FUZZY LOGIC BASED AQM CONGESTION CONTROL IN TCP/IP NETWORKS

Chrysostomos Chrysostomou

Advisor

Prof. Andreas Pitsillides

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Cyprus

Recommended for Acceptance

by the Department of Computer Science

September 2006

2006

Acknowledgements

I would like to express my gratitude to my advisor Prof. Andreas Pitsillides for his precious guidance and continuous support throughout my Ph.D. study. His trust in my work and his advices always helped me to shape my research work towards something more meaningful.

I would also like to thank our research collaborators Prof. Marios Polycarpou of University of Cyprus and Dr. Ahmet Sekercioglu of Monash University of Australia for their fruitful comments and reviews on my research work.

Further, I would like to thank my colleagues in the Networks Group, and especially to George Hadjipollas and Yiannos Mylonas for their support and encouragement. Special thanks go to my former colleague Loukas Rossides who had introduced me to the world of fuzzy logic.

Finally, I wish to express my gratitude to my parents, Pantelis and Maria, who under difficult financial conditions have always been encouraging me to achieve high education. My parents, together with my fiancée Anna, were always very supportive, and kept me going on the right path to complete my Ph.D. study, and so I deeply dedicate this thesis to them.

Abstract

Network management and control is a complex problem that requires robust, intelligent, control methodologies to obtain satisfactory performance. Active Queue Management (AQM) mechanisms have been introduced for router support to assist the TCP congestion control to perform satisfactorily in all circumstances. However, certain, well-known limitations, identified in the AQM literature, motivates the need to investigate alternative control techniques to control the time-varying dynamics, high variability, and nonlinearities of TCP/IP systems.

We present a new fuzzy logic based AQM control methodology to provide effective congestion control in TCP/IP networks. We adopt fuzzy logic due to its reported strength in controlling nonlinear systems using linguistic information. This methodology is developed to offer a simple and generic process, and thus it is adequately adopted in both best-effort and differentiated services (Diff-Serv) TCP/IP environments, providing acceptable quality of service (QoS).

The proposed fuzzy logic approach for congestion control allows the use of linguistic knowledge to capture the dynamics of nonlinear probability marking functions, and uses multiple inputs to capture the (dynamic) state of the network more accurately. The potential of Fuzzy logic control methodology to incorporate human knowledge into such a control strategy is demonstrated, and the capability to qualitatively capture the attributes of a control system based on observable phenomena is a main feature of fuzzy logic control and has been shown in the simulative evaluation.

We demonstrate through simulation evaluation, under widely differing operating conditions, that the fuzzy control methodology (for both best-effort and Diff-Serv

environments) satisfies all the design requirements. It provides quality of service and high link utilization, with minimal losses, and bounded queue fluctuations and delays. In the framework of Diff-Serv, an adequate differentiation is further offered among different drop precedence's traffic. The proposed fuzzy control methodology is shown to exhibit many desirable properties, like robustness and fast system response, with capabilities of adapting to highly variability and uncertainty in network, in contrast with other schemes compared with. It offers significant improvements in controlling congestion in TCP/IP networks, without the need for retuning.

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

ACK	Acknowledgment
ACP	Adaptive Congestion Protocol
AF PHB	Assured Forwarding Per-hop Behavior
AIMD	Additive-Increase, Multiplicative-Decrease
AQM	Active Queue Management
A-RED	Adaptive Random Early Detection
ATM	Asynchronous Transfer Mode
AVQ	Adaptive Virtual Queue
CE	Congestion Experienced
CI	Computational Intelligence
CU	Currently Unused
CWR	Congestion Window Reduced
DCCP	Datagram Congestion Control Protocol
Diff-Serv	Differentiated Services
DSfield	Differentiated Services field
DT	Drop Tail
ECE	ECN-Echo
ECN	Explicit Congestion Notification
ECT	ECN-Capable Transport
EF PHB	Expedited Forwarding Per-hop Behavior
EWMA	Exponentially Weighted Moving Average
FEM	Fuzzy Explicit Marking
FIE	Fuzzy Inference Engine
FIO	Fuzzy Explicit Marking In/Out
FLC	Fuzzy Logic Control

