4].

Integrated IS-IS [5] and the Open Shortest Path First (OSPF) [6] are two routing algorithms which are capable of TOS routing.

The integrated ISIS protocol provides IP Type of Service (TOS) routing, through use of the Quality of Service (QOS) feature of ISIS. This allows for routing on the basis of throughput (the default metric), delay, expense, or residual error probability. Note than any particular packet may be routed on the basis of any one of these four metrics. Routing on the basis of general combinations of metrics is not supported. The support for TOS/QOS is optional. If a particular packet calls for a specific TOS, and the correct path from the source to destination is made up of routers all of which support that particular TOS, then the packet will be routed on the optimal path.

Bit 0	Bit 1	Bit 2	Precedence Control
1	1	1	Network Control
1	1	0	Internetwork Control
1	0	1	CRITIC / ECP
1	0	0	Flash Override
0	1	1	Flash
0	1	0	Immediate
0	0	1	Priority
0	0	0	Routine

Bit 3	Delay
0	Low
1	Normal

Bit 4	Throughput
0	Normal
1	High

Bit 5	Reliability
0	Normal
1	High

Bit 6 - 7	Unused
0	Undefined
1	Undefined

Figure 1.1 Interpretation of TOS Byte in IP

However, if there is no path from the source to destination made up of routers which support that particular type of service, then the packet will be forwarded using the default metric instead. This allows for TOS service in those environments where it is needed, while still providing acceptable service in the case where an unsupported TOS is requested. The IS-IS retains the same format for the TOS byte as given in Internet Protocol [2]. The first three bits specify the preference and though it does not affect the route, it may affect certain aspects of packet forwarding. This routing algorithm also retains the same format for the TOS field specified in Internet Protocol [2].

As far as the OSPF routing algorithm is concerned, it calculates different routes to a destination for each IP type of service. That is, in other words, it constructs different trees for each TOS. In order to differentiate routes based on the TOS, costs are associated with the routes. The OSPF encodes the TOS in the link state advertisements.

Type of Service in the Internet [7] discusses extensively on the TOS routing. The TOS structure defined in [7] is shown in Figure 1.2. This structure is different from the one proposed in [2], in that the bit 6 is also used as a TOS bit. The first three bits (precedence bits) indicate the importance of the packet. The TOS bits indicate the tradeoffs between the delay, throughput, reliability and cost. The last bit is unused. Even though there is a difference in the TOS byte format, this scheme is designed to be compatible with the routing algorithms that were just discussed. The four bits of the TOS byte are treated in the following manner.

This specification considers the TOS field as an integer rather than as a sequence of bits. This means that a request can be made only for one of maximizing throughput / minimizing delay / maximizing reliability / minimizing cost.

Bit 3	Bit 4	Bit 5	Bit 6	Interpretation
1	0	0	0	Minimize delay
0	1	0	0	Maximize throughput
0	0	1	0	Maximize reliability
0	0	0	1	Minimize monetary cost
0	0	0	0	Normal service

Figure 1.2 Redefinition of TOS field (RFC-1349)

1.2.2 Integrated Services (IntServ)

IntServ [8, 9] is a flow oriented reservation-based QoS mechanism. It reserves resources explicitly for individual flows using a dynamic signaling protocol and employs admission control, packet classification, and scheduling to achieve the desired QoS. A flow is defined as a stream of packets that originate from a specific user activity e.g. a single application session. A flow may be identified by a variety of mechanisms: IPv4 uses source and destination IP address and the destination port number. IntServ reserves bandwidth for individual flows to provide QoS.

Two key features lie at the heart of Intserv architecture:

1.2.2.1 Resources Reservation

IntServ requires that resources be reserved for flows in order to provide the requested QoS. This can be done via a dynamic reservation protocol, manual configuration, or by using a network management protocol. IntServ is not tied to any specific mechanism. However, RSVP is a protocol designed to provide resource reservations and it is designed to work with IntServ, though it can be used with other service models as well.

Two important characteristics of RSVP are:

Receiver oriented

The protocol requires receivers to make reservations. Receivers request resource reservations based on senders traffic specifications and path characteristics.

No built-in mechanisms for routing or packet scheduling

RSVP is just a signaling protocol. It relies on IP to compute the reservation route. RSVP is also not concerned with how nodes implement the reservation requests (e.g., admission control, packet classification, packet scheduling).

1.2.2.2 Call Setup / Call admission

A session requiring QoS guarantees must first be able to reserve sufficient resources at each network router on its sources-to-destination path to ensure that its end-to-end QoS requirement is met. This call setup process requires the participation of each router on the path. Each router must determine the local resources required by the session, consider the amounts of its resources that are already committed to other ongoing sessions, and determine whether it has sufficient resources to satisfy the per-hop QoS requirement of the session at this router without violating local QoS guarantees made to an already-admitted session.

Steps involved in call admission are as follow;

Traffic characterization and specification of the desired QoS

In order for a router to determine whether or not its resources are sufficient to meet the QoS requirements of a session, that session must first declare its QoS requirement, as well as characterize the traffic that it will be sending into the network, and for which it requires a QoS guarantee. In the Intserv architecture, the so-called Rsepc (Reservation specifications) defines the specific QoS being requested by a connection; the so-called Tspec (Traffic specifications) characterizes the traffic the sender will be sending into the network. The specific form of the Rspec and Tspec will vary, depending on the service requested.

Signaling for call setup

A session's Tspec and Rspec must be carried to the routers at which resources will be reserved for the session. In the internet, the RSVP protocol is the signaling protocol of choice.

Per-element call admission

Once a router receives the Tspec and Rspec for a session requesting a QoS guarantee, it can determine whether or not it can admit the call. This call admission decision will depend on the Tspec, the requested type of service, and the existing resource commitments.

Two main service categories are defined based on the delay and loss requirements.

1.2.2.3 Guaranteed QoS

Guaranteed Delay service provides absolute guarantees on the delay and loss experienced by a flow. Packets are not lost, nor do they experience delay exceeding the specified bound. These firm guarantees are provided using resource reservations.

1.2.2.4 Controlled-Load Network Service

Controlled Load service provides service equivalent to that of an unloaded network. Most packets are not lost, nor do they experience queuing delay. However, Controlled-Load service makes no quantitative guarantees about performance i.e. it does not specify what constitutes a very high %age of packets nor what QoS closely approximates that of an unloaded network element.

The Controlled-Load service targets real-time multimedia applications that have been developed for today's Internet. These applications perform quite well when the network is unloaded, but rapidly degrade in performance as the network becomes more loaded.

1.2.2.6 Limitations of IntServ

The most important concern about IntServ is whether it can scale to large backbones.

- The number of individual flows in a backbone network can be very large. The number of control messages for making resource reservation for large number of flows can be large and may require a lot of processing power. Similarly, maintaining state information for all the flows can require a lot of storage capacity. There is also a need to classify a large number of packets and schedule numerous queues making the router extremely complex.
- Policy issues need to be resolved to determine who can make reservations. Similarly, security issues need to be resolved to ensure that unauthorized sources do not make spurious reservations.
- It is believed that IntServ is appropriate for small intranets where there are a small number of flows and where policy and security issues can be managed easily. Large backbone networks will need more scalable mechanisms for differentiating traffic and providing differentiated services to them.

1.2.3 Differentiated Services (DiffServ)

DiffServ [8] is a reservation-less mechanism. No explicit resource reservation or admission control is used, though the network has to use some method to differentiate traffic.

The key features of DiffServ that overcome some of the limitations of IntServ are:

Coarse Differentiation

DiffServ does not differentiate per flow traffic. Instead there are a small number of well defined classes which are provided differentiated services according to their priority.

No Packet Classification in the Network

In RSVP, each router implements packet classification in order to provide different levels of service. To make the solution more scalable, in DiffServ, packet classification is moved to the edge of the network. Edge routers classify and mark packets appropriately. Interior routers simply process packets based on these markings. This implies that interior routers do not recognize individual flows, instead, they deal with aggregate classes.

Static Provisioning

IntServ requires dynamic resource reservation. This may result in a large number of control packets. It also requires dynamic admission control in each router. DiffServ moves admission control to the edge of the network. It also does not require dynamic reservations. Instead, it uses long-term static provisioning to establish service agreements with the users of the network, whose traffic it can police at the ingress to the network.

No Absolute Guarantees

The guaranteed delay service in IntServ provides hard bounds on the delay experienced by packets by explicitly reserving resources along the path. DiffServ, in general, does not provide hard guarantees. The goal in DiffServ is to monitor the traffic that enters the network at the ingress node and check for compliance against some predefined service profiles. Based on this, packets can be marked as being "in" or "out" of their profiles. Inside the network, routers preferentially drop packets that are tagged as being “Out”.

In addition to drop precedence, there is additional information in the packet headers that communicates the type of service (TOS) desired by the packet. For instance, a premium service can offer the equivalent of a Constant bit rate (CBR) connection. Similarly, another example is that of an assured service that is characterized by bursty behavior and is provisioned using expected capacity; hence its bandwidth is allocated statistically.

Service Profiles

A service profile indicates which traffic is to be treated differentially and what type of service is requested. The former may be indicated by setting a packet filter based on packet header fields such as IP addresses. The latter can be specified using a token bucket filter. The service profiles are set up at the edge nodes based on customer subscriptions. They are therefore relatively static. The decision to accept a new subscription can be made centrally based on knowledge of network topology and capacity. Thus a network provider can provision its network according to the expected demand and subscriptions.

Packet Classification and Marking

The ingress router must check all received packets against service profiles to check if a packet should receive differential treatment. Packets that do not meet service profiles can either be discarded or sent into the network with higher drop precedence. The source can also police and shape the traffic it is offering to the network in order to maximize the probability that the traffic will meet the service profile and receive the desired quality of service. The ingress router marks the packets as they enter the network so that interior routers can handle the packets differentially. The marking use header fields, for example for IPv4 packets, the TOS octet is used.

Differential Queuing

Differentiated packets have to be handled differently. In order to do do so, the interiors router employ multiple queues with Class Based Queuing (CBQ). Generally, delay-sensitive traffic needs to be serviced sooner, and loss-sensitive traffic needs to be given larger buffers. The loss behavior can also be controlled using various forms of Random Early Detection (RED). These methods use probabilistic methods to start dropping packets when certain queue thresholds are reached, in order to increase the probability that higher priority packets can be buffered at the expense of more dispensable packets. For example, packets of different service types are put into different queues, and within a given service type, packets with higher drop precedence are discarded earlier than those with lower drop precedence.

1.2.4 Multi-Protocol Label Switching (MPLS)

Although not primarily a QoS mechanism, MPLS can be an important tool for backbone service providers. The MPLS approach to IP QoS is different from DiffServ. MPLS uses fields in the 32-bit (4-byte) label it adds to the IP packet. This label is intended to improve efficiency of the router network and allow routers to forward packets using predetermined paths according to, among other things, specified QoS levels. At the edge of the MPLS network, a label is added to each packet containing information that alerts the next hop MPLS router to the packet's predefined path. As the packet traverses the network, it may be relabeled to travel a more efficient path. Upon leaving the MPLS network the packet is stripped of its label and restored to its original size.

MPLS attempts to set up paths in a network along which packets that carry appropriate labels can be forwarded very efficiently since the forwarding engine would not look at the entire packet header, rather only at the label and use that to forward the packet. This not only allows packets to be forwarded more quickly, it also allows the paths to be set up in a variety of ways: the path could represent the normal destination-based routing path, a policy-based explicit route, or a reservation-based flow path. Ingress routers classify incoming packets and wrap them in an MPLS header that carries the appropriate label for forwarding by the interior routers. The labels are distributed by a dynamic Label Distribution Protocol (LDP), which effectively sets up a Label Switched Path (LSP) along the Label Switched Routers (LSR).

When a packet enters the MPLS network, a Label Switched Router (LSR) may analyze the IP header to determine its desired service level. As with addressing, the MPLS network will read this information only once. The label also contains 3 bits, called experimental bits, that may be used for specifying QoS. These bits will permit the Label Switched Routers (LSRs) to examine a packet's required service level and handle it accordingly. MPLS also permits explicit routing, where the hops a packet will take are specified in advance and the label is used to indicate this route. Explicit routing is a useful capability for allowing QoS and enabling network managers to set up defined paths through the MPLS network that apply to certain traffic streams.

CHAPTER 2

Strategy for application layer QoS

From the point of view of application layer's QoS if we check the strategy of existing performance software. They usually gather the data that we request, this data is used for construction of network history and to check the trends such as traffic analysis, top users, applications, protocol usage etc. Application layer QoS solution can use this information to prevent network problem before it arises by detecting network bottlenecks, making timely informed decisions during a crises or failure and planning access levels for different services according to some constraints e.g. what if MP3 music downloads use 65% of your bandwidth budget? Or file transfers and print traffic (important but not time-sensitive) monopolize your link while SAP and Oracle are starved for bandwidth?