FLCM	Fuzzy Logic-based Control Methodology
FTP	File Transfer Protocol
Goodput	Useful Throughput
ICCRG	Internet Congestion Control Research Group
IETF	Internet Engineering Task Force
Int-Serv	Integrated Services
IP	Internet Protocol
Jitter	Variation in delay
Latency	End-to-End Delay
Mbps	Mega bits per second
MISO	Multiple Input Single Output
NS-2	Network Simulator - 2
PHB	Per-Hop Behavior
PI	Proportional Integral
QoS	Quality of Service
RED	Random Early Detection
REM	Random Exponential Marking
RFC	Request For Comments
RIO	Random Early Detection In/Out
RTT	Round Trip Time
SACK	Selective Acknowledgement
SLA	Service Level Agreement
TCP	Transmission Control Protocol
TFRC	TCP Friendly Rate Control
TOS	Type Of Service
TQL	Target Queue Length
UDP	User Datagram Protocol
VoIP	Voice-over-IP
WG	Working Group
XCP	Explicit Control Protocol

Chapter 1

Introduction

1.1 Problem Statement

Network management and control is a complex problem, which is becoming even more difficult with the increased demand to use the Internet for time/delay-sensitive applications with differing Quality of Service (QoS) requirements (e.g. Voice over IP, video streaming, Peer-to-Peer, interactive games). The existing TCP congestion avoidance/control mechanisms, while necessary and powerful, are not sufficient to provide good service in all circumstances. The insufficiencies of the implicit end-to-end feedback adopted by the TCP paradigm necessitate the design and utilization of new effective congestion control algorithms, to supplement the standard TCP based congestion control, since the replacement of the current TCP congestion control algorithm does not appear to be realistic at this point in time. Further, given the need for providing adequate QoS new network architectures have been proposed, such as the Differentiated Services (Diff-Serv) architecture (Blake et al., 1998) to deliver aggregated QoS in IP networks[*].

Basically, there is a limit to how much control can be accomplished from the edges of the network of such an end-to-end implicit feedback based congestion control. Some additional mechanisms are needed particularly in the routers to complement the endpoint congestion control methods. Thus the need for router control has recently led to the concept of active queue management (AQM). We will focus our

[*] Other architectures were also proposed, such as Integrated Services, but were not widely adopted, mainly due to identified scalability problems.

attention to AQM based schemes that aim to provide high network utilization with low loss and delay.

1.2 Motivation

The problem of network congestion control remains a critical issue and a high priority; despite the many years of research efforts and the large number of different control schemes proposed, there are still no universally acceptable congestion control solutions. Current solutions of existing AQM mechanisms, introduced to assist the TCP congestion control, are ineffective to meet the diverse needs of today's Internet, due to the dynamic, time-varying nature of TCP/IP networks. It is widely accepted that they have serious limitations and drawbacks, including:

- The linearity of the control functions of existing AQM mechanisms that cannot capture effectively the nonlinearities of the TCP network.
- The dependency of AQM control parameters on dynamic network parameters, like the number of flows and the round trip propagation delays.
- The linearization of the existing models to allow analysis and design of AQM-based controllers, often making stability bounds overly conservative, and performance sluggish when dynamic changes occur.
- The accuracy of the existing TCP/AQM models, as they ignore the slow start phase of TCP and/or timeout events that are prominent conditions in today's Internet with the existence of short-lived TCP/Web flows.

Thus, despite the classical control system techniques used from various researchers, these still do not perform sufficiently to control the dynamics, and the nonlinearities of the TCP/IP networks. Given the need to capture such important attributes of the controlled system, the design of robust, intelligent control methodologies is required.

Hence, given the need for such control methodology – to capture the dynamics, the highly bursty network traffic, and the nonlinearities of the TCP/IP system, under

widely differing operating conditions, we investigate the usefulness of fuzzy logic control to meet such objectives. Fuzzy Logic Control can be considered as suitable candidate for AQM-based control mechanism due to its reported strength in controlling nonlinear systems using linguistic information.

The capability to qualitatively capture the attributes of a control system based on observable phenomena is a main feature of fuzzy logic control and has been demonstrated in various places in the research literature as well as in commercial products. The main idea is that if the fuzzy logic control is designed with a good (intuitive) understanding of the system to be controlled, the limitations due to the complexity system's parameters introduced on a mathematical model can be avoided. A common approach in the networking literature is to either ignore such complex parameters in the mathematical model (e.g., Misra, Gong, & Towesly, 2000), or to simplify the model (e.g., Hollot, Misra, Towsley, & Gong, 2001) to such an extent (in order to obtain some stability results), which render the designed controllers and their derived stability bounds overly conservative.

Therefore, the application of fuzzy control techniques to the problem of congestion control in TCP/IP networks is worthy of investigation, due to the difficulties in obtaining a precise enough mathematical model (amicable to analysis) using conventional analytical methods, while some intuitive understanding of congestion control is available.

1.3 Contributions of the Thesis

The complex, but challenging, concept of TCP/AQM congestion control, in both best-effort and Diff-Serv environments, is the key issue of our research study. In this thesis we investigate the suitability of fuzzy logic control to capture efficiently the dynamics, the high burstiness of the network traffic, and the nonlinearities of the TCP/IP system and obtain satisfactory performance.

In particular, we make a significant contribution in formulating an effective, robust, and generic AQM control methodology, using fuzzy logic based control,

easily adopted in TCP/IP best-effort and Diff-Serv networks to solve the problem of congestion control.

The proposed fuzzy control methodology offers significant improvements in controlling congestion in TCP/IP networks, under differing operating conditions, without the need for retuning, and thus provides acceptable QoS, with high link utilization, minimal losses, and bounded queue fluctuations and delays. Furthermore, the proposed methodology provides adequate and effective differentiation among different drop precedence's traffic in the presence of congestion in a Diff-Serv environment.

1.4 Thesis Structure

The structure of the thesis is as follows: In Chapter 2 we discuss congestion control, and give a short overview of the main features and functionalities of the Transmission Control Protocol (TCP), its evolution, as well as other TCP-like/friendly congestion control mechanisms. We further discuss the use of router support to congestion control, either by having single-bit feedback (like Explicit Congestion Notification – ECN), or multi-bit feedback (like eXplicit Control Protocol – XCP, and Adaptive Congestion Protocol – ACP).

In Chapter 3 we study a number of existing AQM-based mechanisms for congestion control in TCP/IP best-effort networks and identify common limitations. In Chapter 4 we give a brief overview of the Diff-Serv architecture, and we further discuss the properties of the Diff-Serv congestion control.

Chapter 5 reviews some of the properties of Fuzzy Logic Control, and the design steps of a Fuzzy Logic Controller. We further give a brief overview of the application of fuzzy logic in networks, in recent years.

In Chapter 6 we present our proposed non-linear Fuzzy Logic based AQM Control methodology applied in TCP/IP best-effort networks, namely Fuzzy Explicit Marking (FEM). We discuss the design steps, the complexity of the implementation, and the sensitivity of the results to the control parameters.

Chapter 7 presents detailed simulation results and discusses the performance of FEM as compared to other representative AQM schemes. It is demonstrated through extensive simulations over a wide range of network conditions that the fuzzy logic based AQM control methodology better handles the nonlinearities of the TCP network, and thus provides an effective control to congestion.

In Chapter 8 we present the extended non-linear Fuzzy Logic based Control methodology applied in TCP/IP Diff-Serv networks, namely Fuzzy Explicit Marking In/Out (FIO), and in Chapter 9 we present simulation results and discuss the performance of FIO.

Finally, in Chapter 10 we present the main conclusions of this thesis, and we introduce some thoughts for future work.