An application layer IP QoS solution should be able to diagnose performance bottlenecks by managing and scheduling traffic according to QoS principles and mechanisms. And, if every node, somehow, applies some sort of same paradigm, the performance of the core network will gradually improve.

The basic idea behind proposed application layer QoS strategy is illustrated in Figure 2.1. The following separated functional entities can be extracted from Figure 2.1:

- Traffic measurement

  The basic level or the first stage of our proposed solution is traffic measurement. General rule is that the more measurement points, the more accurately the network behavior can be determined, while the analysis also gets more complicated.

  Both active and passive measurements have their merits and demerits, theme of our proposed strategy is to diagnose the performance degrade regardless of whatsoever is the performance measurement technique.

  It is very important that the measurement tool has minimal effect to the traffic itself otherwise it can cause a bottleneck by itself. At this stage some preliminary processing can be done to the measurement information as well.

- QoS analysis

  In the next stage data provided by traffic measurement stage is used to calculate the actual QoS metrics. In practice, this means, for example, delay calculation of single packets traveling through measurement points. There usually exists a single QoS

analysis entity in the measurement system. We are more interested in net performance of over all contending traffic and not in the complicated per flow network performance details. Therefore we shall focus on simple performance parameters for analysis so that analysis stage couldn't be a bottleneck.

- QoS management

  In the final stage, using the information generated from QoS analysis, ultimate aim is to implement QoS principles and mechanisms. This information can simply be a small database for more accurate post analysis purposes, or the information might be in real-time format, in which case some sort of real-time QoS monitoring tool is needed. But the application should be able to manage the QoS to satisfy the requirements.

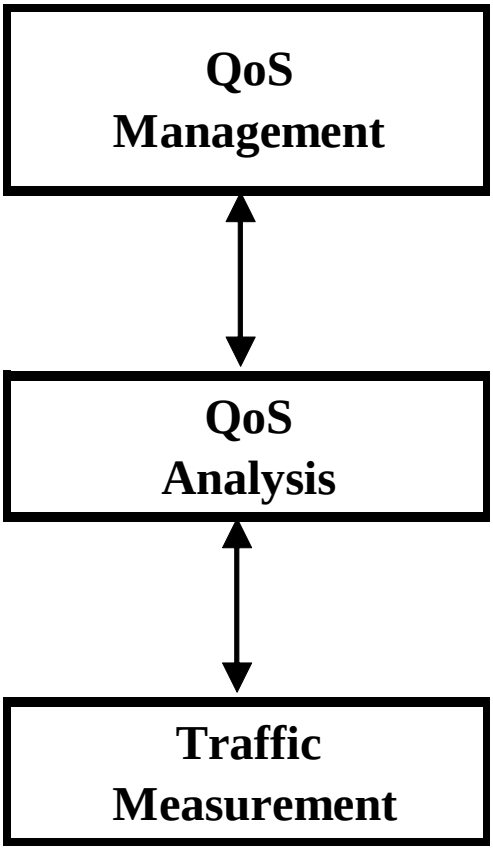


Figure 2.1 Basic strategy for applying QoS

2.1 Traffic Measurement

Measuring network quality of service (QoS) is basically very close to network traffic measurements. In fact, QoS measurements lie usually on top of traffic measurements. The difference is the way how the measurement results are analyzed. In network traffic studies, the interesting thing is the traffic itself and its effects: network load, queuing performance, source traffic processes, large scale traffic flow models. Especially the analysis of traffic processes and models requires accurate information of collected traffic. In QoS analysis, on the other hand, network traffic itself is not the interesting thing, but it is rather just used as a tool to reveal the performance characteristics (delay, maximum throughput, etc.) of links, networks and systems.

Generally, the measurement methods can be categorized to passive and active methods. These can be further divided to probing, tracing and monitoring [11].

Tracing is usually active (the traffic is specially generated for the measurements) and the purpose is often to trace the performance of a certain route hop by hop. Monitoring, on the other hand, usually refers to passive methods, where information about the existing network traffic passing through some nodes (routers, servers, etc.) is collected. Probing can be either active or passive. Active probing measures the effects of specially generated test flows, on the other hand passive probing collects information about the effects of existing network traffic.

2.1.1 Active Measurement

Active measurement techniques mainly consist of software programs that use the network and then try to analyze its performance through direct measurement observation by introducing data into the network strictly for the purpose of sampling bandwidth; this is also referred to as probing. Injected traffic takes the form appropriate to the subject of investigation e.g. ICMP ping packets to establish reachability, HTTP requests to monitor server response times etc. The data gathered is largely pertinent only to the subject of investigation, and may be discarded at the time of collection, or may be stored for global analysis. The different service quality metrics commonly assessed by directly measuring the network's response to the injected stimulus can include end-to-end delay, packet loss, bulk throughput, end-to-end path properties, and bandwidth characteristics.

Limitations

There is always the risk that the injected traffic, being obtrusive, may in itself colour results or that incorrect or inappropriate framing may produce misleading data. Active techniques result

in spurious traffic patterns, exacerbate network congestion, and hinder application performance.

Due to the generation of specific types of traffic, based on which measurements are conducted, active measurement techniques suffer the inherent limitation of only being able to measure the performance experienced by this special purpose traffic during the measurement interval.

- ICMP-based active measurements

  ICMP-based active measurements mainly suffer from the special response and treatment that ICMP packets might elicit from the network. Network operators can give lower or even higher priority to ICMP packets from the rest of the traffic, in order to minimize their impact. At the same time, QoS strategies deployed on access AS routers can give ICMP packets lower priority in order to increase the performance of other types by giving higher priority to TCP and UDP packets. Core network nodes can be configured to even drop ICMP packets and not processing them at all. Additionally, the use of ICMP packet in certain types of security attacks has made operators to often block ICMP packets from entering their networks.

- Dedicated measurement protocols

  Dedicated measurement protocols like the IPMP can actually be characterized as measurement-oriented enhanced versions of ICMP. They deploy echo request-reply mechanisms similar to ICMP, but their header fields are defined to provide space for carrying more extensive and informative measurement data. IPMP packets can carry path records identifying all the network nodes a packet has visited, and also provide space for a 64-bit timestamp to be inserted at each node. However, the main limitation influencing the accuracy of ICMP-based measurements also applies to dedicated measurement protocol traffic.

- UDP-based active measurements

  In UDP-based active measurements, test traffic packets are usually between a few tens and a few hundreds of bytes long which can result in bandwidth starvation of competing TCP flows, whose performance can greatly decrease due to backing-off in response to packet drops. At the same time, the performance of UDP flows can stay nearly constant, falsely reporting a better level of service that the network actually delivers. Competing UDP flows can also experience randomly different performance

than that reported by the UDP-based test traffic, depending on their sending rates, the queuing disciplines and the network load during the measurement intervals.

- TCP-based active measurements

  The end-to-end performance observed by TCP transfers can match closely the service experienced by the users of the network. However, the TCP protocol behavior is complex and in the presence of different TCP implementations on different end-systems, it proves difficult to determine which facets of the overall connection behavior are due to the state of the network path, and which are due to the behavior of the TCP implementations at the end-points. Also, the complex and CPU intensive operations within system's TCP stacks introduce additional components (mainly) of delay which are difficult to distinguish from genuine network pathologies. The reliable stream transport service provided by TCP demands data to be exchanged in both directions of an individual connection i.e. TCP's data (forward path) and the small acknowledgement packets (reverse path). The different characteristics of the traffic in the forward and reverse TCP paths, as well as the required knowledge of the internals of individual TCP implementations, makes the use of TCP by infrastructures for scheduled test-traffic measurements a challenging task.

2.1.2 Passive Measurement

Another way to gather the traffic data is to choose an appropriate site and passively capture every IP packet through it. It is important to choose a good monitoring point. Passive measurement techniques only monitor existing network traffic in a non-intrusive manner. Therefore, passive measurement techniques circumvent both contention and application performance problems. Data may be directly gathered from links (on-line monitoring) using probes (e.g. tcpdump) to observe passing traffic, or from attached hosts (usually routers/switches or servers, e.g. netflow, HTTP server logs).

Data thus collected is typically stored for later analysis (e.g. long term trends of traffic etc) but may be the subject of real-time analysis, or forwarded to collection / analysis servers for further processing.

Limitations

Although passive measurement systems monitor the operational traffic flows, they are inadequate of targeting the service experienced by the actual (end-to-end) application traffic; what they can ultimately measure is an approximation of the service offered by a single “network cloud”, as seen from an operator’s (and not enduser/application’s) perspective.

Operating at a single point in the network allows passive techniques to monitor data and update relevant counters at certain network nodes and links. However, correlating such partial information from multiple monitoring points to reveal the performance of traffic carried between two (or more) Points-of-Presence (PoPs) in the network is a computationally challenging task, that consumes vast amounts of processing and network resources, and cannot be undertaken in real-time.

SNMP is widely implemented into network nodes and adopts an element-centric view of network management, primarily focusing on reporting the operational status of the network elements, preventing and isolating fault conditions, and restoring normal operation in case of equipment failures. Even the traffic-related variables maintained by SNMP agents only maintain counters describing statistics relevant to the implementation and execution of parts of the networking stack at a node, and not how they relate with the actual traffic flows and their properties.

RMON provided a great enhancement for SNMP, facilitating remote and distributed monitoring of LANs. It can ideally provide for high-level traffic statistics and filtering mechanisms tuned to monitor specific traffic flows and packets. In fact, it has been stated that if RMON were universally deployed, the need for additional packet and flow monitoring support would be

obviated. However, their implementation is so costly and infeasible for high-speed backbone interfaces that they can only be partially realized at low speed router interfaces, and hence be relegated at the edge of the network.

2.1.2.1 Data Extraction Techniques for passive measurement

Data extraction techniques for passive measurement can be classified into two

- Event-Driven

  In event-driven technique system automatically supplies data to the monitoring tool. Monitoring program simply has to wait and listen for data, whenever some new information is available; system automatically sends it to the monitoring system in form of an event. Upon receiving new event the monitoring system takes the appropriate action. Event-driven data extraction in this way is real-time or on-line monitoring. Advantage of event-driven data extraction is that it enables presentation and analysis of information as it happens.

  Disadvantage is that during high activity time there is a need to efficiently handle the monitoring data.

- Polling

  In this technique, the monitoring tool has to ask the system for data (normally at specific time intervals). Polling in this way is non real-time technique, unless the data is polled very frequently, which in turn cause performance overhead.

2.1.2.2 Packet Capture Libraries for Passive Measurement

Packet capture libraries provide interfaces for applications such as network analyzers for straightforward packet capture from desired protocol level.

Libpcap

Libpcap is a packet capture library, which provides access to the data link layer. It has functions to dump the binary packet contents to a file, to read the dumped file and a possibility to filter the capture. Libpcap works in UNIX environments.

“libpcap, provides implementation-independent access to the underlying packet capture facility provided by the operating system. Currently, it supports only the reading of packets (although adding a few lines of code to the library lets one write datalink packets too)."

Many network monitoring software use libpcap for capturing the packets, Ethereal for example.

WinPcap

WinPcap is an open source packet capture and network analysis library for Win32 platforms. It has a kernel-level packet filter, a low-level dynamic link library (packet.dll), and a high-level and system-independent library (wpcap.dll).

Packet.dll is used as an API, which offers the packet driver functions independent from the Windows OS. Wpcap.dll offers libpcap compatible high-level capture primitives, which makes the capture independent of the underlying operating system.

For traffic measurement purpose our solution uses offline passive measurement with polling as a data extraction technique. Polling is selected because it is easier data sampling and extraction of specified detailed information on request rather than collected regardless.

2.2 Basic Measurement Counters

We shall focus on the following four simple measurement counters:

Transmitted Datagrams (Trans-Dgrams)

This counter represents the overall departing IP traffic i.e. both UDP and TCP.

Transmitted datagrams discarded (Disc-Dgrams)

This counter represents the overall dropped datagrams after the departure.

Transmitted Segments (Trans-Segs)

This counter represents the departing TCP traffic.

Re-transmitted Segments (Re-Trans-Segs)

This counter represents the re-transmitted segments.

2.3 Calculated Counters

From basic measurement counters, two other counters are calculated:

Datagram Discard Percentage (Dgram Disc %age)

Percentage of discarded datagrams out of total transmitted datagrams, it gives us overall IP drop percentage.

Segment Re-transmission Percentage (Seg.Re-trans %age)

We also calculate the percentage of segment re-transmission out of total segments. It gives TCP drop percentage.