Appendix A of the thesis gives in detail the linguistic rules chosen in the proposed Fuzzy Logic based Control Methodology. Appendix B and C contain simulation results concerning the performance of FEM and FIO, and their counterparts, respectively.

Chapter 2

Congestion Control in Internet Protocol Networks

2.1 Introduction

Congestion control is a critical issue in Internet Protocol (IP) networks. Many research proposals can be found in the literature to provide means of avoiding and/or controlling the congestion. The fundamental principles of congestion, and different approaches to avoid or/and control congestion are widely discussed. In this Chapter, the main functionalities of the standard Transmission Control Protocol (TCP) congestion control mechanisms are explained. In addition, many variants of TCP which have been proposed to meet Internet's today needs are briefly described. As there is strong trend to progressively move the controls inside the network, closer to where it can be sensed, we discuss the use of router support to congestion control, either by having explicit single-bit feedback, or multi-bit feedback. Due to its current practical significance, in this thesis we focus on explicit single-bit feedback.

2.2 Defining Congestion

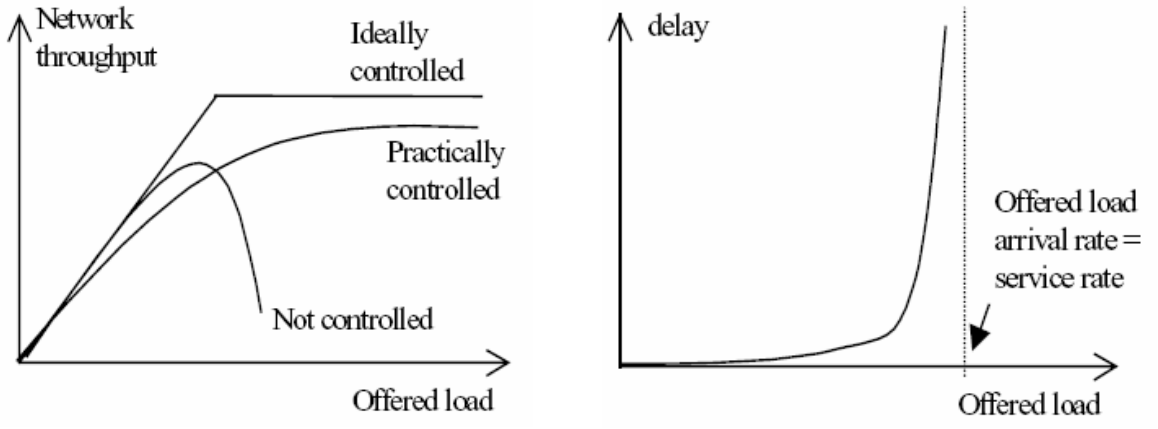
Congestion is a complex process to define. Despite the many years of research efforts in congestion control, currently there is no agreed definition. One may refer to

the ongoing discussion between the active members of the networking community as to give the right definition for congestion (ICCRG, 2006).

Two perspectives on network congestion are the user perspective and the network perspective.

Keshav (1991) states that “Network congestion is a state of degraded performance from the perspective of a particular user. A network is said to be congested from the perspective of a user if that user’s utility has decreased due to an increase in network load”. The user experiences long delays in the delivery of data, 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, which 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 – much faster than when the network is not heavily loaded.

Yang and Reddy (1995) give a network-centric definition of congestion, 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 (Pitsillides & Sekercioglu, 2000). Networks need to serve all users requests, which may be unpredictable and bursty in their behaviour. 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 optimal control of networks of queues is a well-known, much studied, and notoriously difficult problem, even for the simplest of cases (e.g., Hassan & Sirisena, 2001; Andrews & Slivkins, 2006).