CHAPTER 3

QoS Analysis and Management

3.1 QoS analysis

Even though the focus is on QoS analysis, network analysis very important and is needed in a way as a tool for QoS analysis, i.e., QoS analysis operates logically above network analysis.

3.1.1 Network analyzers

Network analysis means the event of capturing the packets traveling in the network and analyzing them to conclude what is happening on the network. Network analysis software is used to decode the common protocols and show the network traffic in a form that is human-readable. The capability to set the network interface into promiscuous mode combined with the ability to read the packets from the data link layer gives application a chance to monitor all the packets on the local cable.

The following is a list of some of the common network analyzers:

Ethereal

Ethereal [12] probably one of the best sniffers available. Ethereal is a free open source network monitoring tool which has become one of the most popular products used. It has a graphical user interface, can decode more than 400 protocols and runs on both UNIX- and Windows based systems. Ethereal is described in more detail later.

Tcpdump

This UNIX-based sniffer is one of the oldest and widely used.

WinDump

A Windows version of tcpdump.

Analyzer

A free Windows-based sniffer from the developers of WinPcap and WinDump.

Packetyzer

This is a free Windows user interface for Ethereal and can decode more than 483 protocols. It is built with Ethereal and winpcap.

Carnivore

The codename for FBI's network analyzer. Carnivore is used to monitor network communication in connection with criminal investigation.

3.1.2 Higher layer QoS Monitoring Tools

Following are some of the QoS monitoring tools working at application-layer level.

ntop

ntop (network top) [13] is a simple, open source (GPL), portable traffic measurement and monitoring tool, which supports various activities, including traffic measurement, traffic characterization and monitoring, network optimization and planning, and detection of network security violations.

It best fits end-user needs like, there is no programming needed, easy to use and customize, standard Interface (Web, SNMP), open and Portable, and good performance and minimal resource requirements.

RMON2

With RMON2 probe it is possible to monitor and decode also the protocols operating higher than network layer. It sees above the IP layer and can read the encapsulated higher-layer headers. This provides application-level monitoring.

RTFM

Real-time flow measurement architecture was developed to monitor the network flows. A flow is identified by its source and destination addresses at various layers so this mechanism provides application-level monitoring as well.

RTCP

It is possible to perform end-to-end QoS monitoring with Real-time control protocol (RTCP) messages. They contain the fields such as timestamps, which can be used to calculate QoS parameters.

Name	Free Available	Solution	Open-source	Probe Technique	Data Extraction
NTop		Software		Passive	Packet Capturing (LibPACP)
RMon2	X	Software & Hardware	X	Active	
Network Probe		Software	X	Passive	Packet Capturing (WinPACP)
QCheck		Software	X	Active	Through Agents
RTFM	X	Software	X	Active	Through Agents
RTCP	X	Software	X	Active	Through Agents

Table 3.1 Higher Layer QoS monitoring tools.

3.1.3 QoS performance metrics

In QoS analysis stage, performance counters are analyzed against some QoS performance metrics.

Probably the most essential and general performance metrics are:

- End-to-end delay

  End-to-end delay is the average time it takes for a network packet to traverse the network from one endpoint to the other. Single-point measurements provide end-to-end performance information, which can be used to determine QoS. This setup enables the possibility to measure round trip time (RTT), i.e. the time from the initiation of the service request to the reception of the service reply. This information is valuable QoS metric and gives direct insight of the total performance of the system.

- Jitter

  Jitter is the variation in the end-to-end delay of sequential packets. It can be calculated from the exact delay measurements, but not from the average values. Jitter is easier to calculate than absolute delay, since it only needs the delay difference between sequential packets, but not absolute clock synchronization of the measurement points. Jitter can be controlled with buffers, but this is done at the expense of delay. Jitter has many causes, including: variations in queue length; variations in the processing time needed to reorder packets that arrived out of order because they traveled over different paths; and variations in the processing time needed to reassemble packets that were segmented by the source before being transmitted.

- Throughput

  The amount of data transferred from one end to the other end in a specified amount of time. Typically, throughputs are measured in Kbps, Mbps and Gbps.

- Reliability / Packet loss

  Reliability is measured in terms of packet loss, which in turn is measured as the percent of transmitted packets that never reach the intended destination. Network devices like routers, sometimes have to hold data packets in buffered queues when a link gets congested. If the link remains congested for too long, the buffered queues will overflow and data will be lost, assuming that the application that sent the packet will retransmit it. The lost packets must be retransmitted, adding, of course, to the total transmission time.

Network Characteristic	Description
End-to-end delay	Average time for a network packet to traverse the network from one endpoint to the other.
Jitter	The variation in end-to-end delay.
Throughput	Amount of data transferred from one end to the other in a specified amount of time.
Reliability	The percentage of packet loss

Table 3.2 Network Characteristics Managed by QoS

Jitter is also related to delay, since it is understood as the variation of the delay. Packet loss is more or less independent metric as compared to the other metrics even though some approximation of it can be drawn from raw throughput, if offered traffic load is known. Lost packets are usually detected via sequence numbers and ARQ-methods (automate repeat request). Packet loss has its effect to the other metrics. Consider e.g. an application layer jitter measurement, where jitter is measured directly from sequentially arriving packets. If lower layers fail to deliver some packets correctly, jitter is increased because of the gaps of missing packets. It might also be the case that the application layer packets are all delivered correctly, but jitter is still increased because of the erroneous lower layer packets and retransmissions provided by ARQ-methods. In this case the delay is also increased.

3.1.4 Selected measurement mechanism

For accurate QoS measurements, the effect of the network traffic itself must be included. Measured traffic having large traffic load of the measured network causes extra queuing delays, which in turn give false information about the real network performance. Therefore, one should be aware about the current load of the network when performing measurements. Otherwise no accurate decisions can be drawn from the basis of the measurements. On the other hand, the end user is only interested in end-to-end quality of service, regardless of the underlying technologies, i.e., the user is only interested that does the service work or not, and does not care what are the reasons behind bad service level.

In this respect, simple end-to-end performance measurements is needed requiring. Best suitable choice in this regard is offline passive measurement without causing any extra traffic load. It is a non-real-time technique to avoid performance overhead.

Polling as a data extraction technique is used because we want to extract only the specific detailed info when required, rather than having a huge database of unwanted data as well.

3.1.5 Main performance parameters

Our performance parameters are packet-drop rate and segment re-transmission rate.

Packet-drop rate

Packet-drop rate shows the overall performance degrades. It is calculated by the following formula:

Packet-drop rate = Datagram Discarded Percentage x W

Where W is the weight to cover up certain factors, like transient congestion and long-term congestion.

For lower packet counts, it has conservative value; on the other hand, it has aggressive value for higher packet counts.

Segment Re-transmission rate

It shows the TCP drop rate. Formula for its calculation is similar to the packet-drop rate calculation formula i.e.

Segment Re-transmission rate = Segment Re-transmission percentage x W

Here W has the same semantics as in case of packet-drop rate.

As an example W can be calculated as follow;

```
if   Packet count < 1 K  then
        W = 0
else if   Packet count < 10 K  then
        W = 1
        .
        .
else if   Packet count < 1 G  then
        W = 6
else
        W = 7
end if;
```

3.2 QoS management

Ultimate aim is to apply QoS principles and mechanisms to diagnose network performance. Network performance is directly related to congestion.

3.2.1 Congestion Control

Congestion [17] can be defined as a network state in which performance degrades due to the saturation of network resources, such as communication links, processor cycles, and memory buffers.

For example, if a communication link delivers packets to a queue at a higher rate than the service rate of the queue, then the size of the queue will grow. If the queue space is finite then in addition to the delay experienced by the packets until service, losses will also occur. Observe that congestion is not a static resource shortage problem, but rather a dynamic resource allocation problem. Networks need to serve all users requests, which may be unpredictable and bursty in their behavior (starting time, bit rate, and duration). However, network resources are finite, and must be managed for sharing among the competing users. Congestion will occur, if the resources are not managed effectively.

The effect of network congestion is degradation in the network performance. The user experiences long delays in the delivery of messages, perhaps with heavy losses caused by buffer overflows. Thus there is degradation in the quality of the delivered service, with the need for retransmissions of packets (for services intolerant to loss). In the event of retransmissions, there is a drop in the throughput—a wastage of system resources—and leads to a collapse of network throughput when a substantial part of the carried traffic is due to retransmissions (in that state not much useful traffic is carried). In the region of congestion, queue lengths, hence queuing delays, grow at a rapid pace.

Let us have a look at how we can measure or sense congestion and where.

Congestion can be sensed (or predicted) by:

Packet loss sensed by

- The queue as an overflow,
- Destination (through sequence numbers) and acknowledged to a user,
- Sender due to a lack of acknowledgment (timeout mechanism) to indicate loss.

Packet delay

- Can be inferred by the queue size,
- Observed by the destination and acknowledged to a user (e.g. using time stamps in the packet headers),
- Observed by the sender, for example by a packet probe to measure RTT.

Loss of throughput

- Observed by the sender queue size (waiting time in queue).

Other calculated or observed event though which congestion can be inferred

- Increased network queue length and its growth
- Calculated from measured data, e.g. queue inflow and its effect on future queue behavior.

Congestion control refers to the set of actions taken by the network to minimize the intensity, spread, and duration of congestion. Examples of network controls include admission and regulation of traffic flow into the network. These controls should be achieved without causing network congestion, or at least exhibit no degradation of performance beyond acceptable levels to the user, i.e. no degradation of quality of service in terms of loss and delay, and no appreciable reduction in throughput.

One major step in congestion control is keeping the average queue occupancy low because when operating with a high average occupancy, space available to absorb bursts reduces causing packet drops and hence re-transmissions. When a burst arrives while the queue occupancy is high, multiple packets may be discarded simultaneously. If the drops affect many TCP flows at one time, the congestion-avoidance behavior of all flows becomes synchronized and a severe drop occurs in average performance. Also the average latency experienced by traffic sharing a given queue increases as the average queue occupancy increases which results in increased end-to-end latency.

3.2.2 Congestion Avoidance used by transport layer

To control congestion at application layer we should focus on widely used congestion control mechanisms for transport layer. First priority for congestion control is congestion avoidance. Congestion can be avoided by decreasing queue occupancy because queue occupancy can be considered to reflect the level of congestion currently being experienced at the queue's output interface.

To reduce queue occupancy we need some method of triggering congestion-avoidance behavior in the transport protocols generating the flows passing through the queue. Queue management algorithm [10] should consider two basic types of congestion :

- Transient congestion

  Transient congestion occurs over a time periods shorter than the reaction time of congestion avoiding transport protocols.

- Long-term (average) congestion

  It results from the approximately steady-state rates of all the flows passing through the queue.

Transient congestion is caused by short, correlated bursts of traffic from one or more flows. Usually such queue sizes are used that can cover the typically expected burstiness. However, there is always the chance that a burst will fill the queue, at which point the packet drop is the only option available.

The average queue occupancy (measured over some recent time interval) is a significant issue when a queue is shared by multiple traffic flows because the average occupancy affects the latency experienced by all packets passing through that queue. A queue manager needs to continuously supply feedback to transport protocols to keep the long-term occupancy down.

In principle feedback can be applied in two ways:

- In-band marking of packets and ECN.
- Dropping of packets.

In-band marking of packets and Explicit Congestion Notification (ECN)

Although packet dropping is currently the preferred way to apply feedback (because TCP uses packet loss to trigger its congestion-avoidance behavior and also the packet dropping immediately reduces the downstream load), nondestructive methods of signaling congestion are being designed and evaluated. Dropping packets essentially wastes the resources used to get the packet up to the congestion point.

One example is known as explicit congestion notification (ECN). The two currently unused (CU) bits from the DiffServ field are redefined as the ECN Capable Transport (ECT) and Congestion Experienced (CE) bits. A transport protocol sender sets the ECT bit on outbound packets when it knows that both ends of the flow understand the meaning of the CE bit. If no congestion control feedback is required, the CE bit is ignored. When a router along the path wants to apply congestion control feedback, it has two options:

- If the ECT bit is set, set the CE bit.
- If the ECT is reset, drop the packet.

ECN is better for long-term congestion levels but not suitable for transient-congestion. Also from application layer's point of view it is useless.

Dropping of packets and Random Early Detection (RED)

The Internet Research Task Force (IRTF) developed this mechanism. RED [15] uses the queue's average occupancy as a parameter to a random function that decides whether congestion-avoidance mechanisms ought to be triggered (assume that the trigger is "packet drop"). As the average occupancy increases, the probability of a packet-drop action increases.

- For occupancy up to a lower threshold, minth, packets pass through untouched (drop probability is zero).
- Above minth, the probability of packet drop rises linearly toward a probability of maxp reached for occupancy of maxth.
- At and above maxth, packets are guaranteed to be dropped.