(a) (b)

Figure 2.1 Network throughput and delay vs offered load

Figure 2.1a shows the throughput-load relationship in a packet-switching network (Schwartz, 1988). This plot shows the effect of excessive loading on the network throughput for three cases: no control, ideally controlled, and practically controlled. In the case of ideal control, the throughput increases linearly until saturation of resources, where it flattens off and remain constant, irrespective of the increase of loading beyond the capacity of the system. Obviously, this type of control is impossible in practice. Hence for the practically controlled case, we observe some loss of throughput, as there is some communication overhead associated with the controls, possible some inaccuracy of feedback state information as well as some time delay in its delivery. Finally, for the uncontrolled case, congestion collapse may occur whereby as the network is increasingly overloaded the network throughput collapses, i.e. very little useful network traffic is carried – due to retransmissions or deadlock situations.

Figure 2.1b shows the corresponding delay-load relationship. The delay (response time) plot follows a similar pattern as the throughput plot. At first, the delay rises slowly with the offered load even for fast increments of the throughput. Then after the knee point is reached (i.e., the queues start building), the delay curve jumps significantly while the throughput stays flat. Finally, the delay grows indefinitely when the network becomes congested (i.e., the queues start overflowing).

2.3 Congestion Control Principles

Chiu and Jain (1989) classify most congestion control approaches into two categories: approaches for congestion avoidance and approaches for congestion recovery. Congestion avoidance mechanisms allow a network to operate in the optimal region of low delay and high throughput, thus, preventing the network from becoming congested. In contrast, the congestion recovery mechanism allows the network to recover from the congested state of high delay and losses, and low throughput. Even if a network adopts a strategy of congestion avoidance, congestion recovery schemes would still be required to retain throughput in the case of abrupt changes in a network that may cause congestion.

Both types of approaches are basically resource management problems. They can be formulated as system control problems, in which the system senses its state and feeds this back to its users who adjust their control (Chiu & Jain, 1989). This simple classification only provides a very general picture of common properties between separating groups of approaches.

A number of taxonomies of congestion control were/could be considered. A detailed taxonomy for congestion control algorithms is proposed by Yang and Reddy (1995), which focuses on the decision-making process of individual congestion control algorithms. The main categories introduced by the Yang and Reddy (1995) taxonomy are:

- Open loop: These are the mechanisms in which the control decisions of algorithms do not depend on any sort of feedback information from the congested spots in the network, that is, they do not monitor the state of the network dynamically.
- Closed loop: These are the mechanisms that make their control decisions based on some sort of feedback information to the sources. With the provision of feedback, these mechanisms are able to monitor the network performance dynamically. The feedback involved may be implicit or explicit. In the explicit feedback scheme, feedbacks have to be sent explicitly as separate packets (or can

be *piggybacked*) (e.g. Ramakrishnan, Floyd, & Black, 2001). If there is no necessity of sending the feedback explicitly, the scheme is said to be an implicit feedback scheme. Some examples of such implicit feedbacks are time delays of acknowledgment or timeouts, and packet loss (e.g. Jacobson, 1988; Stevens, 1997 - an implicit binary feedback scheme).

- The feedback can be further categorised into binary and "full" feedback. A single bit in the packet header is used as a binary feedback mechanism (e.g. Ramakrishnan, Floyd, & Black, 2001 – an explicit binary feedback scheme). "Full" feedback incorporates use of more than one bit in the packet header that are used to send a whole (i.e. "full") information about the status of the network, like the exact sending rate, the round-trip time, etc (e.g. Katabi, Handley, & Rohrs, 2002 – an explicit multi-bit ("full") feedback scheme).

A congestion control system should be preventive, if possible. Otherwise, it should react quickly and minimise the spread of congestion and its duration. A good engineering practice will be to design the system in such a way as to avoid congestion. But taken to the extreme (i.e. to guarantee zero losses and zero queuing delay), this would not be economical. For example, assuring zero waiting at a buffer implies increasing the service rate at its limit to infinity. A good compromise would be to allow for some deterioration of performance, but never allow it to become intolerable (congested). The challenge is to keep the intolerance at limits acceptable to the users. Note the *fuzziness* present in defining when congestion is actually experienced.

The difficulty of the congestion control problems has caused a lot of debate as to what are appropriate control techniques for the control of congestion, and depending on one's point of view, many different schools of thought were followed, with many published ideas and control techniques.

2.4 Internet Congestion Control

The Internet Protocol (IP) architecture is based on a connectionless end-to-end packet service. Transmission Control Protocol (TCP) is an end-to-end transport protocol that provides reliable, in-order service. Congestion control is implemented via a reactive, closed-loop, dynamic window control scheme (Jacobson, 1988). This window-based scheme operates in the hosts to cause TCP connections to "back off" during congestion. That is, TCP flows are responsive to congestion signals (i.e. dropped packets indicated by a timeout or a triple duplicate acknowledgment) from the network. It is primarily these TCP congestion avoidance algorithms that prevent the congestion collapse of today's Internet.

A fundamental aspect of TCP is that it obeys a "conservation of packets" principle, where a new segment is not sent into the network until an old segment has left. TCP implements this strategy via a self-clocking mechanism (acknowledgements received by the sender are used to trigger the transmission of new segments). This self-clocking property is the key to TCP's congestion control strategy. Other elements of TCP's congestion control include the congestion avoidance algorithm, the congestion recovery algorithm (i.e. slow-start), and the fast retransmit/recovery algorithms (Stevens, 1994, 1997).

A TCP sender additively increases its rate when it perceives that the end-path is congestion-free, and multiplicatively decreases its rate when it detects (via a loss event) that the path is congested. Thus, in such situations, TCP congestion control deploys the so called *additive-increase, multiplicative-decrease* (AIMD) algorithm. The linear increase phase of TCP's congestion control protocol is known as *congestion avoidance*. The value of congestion window repeatedly goes through cycles during which it increases linearly and then suddenly drops to half its current value (when a loss event occurs, and particularly a triple duplicate acknowledgment), giving rise to a saw-toothed pattern in long-lived TCP connections (Shenker, Zhang, & Clark, 1990; Lakshman & Madhow, 1997).

During the initial phase of TCP's congestion control, which is called *slow-start*, the TCP sender begins by transmitting at a slow rate but increases its sending rate

exponentially, until a slow-start threshold is reached, where the congestion avoidance phase begins. In the case of a loss event, the AIMD saw-toothed pattern begins.

The TCP congestion control reacts differently to a loss event that is detected via a timeout event, than it does to a loss event detected via receipt of a triple duplicate acknowledgment (ACK). After a triple duplicate ACK, the congestion window is cut in half and then increases linearly (i.e. AIMD). However, after a timeout event, a TCP sender enters a slow-start phase, where the congestion window is set to 1, and then it grows exponentially, until it reaches one half of the value it had before the timeout event. At that point, the TCP enters congestion avoidance phase.

2.4.1 TCP Feedback Signalling Scheme

The TCP feedback signalling scheme in the current Internet is binary and implicit (i.e., network congestion is detected at the sources by loss events). Due to the binary nature of the feedback, and consequently the AIMD saw-toothed pattern, the system does not generally converge to a single steady state. The system reaches an "equilibrium", in which it oscillates around the optimal state (Chiu & Jain, 1989) (see Figure 2.2). The time taken to reach the equilibrium (that determines the responsiveness of the control), and the size of the oscillations (that determines the

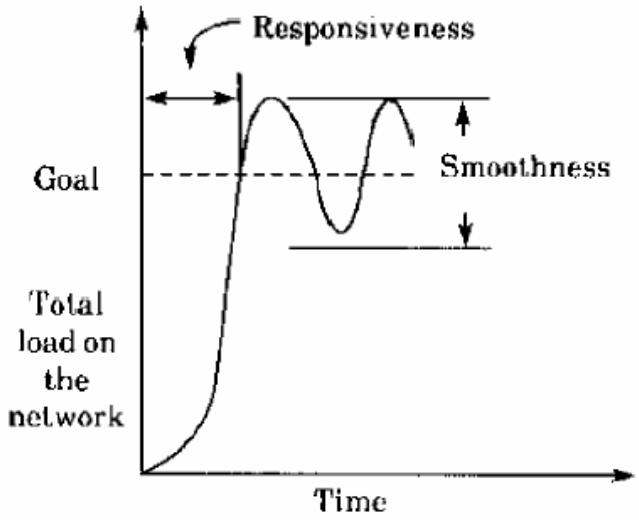


Figure 2.2 Responsiveness and smoothness of the control (Chiu & Jain, 1989)