These three phases are sometimes referred to as normal, congestion avoidance, and congestion control, respectively. Worst-case long-term queue size is limited to maxth (where probability function jumps to 1).

Average occupancy is recalculated every time a packet arrives and is based on a low-pass filter, or exponentially weighted moving average (EWMA), of the instantaneous queue occupancy. Formula is

$$Qavg = (1 - Wq) \times Qavg + Qinst \times Wq$$

Qavg is the average occupancy, Qinst is the instantaneous occupancy, and Wq is the weight for the moving-average function. Wq affects how closely the average occupancy parameter tracks the queue's instantaneous occupancy, higher values are more aggressive, and lower values are more conservative. The goal is to pick a value that allows RED to ignore short-term transients without inducing packet loss but to react to sustained levels of occupancy before everyone's latency is adversely affected or multi-flow synchronization of TCP's congestion avoidance is felt.

Two key assumptions underlie drop-based active queue management:

- Many or most of the flows causing transient congestion are TCP-based and, therefore, respond to the negative feedback of early packet loss.
- Most of the traffic arriving during a congestion interval will belong to the aggressive flows i.e. TCP. Hence, packets dropped belong to TCP flows causing the congestion.

Many, variants of RED has been proposed like Adaptive RED, Fuzzy RED [21] etc. but we are interested only in RED formula for calculating average queue occupancy.

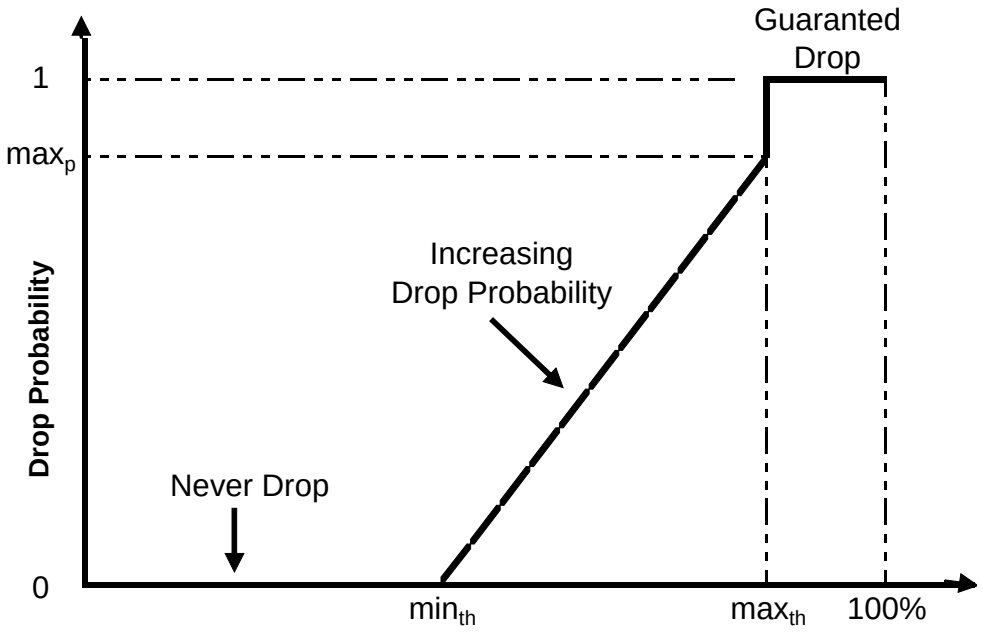


Figure 3.1 Variance of drop probability with queue occupancy

3.2.3 Queue Management for Application layer.

Packet-drop rate and segment re-transmission rate are the performance parameters calculated during QoS analysis stage. These are the instantaneous values. We use these values in RED formula, used for early detection of congestion, to get the average values. RED formula is as follow:

$$Qavg = (1 - Wq) \times Qavg + Qinst \times Wq$$

Here, Qavg is average queue occupancy, Qinst is instantaneous occupancy and Wq is the weight for that queue. Wq affects how closely the average occupancy parameter tracks the queue's instantaneous occupancy, higher values are more aggressive, and lower values are more conservative. The goal is to pick a value that allows RED to ignore short-term transients without inducing packet loss but to react to sustained levels of occupancy before everyone's latency is adversely affected or multiflow synchronization of TCP's congestion avoidance is felt.

In our case packet-drop rate and segment re-transmission rate are the instantaneous values while, packet-drop average and segment re-transmission average are the long-term averages used to detect the congestion.

Let us consider packet-drop rate. The formula would be as follow:

$$Davg = (1 - Wq) \times Davg + Drate \times Wq$$

Here Davg is drop average and Drate is drop rate. Suppose that Davg was 10, and Drate is 15 then effect of different values of Wq can be explained from the Table 3.3.

Davg is the indicator for long-term congestion, while, Drate (i.e. instantaneous value) is the indicator of transient congestion.

We will choose neither very conservative value (causing delay in detection) nor very aggressive value (transient congestion indicator). Rather, we shall focus on mid value initially. Subsequently the solution will adjust the value according to congestion level.

Suppose **Davg = 0.4**
Drate = 0.8

Then effect of different values of **Wq** is given by

Wq	Davg = (1 – Wq) x Davg + Drate x Wq	Observations
0.0	0.4	No effect of instantaneous value.
0.1	0.4	Conservative values
0.2	0.5	
0.3	0.5	
0.4	0.6	
0.5	0.6	Generates mid value of average and instantaneous value.
0.6	0.6	Aggressive values
0.7	0.7	
0.8	0.7	
0.9	0.8	
1.0	0.8	No effect of average value.

Table 3.3 Effect of Wq on average rate calculation

3.2.4 Classification

Queuing is enabled by some type of packet classification or prioritization scheme. Application data is transmitted using TCP, which is a non-real-time protocol that prescribes lost packet retransmission. Real-time data like voice and video data use UDP transmission protocol that does not allow retransmission of lost packets.

One point of view is that only a small number of traffic classes need to be differentiated at any given hop, therefore a usual solution is to assign a handful of bits at a fixed, known location within the packet header for classification. IPv4's ToS octet, IPv6's TC octet, and the DiffServ field all fit into this category. Another point of view is that the classification scheme should cover multiple fields of the IP packet header, scheme is known as multi-field (MF) classification. It includes source and destination addresses, protocol identification and source and destination port numbers.

For an application layer solution which is usually flow oriented multi-field (MF) classification is best. Also for an end-user solution there is no need of source address because it is unique. Also, there is no need of destination port because we are focusing on outgoing traffic only; hence, source port uniquely identifies the outgoing flows.

Within this context, destination address, protocol id and the source port uniquely identifies the flows. This classification enormously decreases the number of identifiable flows and also simplifies the further processing.

CHAPTER 4

Fuzzy Logic with Adaptive Sampling

As the complexity of some system increases, it becomes more difficult and eventually impossible to make a precise statement about its behavior, eventually arriving at a point of complexity where the fuzzy logic method born in humans is the only way to get at the problem. In our solution it is quite difficult to predict the exact values of certain parameters (like bandwidth in a shared link etc) because of their varying nature. Therefore, we will try fuzzy logic in our solution rather than precise values.

Fuzzy logic [18] makes use of human common sense. The degree of fuzziness of a system analysis rule can vary between being very precise, in which case we would not call it "fuzzy", to being based on an opinion held by a human, which would be "fuzzy." Being fuzzy or not fuzzy, therefore, has to do with the degree of precision of a system analysis rule.

Fuzzy Sets

A fuzzy set is a group of anything that cannot be precisely defined. Consider the fuzzy set of "transmission rate." How high is transmission rate? Where is the dividing line between low transmission rate and high transmission rate? Is a 10 mbps high transmission rate? How about 20 mbps? What about 9.9 mbps? The assessment is in the eyes of the user.

We must have a way to assign some rational value to intuitive assessments of individual elements of a fuzzy set. We do this by assigning assessment of conditions a value from zero to 1.0. For "high is the packet transmission rate" we might rate it at 0.8 or 0.9 or even 1.0, if the rate is near the bandwidth, and we might rate the transmission rate at 0.1 or 0.2 if transmission rate is small. We can see these perceptions are fuzzy, just intuitive assessments, not precisely measured facts. By making fuzzy evaluations, with zero at the bottom of the scale and 1.0 at the top, we have a basis for analysis rules for the fuzzy logic method.

Degree of Membership

The degree of membership is the placement in the transition from 0 to 1 of conditions within a fuzzy set. If a particular building's placement on the scale is a rating of .7 in its position in newness among new buildings, then we say its degree of membership in new buildings is .7.

Fuzzy Variable

Words like large, small, etc. are fuzzy and can have many shades and tints. They are just human opinions, not based on precise measurement in angstroms. These words are fuzzy variables. For example, "transmission rate" is a fuzzy variable.

Linguistic Variable

Linguistic means relating to language, in our case plain language words. Examples of linguistic variables are: somewhat high transmission rate, very high transmission rate, real slow transmission rate, etc.

A fuzzy variable becomes a linguistic variable when we modify it with descriptive words, such as somewhat high, very high, real slow, etc. The main function of linguistic variables is to provide a means of working with the complex systems mentioned above as being too complex to handle by conventional mathematics and engineering formulas.

Fuzzy Algorithm

A procedure, usually a computer program, made up of conditional “if-then” statements relating linguistic variables. Examples: If the drop rate is increasing then decrease the transmission rate appropriately.

4.1 Why Fuzzy Logic?

Here is a list of general observations about fuzzy logic:

- Fuzzy logic is conceptually easy to understand. No complex mathematics.
- Fuzzy logic is flexible; it is easy to layer on more functionality without starting again from scratch.
- Fuzzy logic is tolerant of imprecise data.
- Fuzzy logic can model nonlinear functions of arbitrary complexity. You can create a fuzzy system to match any set of input-output data. This process is made particularly easy by adaptive techniques.
- Fuzzy logic can be blended with conventional control techniques rather than replacing conventional control methods.
- Fuzzy logic is based on natural language.

4.2 Node Community

Let us explain node communities in an example network. It is just for demonstrate of our solution in following algorithm.

Community of nodes means all nodes having established sessions among them. There may be many communities in a network. According to their established sessions, nodes could be the part of one or more communities. For example we consider the example network shown in Figure 4.1.

In this network there are following nodes communities;

- Community ‘A’

  Node A exchanges statistics with nodes B, C and D.

- Community ‘B’

  Node B exchanges statistics with nodes A and C.

- Community ‘C’

  Node C exchanges statistics with nodes A and B.

- Community ‘D’

  Node D exchanges statistics with node A.

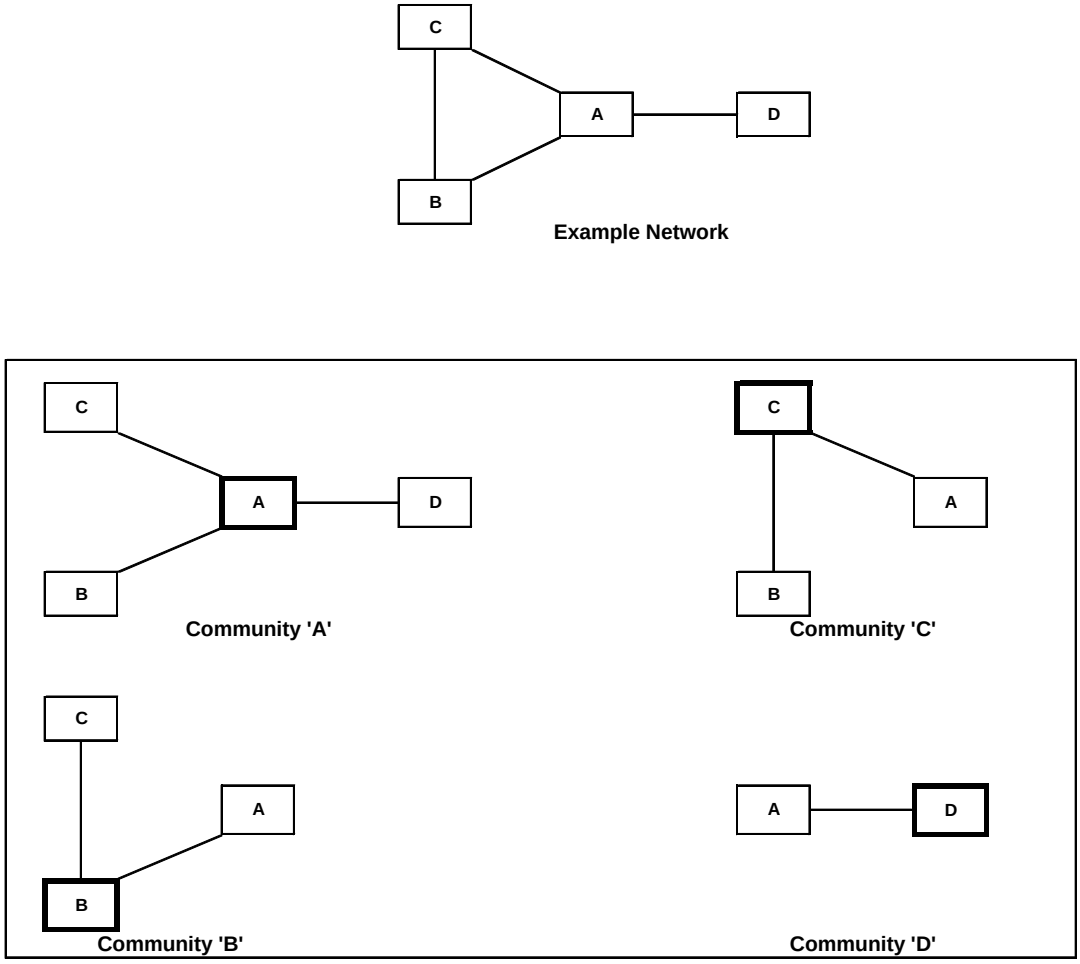


Figure 4.1 Node communities extracted from an example network

4.3 Algorithm

For every node in a community the whole algorithm proceeds as follow;

```
Prioritize services
Loop forever
    {
        Wait for  T  interval of time.
        Exchange statistics with other community nodes.
        Check own statistics.
        If  (there is a performance degrade in own statistics)  then
        {
            Pinpoint low priority services
            Run Fuzzy Logic algorithm to adjust transmission rate
        }
    }
```

Both very small and very large values of "T" are not suitable due to congestion factor and late detection/decision respectively. Value of "T" is adjusted based on previous statistics i.e. adaptive sampling.

4 x Queues can be developed according to the following classification of traffic flows

- TCP flows of Pin-pointed nodes
- UDP flows of Pin-pointed nodes
- TCP flows of normal nodes
- UDP flows of normal nodes

4.4 Sampling Techniques

In network management, accurate measures of network status are needed to aid in planning, troubleshooting, and monitoring. For example, it may be necessary to monitor the bandwidth consumption of several hundred links in a distributed system to pinpoint bottlenecks. If the monitoring is too aggressive, it may create artificial bottlenecks.

With too passive a scheme, the network monitor may miss important events. Network query rates must strike a balance between accurate performance characterization and low bandwidth consumption to avoid changing the behavior of the network while still providing a clear picture of the behavior. This balance is often achieved through sampling.

In network management, status information regarding load, latency, queue occupancy, and other parameters is frequently available in devices such as routers, switches, and network interfaces. Such information is often accessed through the Simple Network Management Protocol (SNMP). In SNMP, a Network Management Station (NMS) queries the network devices, or agents, to periodically assess the status of the network devices or links.

The period of the sampling determines the accuracy of the measured data. Transient activity may not be accurately detected when the sampling interval is large, while small intervals under some traffic loads, simple periodic sampling may be poorly suited to the monitoring task. For example, during periods of idle activity or low network loads, a long sampling interval provides sufficient accuracy at a minimal overhead. However, bursts of high activity require shorter sample intervals to accurately measure network status at the expense of increased sample traffic overhead. To address this issue, adaptive sampling techniques can be employed to dynamically adjust the sampling interval and optimize accuracy and overhead.

Adaptive sampling [16] monitors network behavior by dynamically adjusting the sampling time interval for each parameter monitored. When levels of high activity are detected, a shorter sampling interval is employed to measure the behavior of the network with greater accuracy. When less activity is detected, the sampling interval is lengthened to reduce sampling overhead.

In an effort to balance accuracy with sampling overhead, several sampling disciplines have been applied to network managers.

4.4.1 Conventional Sampling

Traditionally, network management has made use of simple, non-adaptive sampling techniques. Such techniques use a fixed rule to determine when to sample data in each agent. The sampling rule can be deterministic, such as in periodic sampling, or it can involve a random component. Introducing randomness has been shown to improve accuracy in situations where the monitored data is uniform in nature. There are three conventional methods used by network management systems for sampling of agents:

Systematic sampling or periodic sampling

Deterministically samples data at a fixed time interval. Systematic sampling with a period of T seconds.

Random sampling

Employs a random distribution function to determine when each sample should be taken. The distribution may be uniform, exponential, Poisson, etc. Random sampling may take a varying number of samples in a given time interval.

Stratified random sampling

Combines the fixed-time interval used in systematic sampling with random sampling by taking a single sample at a random point during a given time interval.

4.2.2 Adaptive Sampling

Adaptive sampling [16] dynamically adjusts the sampling rate based on the observed sampled data. A key element in adaptive sampling is the prediction of future behavior based on the observed samples. For example, if there is no significant performance degrade in previous samples then the sample interval T is increased, or if there is a significant performance degrade in previous samples then the sample interval T is decreased. In our case adaptive sampling is considered. So with each performance degrade T is reset while on performance increase t is increased exponentially. It will be shown in activity diagram.

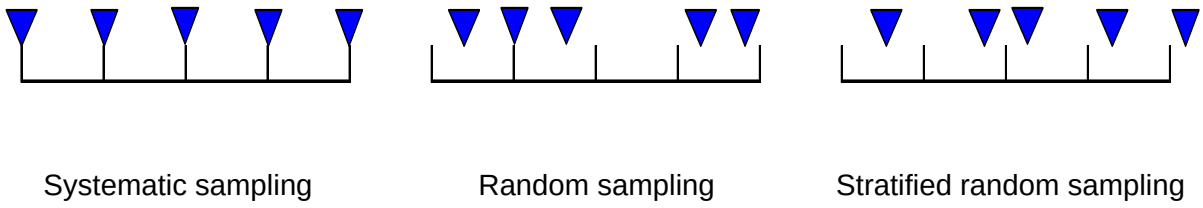


Figure 4.2 Conventional Sampling techniques

4.4 Fuzzy Matrices

After getting statistics from all peers and checking own statistics at every T interval, the next step is to predict the next rate of change for the transmission to minimize the drop rate. For this purpose as a first step, if we put Transmission-Rate (TR) against Drop-Rate (DR) in a 10 x 10 matrices and get function dX = TR – DR, as shown in the Figure 4.3. We can see that there are five areas. Each area defines a generalized rule for transmission rate adjustment. Values 0 to 1.0 used in matrices are just fuzzy values.

- Rule 1

  In this area Drop-Rate (DR) is negligible but Transmission-Rate (TR) is below line rate so we can increase Transmission-Rate (TR). But margin of increase is not precise i.e. fuzzy.

- Rule 2.

  In this area both Transmission-Rate (TR) and Drop-Rate (DR) are non-negligible but TR is higher than the DR. Therefore, TR is reduced by DR to overcome DR. Hence the amount of change in transmission rate i.e. dTR is given by

  dTR = - (DR)

- Rule 3.

  Again in this area both TR and DR are non-Negligible and DR is higher or equal to the TR, therefore TR is reduced to negligible. Hence the amount of change in transmission rate i.e. dTR is given by

  dTR = - (TR)

- Rule 4.

  Some times even though the current Transmission-Rate (TR) is negligible but due to pending packets in the queue the Drop-Rate (DR) is non-negligible. However, No-change (NC) in Transmission-Rate (TR) is possible in this scenario.

- Rule 5.

  In this area Transmission-Rate (TR) is maximum and Drop-Rate (DR) is negligible so No-Change (NC) is required. It is an ideal scenario.

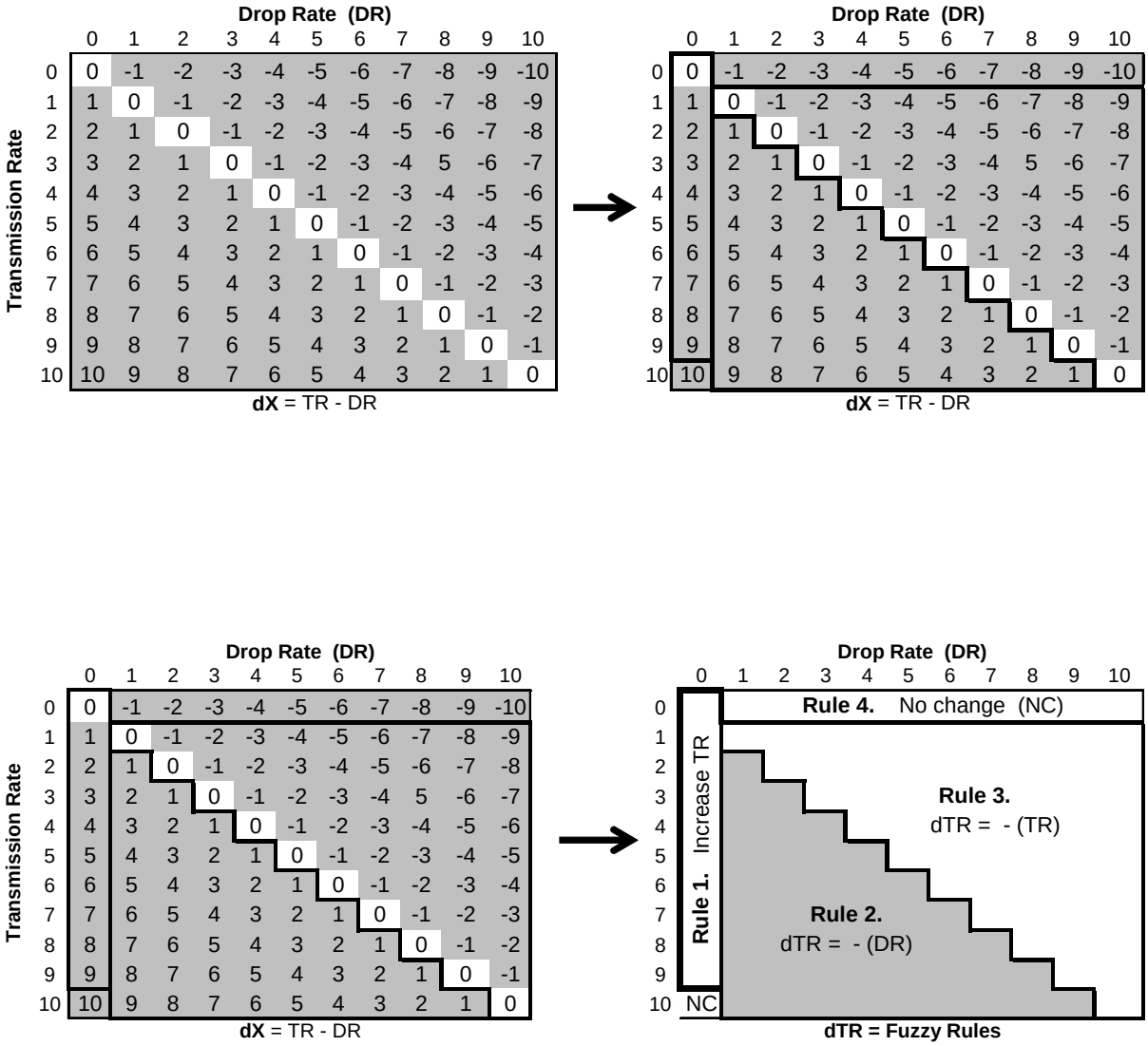


Figure 4.3 Matrices to get generalized rules

Generalized Rules for Transmission Rate Adjustment					
Rule1:	if	(DR == 0	and TR < LR)	then	Increase TR
Rule2 :	if	(DR > 0	and DR < TR)	then	Decrease TR by DR
Rule3:	if	(TR > 0	and DR >=TR)	then	set TR = 0
Rule4:	if	(TR == 0	and DR > 0)	then	No Change
Rule5:	if	(TR == 10	and DR == 0)	then	No Change

Table 4.1 Rules generalized from Matrices

4.5 Fuzzy Membership functions

Unlike traditional digital systems, fuzzy logic differentiates between several levels such as “Large,” “Small” and “Negligible” when making decisions. The numerical ranges assigned to each of the levels are defined in a membership function. Fuzzy algorithm uses one membership function, illustrated in Figure 4.4 to determine a suitable output value given the state of the input values. In this case, there are two membership functions. One for the input and one for the output. For the input membership function, Drop-Rate (DR), the horizontal axis corresponds to the value of the input. A given value of the input is interpreted as being in one or more fuzzy states. Here the Negligible, Slight and Large are the fuzzy sates for input, as shown in Fig 4.4.

The output membership function dTR, as shown in Figure 4.5, falls into one of five fuzzy states: Decrease-Large (DL), Decrease-Small (DS), No-Change (NC), Increase-Small (IS) and Increase-Exponential (IE).

In Figure 4.4 and Figure 4.5, the labels at the top of each peak represent the fuzzy states for the given input or output variable. The y-axis shows the fractional membership in each state for a given value along the x-axis. The x-axis value corresponds to the numerical value of the input or output variable. The membership function for input i.e. DR, describes the amount of packet drop in measured throughput (in terms of SNMP byte-count) between successive samples and ranges from 0 to 1.0 on fuzzy scale.

Finally, the membership function for output i.e. dTR, indicates the required amount of change increase or decrease to be applied to the Transmission-Rate to minimizing Drop-Rate (DR).

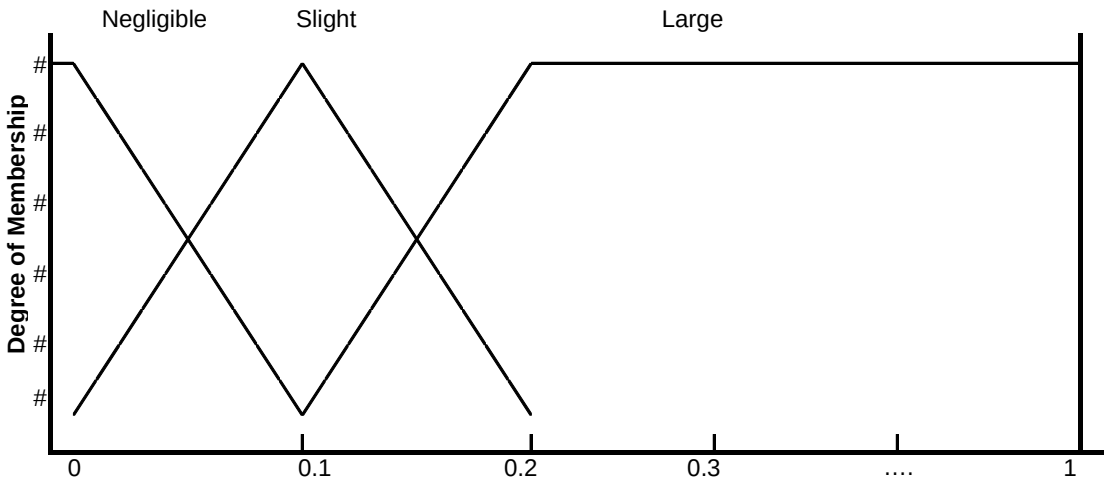


Figure 4.4 Input membership function (Drop Rate (DR))

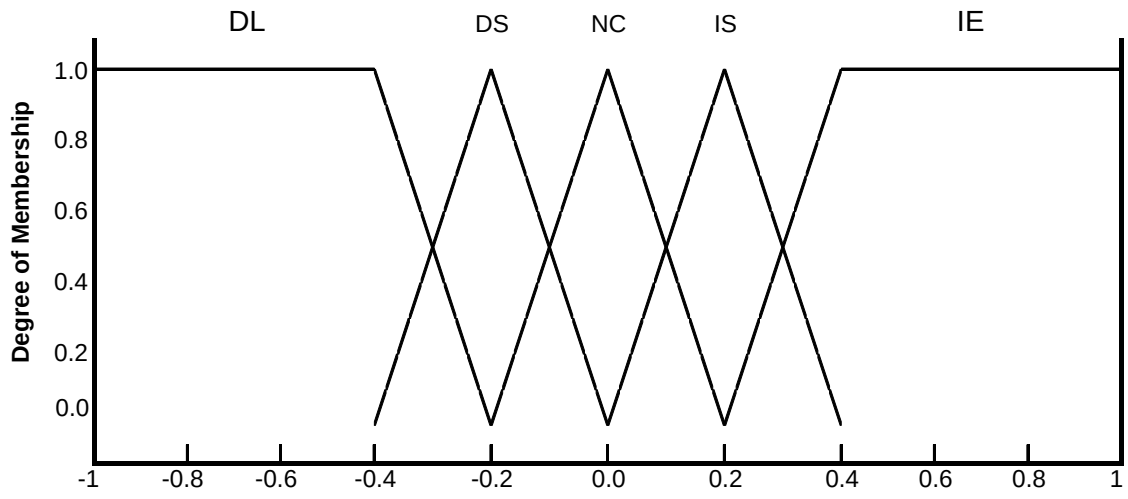


Figure 4.5 Output membership function (Req. amount of change in Trans. Rate (dTR))

4.6 Fuzzy Rules

In addition to the membership functions for the inputs, the Fuzzy Logic algorithm needs a fuzzy set of rules to map the input values to an output response. These fifteen statements shown in Table 4.2 represent a set of fuzzy rules for the proposed fuzzy logic solution.

Each row in this table provides three premises along with a single implication. In the table, the first three columns i.e. current Drop-Rate (DR_N), previous amount of change in transmission rate (dTR_{N-1}) and previous adaptive interval (t_{N-1}) are correlated using the logic operator AND to get outputs i.e. amount of change in transmission rate (dTR_N) and (t_N) respectively. For further elaboration of these fuzzy rules different scenarios are given below and also an activity diagram of the same is shown in Figure 4.6.

Scenario 1: If Drop-Rate (DR) Large

If Drop-Rate (DR_N) is Large then the amount of change for transmission rate (dTR_N) would be Decrease-Large (DL). Because a major transient drop needs quick samples therefore, the adaptive sample interval t_N will be reset to T.

Scenario 2: If Drop-Rate (DR) Slight

If Drop-Rate (DR_N) is Slight then the amount of change for transmission rate (dTR_N) would be Decrease-Slight (DS). Because it is a slight drop therefore, there is no need of change in sample interval. So previous sample interval will be repeated.

Scenario 3: If Drop-Rate (DR) is Negligible

If current Drop-Rate (DR_N) is Negligible then for required amount of change in transmission rate (dTR_N) check the previous amount of change in transmission rate (dTR_{N-1}) and then get the appropriate dTR_N as follow;

- If dTR_{N-1} was IE / IS

  If Increase-Slight (IS) or Increase-Exponential was the previous amount of change, it shows that the increasing Transmission-Rate (TR) caused no Drop-Rate (DR) therefore, Transmission-Rate can be further enhanced to occupy the remaining bandwidth. Hence required amount of change in transmission rate (dTR_N) would be Increase-Exponential (IE). Since exponential increase has raised the chances of packet drop therefore adaptive sample interval t_N is reset to T.

- If dTR_{N-1} was NC

  This scenario shows that the Transmission-Rate (TR) was just about ideal line that's why it was kept unchanged i.e. amount of change was No-Change (NC).

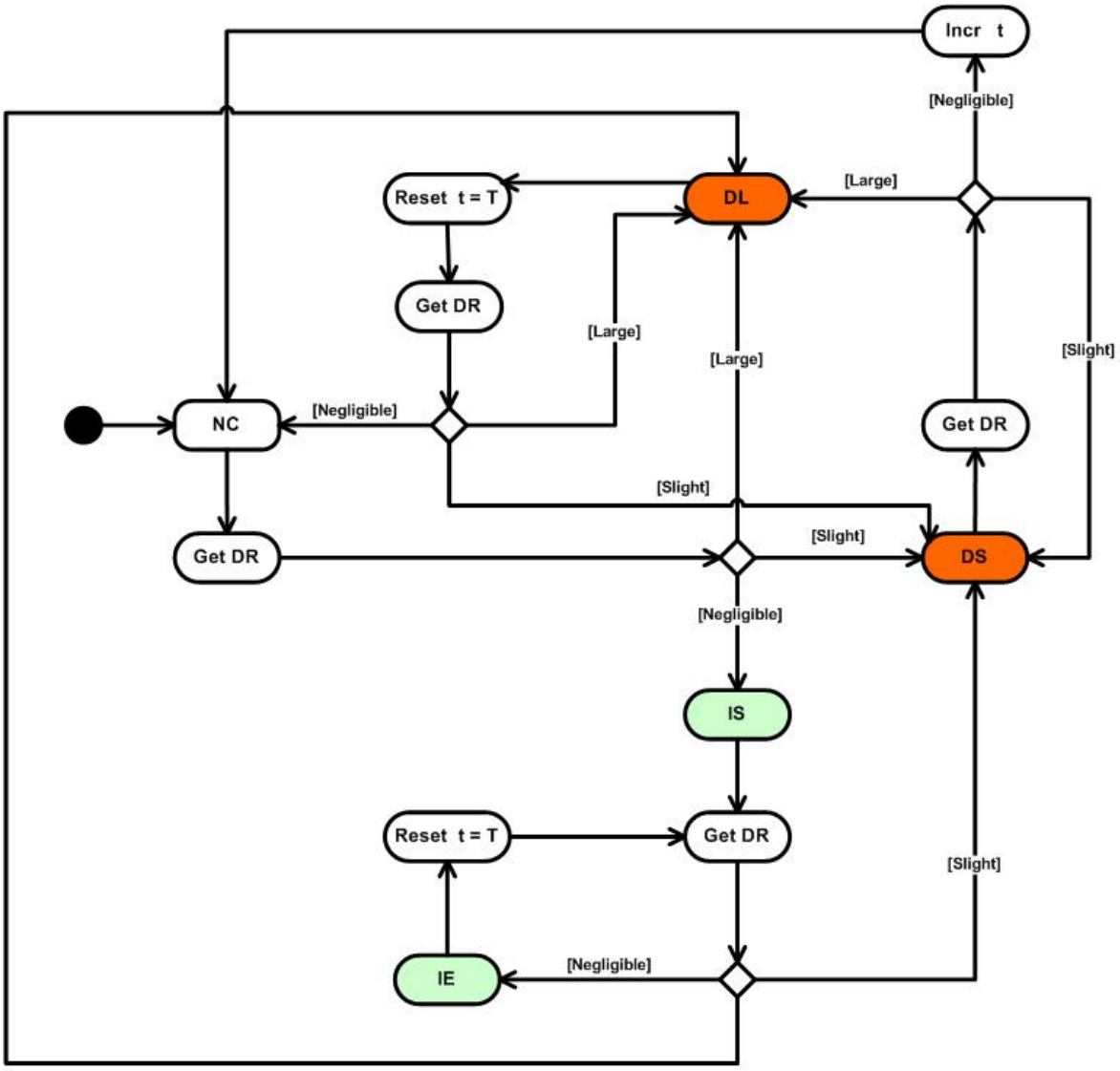
Although it is an ideal situation but the main problem is that in an environment where bandwidth is shared by multiple nodes, bandwidth for different nodes constantly varies. Therefore bandwidth should be constantly probed to check for any extra space recently made. For this purpose if NC state has completed its tenure i.e. adaptive interval t_N, therefore dTR_N would be Increase-Slight (IS). Adaptive interval t_N will remain unchanged i.e. NC.

- If dTR_{N-1} was DS

  If previous amount of change was Decrease-Slight, then it shows that there was a slight decrease in Transmission-Rate (TR) to nullify the Drop-Rate, which resulted in negligible drop rate this time. It shows a slight convergence of transmission rate at ideal curve line. Therefore, we try to farther extend this transmission rate without any change i.e. dTR_N would be No-Change (NC). Since we are entering in peak rate region so adaptive interval t_N which is applicable during NC state is also incremented exponentially so that the fluctuation can be eliminated for longer duration.

- If dTR_{N-1} was DL

  If previous amount of change was Decrease-Large, then it indicates that there was a major packet drop. But now DR is negligible which indicates convergence of curve, therefore, dTR should be NC. But since the solution converged from a major drop therefore, adaptive interval should be reset to T.



LEGEND:

DR	:	Drop Rate
IE	:	Increase Exponential
IS	:	Increase Slight
NC	:	No Change
DS	:	Decrease Slight
DL	:	Decrease Large
t	:	Adaptive Sample Interval for NC state
T	:	Sample Interval for IE, IS, DS, DL states

Figure 4.6 Activity diagram of Transmission Rate adjustment using fuzzy logic.

Rule	DR_N	dTR_{N-1}	dTR_N	t_N
1	Negligible	IE	IE	T
2	Negligible	IS	IE	T
3	Negligible	NC	IS	NC
4	Negligible	DS	NC	Incr
5	Negligible	DL	NC	T
6	Slight	IE	DS	NC
7	Slight	IS	DS	NC
8	Slight	NC	DS	NC
9	Slight	DS	DS	N C
10	Slight	DL	DS	NC
11	Large	IE	DL	T
12	Large	IS	DL	T
13	Large	NC	DL	T
14	Large	DS	DL	T
15	Large	DL	DL	T

Table 4.2 Fuzzy Rules base

Legends:

DR	Drop Rate
dTR	Amount of change in Transmission Rate
IE	Increase Exponential
IS	Increase Slight
NC	No Change
DS	Decrease Slight
DL	Decrease Large
T	Sample Interval for IE, IS, DS, DL states
t	Adaptive Sample Interval for NC state
Incr	Increment value of t

CHAPTER 5

Concept Implementation

5.1 Platform and Environment

As for as network programming is concerned both Java and C++ have their merits and demerits. We developed our demo solution for Microsoft Windows and selected C++.Net environment for development for the reasons that C++.Net simplifies socket programming (Winsock interface), even some of toughest chores, such as asynchronous socket programming, multithreading, and multicasting are made easier and quicker than ever.

5.1.1 Winsock 2 API.

Windows Sockets 2 Application Programming Interface is used for communication among peers. Ws2_32.lib is used to communicate with Winsock 2 API. Winsock 2 has the following key features.

- Access to protocols other than TCP/IP

  Windows Sockets programming has been TCP/IP-centric, this is no longer the case and WinSock 2 is an interface and not a protocol, hence it provides a protocol-independent transport interface that is fully capable of supporting emerging networking capabilities including real-time multimedia communications.

- Quality of Service

  WinSock 2 establishes conventions for applications to negotiate required service levels for parameters such as bandwidth and latency. Other QOS-related enhancements include socket grouping and prioritization, and mechanisms for network-specific QoS extensions.

- Protocol-independent multicast and multipoint

  Applications can discover what type of multipoint or multicast capabilities a transport provides and use these facilities in a generic manner.

- Other frequently requested extensions

  Shared sockets, conditional acceptance, exchange of user data at connection setup/teardown time, protocol-specific extension mechanisms.

5.1.2 Multithreading.

Because nodes will communicate with multiple peers for sending and receiving statistics therefore solution is multithreaded.

5.2 Fuzzy Blue method

Fuzzy Blue is an Active Queue Management [20]. It is used for performance comparison with our fuzzy algorithm therefore just required information is given here. The input linguistic variables in fuzzy blue algorithm are packet loss and queue length. The output linguistic variable is the drop probability (i.e., pm). The term set of linguistic variables packet loss and normalized queue length are defined as bellow:

T (packet loss) = { small, med (medium), big}

T (normalized queue length) = {low, mid (middle), high }

The output term set of fuzzy logic controller is also defined as bellow:

T (Pm) = {zero, low, moderate, high}

Fuzzy Rule base is given in Table 5.1.

Linguistic rules			
if packet loss is small	and	normalized queue length is low	then Drop Prabability is zero
if packet loss is small	and	normalized queue length is midle	then Drop Prabability is zero
if packet loss is small	and	normalized queue length is high	then Drop Prabability is zero
if packet loss is medium	and	normalized queue length is low	then Drop Prabability is zero
if packet loss is medium	and	normalized queue length is midle	then Drop Prabability is zero
if packet loss is medium	and	normalized queue length is high	then Drop Prabability is moderate
if packet loss is big	and	normalized queue length is low	then Drop Prabability is zero
if packet loss is big	and	normalized queue length is midle	then Drop Prabability is low
if packet loss is big	and	normalized queue length is high	then Drop Prabability is high

Table 5.1 Linguistics Rules of Fuzzy Blue method

5.3 Source Code (Important Methods)

Just a few important methods are given here from demo source code for understanding. All names are self descriptive therefore just a little additional description is given.

- Socket Initialization Method

  private: static int InitSocket()

  Initializes WinSock2 API. Returns status as either success or failure.

- Method for Initializing Statistics Table

  private: static DataTable* InitStatsTable()

  Creates and initializes statistics table which is updated after every sample interval. Method returns statistics table.

- Main method which sends / receives statistics.

  private: static void SendRecv()

  It is the main method used for sending / receiving to / from community nodes.

- Adaptive Method for Transmission Rate adjustment

  private: static signed int AdaptiveMethod (Byte dropped, Byte PktRt)

  Implements adaptive technique for transmission rate adjustment. Takes Current instantaneous drop and packet rate as input and returns next amount of change for TR.

- Fuzzy Blue method for Transmission Rate adjustment

  private: static FUZZY_STS FuzzyBlue (int PL, int TR, FUZZY_STS fuzzy)

  Implements Fuzzy Blue technique for transmission rate adjustment. Takes Current Packet Loss, TR and Fuzzy States as input and returns next amount of change for TR.

- Fuzzy Method for Transmission Rate adjustment

  private: static FUZZY_STS FuzzyMethod (int DR, int TR, FUZZY_STS fuzzy)

  Implements our Fuzzy technique for transmission rate adjustment. Takes Current instantaneous drop rate, TR and Fuzzy States as input and returns next amount of change for TR.

- Method for Increasing Transmission Rate

  private: static FUZZY_STS Increase_TR (int TR, FUZZY_STS fuzzy)

- Method for Decreasing Transmission Rate

  private: static FUZZY_STS Decrease_TR (int DR, FUZZY_STS fuzzy)

- Method to get Fuzzy states from numeric values

  private: static signed int get_Fuzzy (signed int Val)

  Gets precise values as input and returns corresponding fuzzy states.

5.4 Performance Analysis

For analysis of results, different scenarios as given in previous chapter (4.6 Fuzzy Rules) are checked in a demo solution. Our Fuzzy method results are analyzed against results of adaptive method and the Fuzzy Blue method as shown in the Table 5.2 and the graphs in Figure 5.1, Figure 5.2, Figure 5.3 and Figure 5.4. In all cases initially fixed Transmission Rate is used until some packet drop occurs, at that time appropriate method is applied.

In Table 5.2 both Drop Rate (DR) and Transmission Rate (TR) of Adaptive, Fuzzy Blue and our Fuzzy method are given.

Figure 5.1 shows comparison of DR in Adaptive and new Fuzzy method. Initially both curves indicate no drop rate because of availability of queue for absorption of extra data. Then as the queue becomes full a major drop occurs starting from 5th sample. Fuzzy method quickly adjusted drop rate (8^{th} sample) compared to the adaptive method (10^{th} sample). Also in successive samples fuzzy method maintains negligible drop rate on the other hand adaptive method shows fluctuating curve i.e. small transient drops till the end.

Figure 5.2 shows comparison of DR in Fuzzy Blue and new Fuzzy method. Initially both curves indicate no drop rate because of availability of queue for absorption of extra data. Then as the queue becomes full a major drop occurs starting from 5^{th} sample. At that point both methods adjust drop rate. Fuzzy Blue in this regard works better than Adaptive method. But in successive samples Fuzzy Blue is also fluctuating like Adaptive method, here new Fuzzy method excellently maintains long smooth intervals instead of fluctuating curves. These intervals will gradually increase due to adaptive sampling technique.

Figure 5.3 repeats comparison of Adaptive method against new Fuzzy method with one addition i.e. Transmission Rate (TR) is also included along with Drop Rate (DR). Comparison shows that both DR and TR have the same trend. It is because we considered packet loss due to congestion and not due to physical media.

Figure 5.4 repeats comparison of Fuzzy Blue method against new Fuzzy method with addition of Transmission Rate (TR). Remaining comparison is same as Figure 5.3.

Method	Samples																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Adapt TR	0	30	30	30	30	27	24	21	18	17	18	17	18	17	18	17	18	17	18	17	18	17	18	17	20
Adapt DR	0	0	0	0	10	10	8	5	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	0	0
Fuzzy TR	0	30	30	30	30	20	17	16	16	18	17	16	16	16	16	18	17	16	16	16	16	16	16	16	16
Fuzzy DR	0	0	0	0	10	3	1	0	0	2	1	0	0	0	0	2	1	0	0	0	0	0	0	0	0
Fuzzy Blue TR	0	30	30	30	30	20	17	18	17	18	17	18	17	18	17	18	17	18	17	18	17	18	17	18	23
Fuzzy Blue DR	0	0	0	0	10	3	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	0	0

Table 5.2 Transmission and Drop Rate comparison.

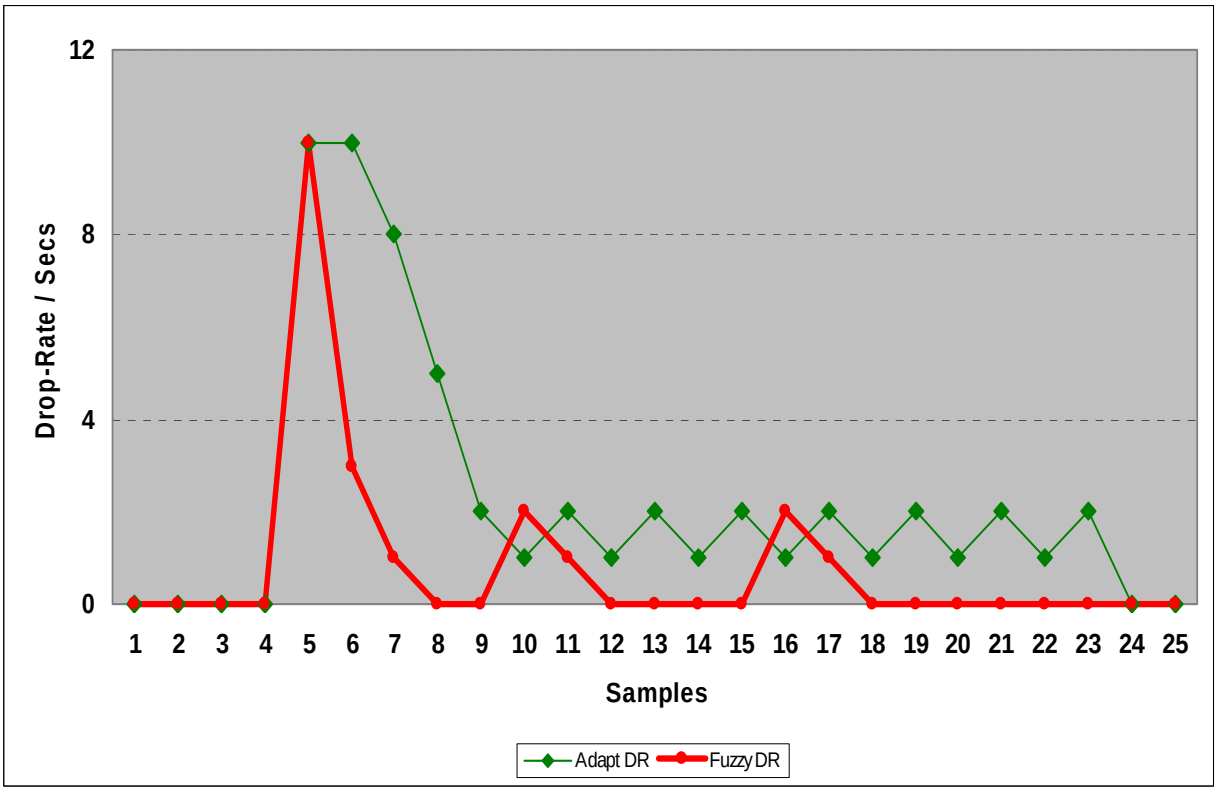


Figure 5.1 Graph of Drop Rate in Fuzzy method against Adaptive method

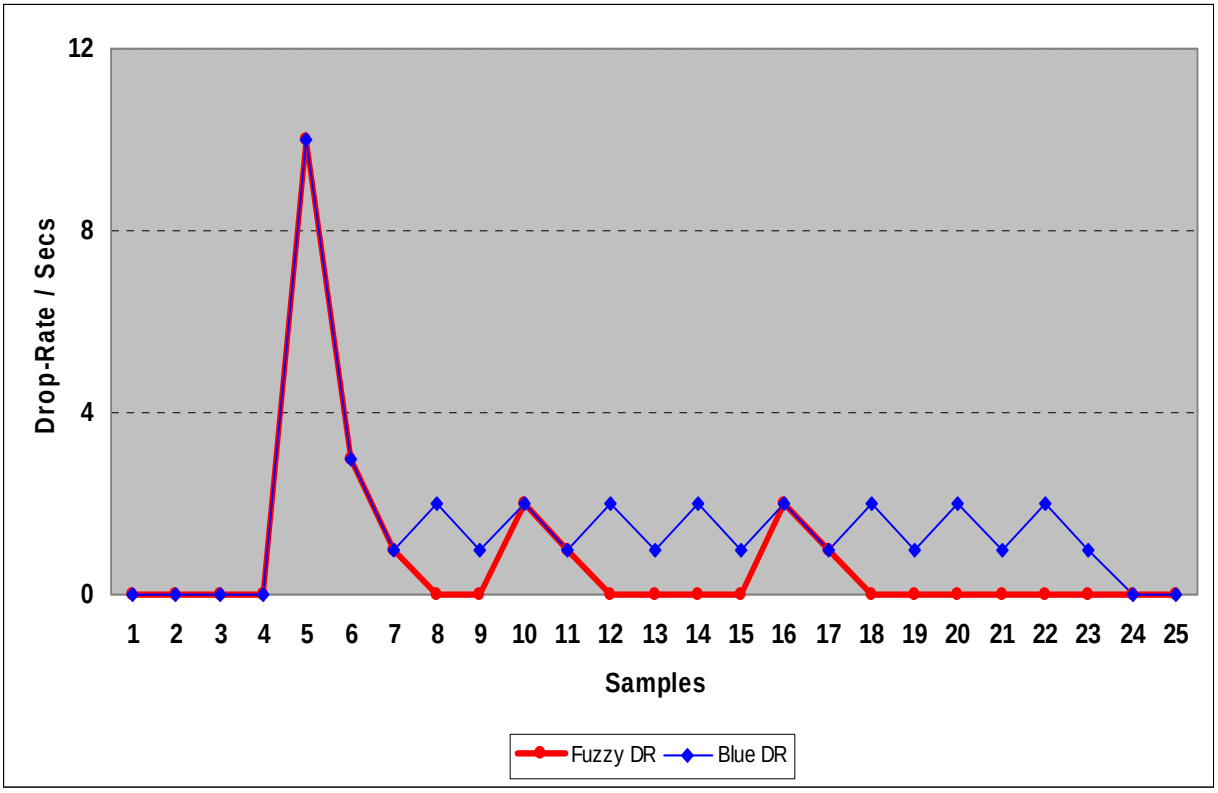


Figure 5.2 Graph of Drop Rate in Fuzzy method against Fuzzy Blue method

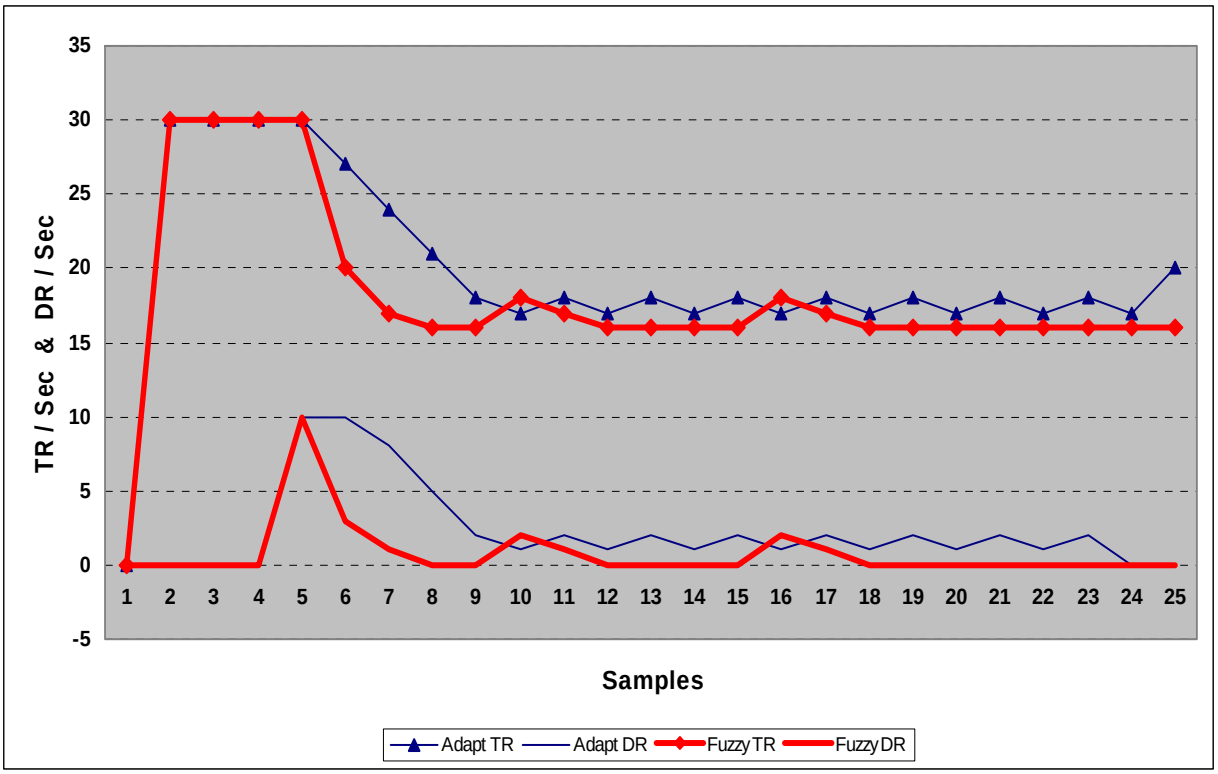


Figure 5.3 Graph of Fuzzy Transmission Rate and Drop Rate against Adaptive method

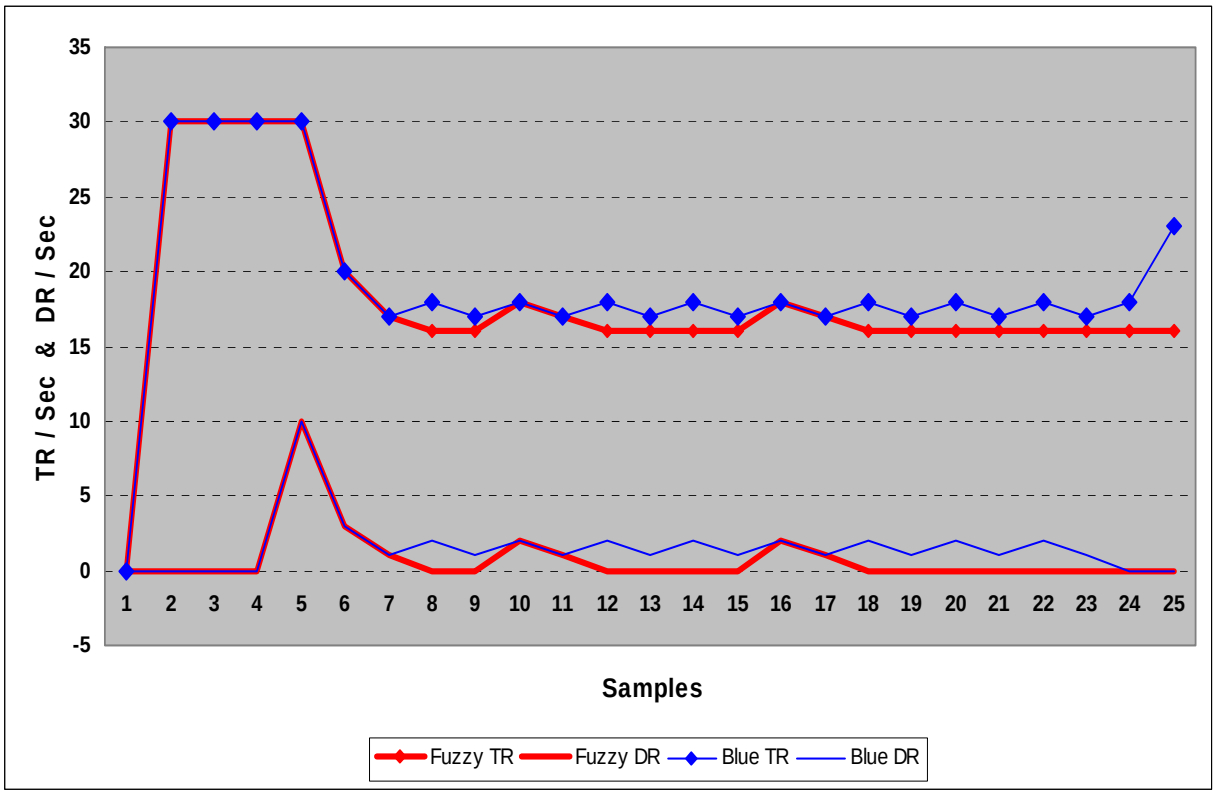


Figure 5.4 Graph of Fuzzy Transmission Rate & Drop Rate against Fuzzy Blue method

CHAPTER 6

Conclusion

In this thesis we presented a strategy to improve application layer IP QoS using fuzzy logic. Our aim was to develop a performance diagnostic solution that continuously observes the performance of the network. Whenever the performance degrades to some threshold level, system automatically tries to resolve the performance bottleneck by using QoS principles and mechanisms. For performance measurement we used passive measurement using polling as data extraction technique. Polling is selected because it is easier data sampling and extraction of specified detailed information on request rather than collected regardless.

First major problem in this regard was that how we can measure the available capacity because due to sharing of common bandwidth it is constantly varying?

To address this problems solution uses fuzzy logic, because due to the complexity of solution it is difficult to establish precise values. Hence, new solution generalized some fuzzy rules and fuzzy rule base to attain peak performance.

Second major problem was that how can we reduce the number of measurement samples needed to construct the network history?

To address this problem, adaptive sampling is used, so when certain pre-determined conditions are met, such as an increase in drop rate, corresponding actions are taken such as a decrease in sampling interval and vice versa.

For performance analysis comparison we checked our performance analysis against performance analysis of adaptive method and also against fuzzy blue method.

In different scenarios initially fixed Transmission Rate is used until some packet drop occurs, at that time appropriate method is applied.

Initially all solution indicate no drop rate because of availability of queue for absorption of extra data. On major drops (due to buffer-overflow), fuzzy methods quickly adjusted compared to the adaptive method. In successive samples fuzzy method tries to maintain negligible drop rate by minimizing bandwidth probing compared to adaptive method which shows fluctuation due to frequent network probing.

New fuzzy solution maintains long smooth intervals instead of fluctuating curves compared to fuzzy blue and adaptive technique. Adaptive increase in sample intervals during smooth phase makes new solution more efficient both against transient and long-term congestion as compared to Adaptive and Fuzzy Blue method.

Performance comparison shows that new solution is quite efficient as compared with the fuzzy blue and the adaptive method.

REFERENCES

[1]. http://www.cs.wustl.edu/ - QoS standards.

[2]. ftp://ftp.rfc-editor.org/in-notes/rfc791.txt - Internet Protocol

[3]. ftp://ftp.rfc-editor.org/in-notes/rfc1122.txt - Requirements for Internet Hosts - Communication layer.

[4]. ftp://ftp.rfc-editor.org/in-notes/rfc1123.txt - Requirements for Internet Hosts - Application and Support.

[5]. ftp://ftp.rfc-editor.org/in-notes/rfc1142.txt - OSI IS-IS Intra-domain Routing Protocol

[6]. ftp://ftp.rfc-editor.org/in-notes/rfc1583.txt - OSPF-2

[7]. ftp://ftp.rfc-editor.org/in-notes/rfc1349.txt - Type of Service in the Internet.

[8]. http://www.cs.wustl.edu/~jain/cis788-99/ftp/qos_products - QoS in data networks

[9]. ftp://ftp.rfc-editor.org/in-notes/rfc1633.txt - Integrated services in Internet Architecture.

[10]. Grenville, Armitage, “Quality of Service in IP networks“, First Edition, April 2000.

[11]. http://www.comp.lancs.ac.uk/ : Network Traffic measurement for next Gen. Internet.

[12]. A. Ore-baugh, "Ethereal Packet Sniffing", Syngress Publishing, 2004.

[13]. Luca Deri, Finsiel S.p.A. Stefano Suin, “Effective Traffic Measurement Using ntop”, University of Pisa.

[14]. BOB MELANDER, “Probing-Based Approaches to Bandwidth Measurements and Network Path Emulation”, UPPSALA University.

[15]. http://www.aciri.org/Floyd/Papers - RED (Random Early Detection).

[16]. Edwin A. Hernandez and Matthew C. Chidester etc, “Adaptive Sampling for Network Management”, Dept. of Electrical and Computer Engineering, University of Florida.

[17]. Andreas Pitsillides and Ahmet Sekercioglu, “Congestion Control”.

[18]. http://www.fuzzy-logic.com/ : “Fuzzy Logic for just plain folks" by Thomas Sowell

[19]. Andreas Pitsillides1, Ahmet Sekercioglu, “Fuzzy logic based Congestion control”, University of Cyprus, Nicosia, Cyprus,

REFERENCES (Continued)

[20]. Mohammad Hossein Yaghmaee, Halleh AminToosi, “A Fuzzy Based Active Queue Management Algorithm”, Ferdowsi University of Mashhad, Iran.

[21]. Andreas Pitsillides, L. Rossides and A. Sekercioglu, “Fuzzy RED, a Congestion Control for TCP/IP Diff-Serv”.

[22]. Saman Taghavi Zargar† and Mohammad Hossein Yaghmaee, “Fuzzy Green, a modified TCP equation-based active queue management using fuzzy logic approach”.

[23]. Andreas Pitsillides and C. Chrysostomou, “Fuzzy logic congestion control in TCP/IP best-effort networks”.

[24]. W. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”, Network Working Group RFC-2001.