smoothness of the control) jointly determine the convergence. Ideally, we would like the time as well as oscillations to be small. Therefore, the controls with smaller time and smaller amplitude of oscillations are called more responsive and smoother, respectively.

2.4.2 TCP Evolution

The congestion control mechanisms continue to be enhanced as TCP/IP evolves to meet new and more demanding requirements.

The early version of TCP, known as *TCP Tahoe*, enters the slow-start phase irrespective of the type of loss event. The newer version of TCP, *TCP Reno*, cancels the slow-start phase after a triple duplicate ACK, which is called *fast recovery*, and resends the lost packet, without waiting for a timeout event, that is a *fast retransmit* occurs. Figure 2.3 illustrates the evolution of TCP's congestion window for both Reno and Tahoe (Kurose & Ross, 2005).

The *TCP NewReno* (Allman, Paxson, & Stevens, 1999; Floyd, Henderson, & Gurtov, 2004) improves the Reno implementation regarding the fast recovery mechanism. The aim of TCP NewReno is to prevent a TCP sender from reducing its congestion window multiple times in case several packets are dropped from a single window of data (a problem Reno has). The NewReno remains in fast recovery until all of the data outstanding by the time the fast recovery was initiated have been acknowledged. NewReno can retransmit one lost packet per round trip time (RTT), until all the lost packets from a particular window of data have been retransmitted. Thus NewReno avoids multiple reductions in the congestion window, or unnecessary retransmit timeout with slow start invocation.

Another proposed modification to TCP, the *TCP Selective Acknowledgement* (SACK) (Mathis, Mahdavi, Floyd, & Romanow, 1996) allows a TCP receiver to acknowledge out-of-order packets selectively rather than just cumulatively acknowledging the last correctly received, in-order packet. Thus TCP Sack may recover multiple lost packets in a window of data in just one single RTT.

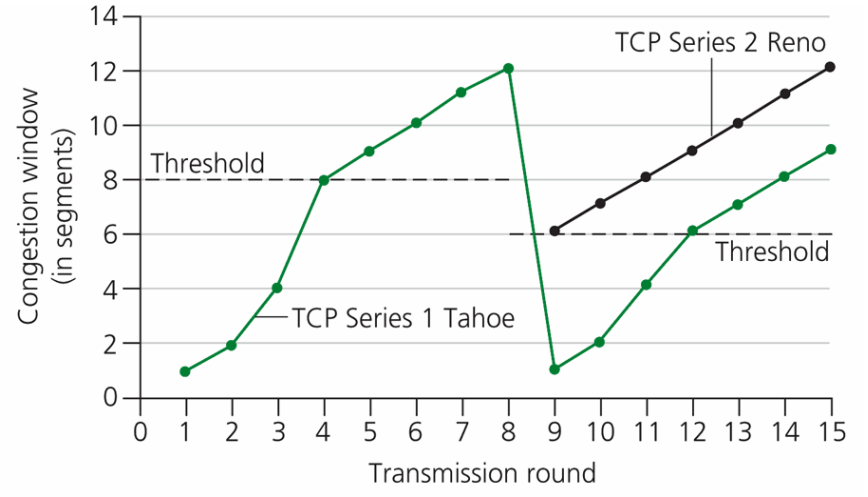


Figure 2.3 Evolution of TCP's congestion window
(Kurose & Ross, 2005)

2.4.3 TCP-like Variants

The behaviour of TCP/IP congestion control still remains a matter of continuous research interest in the TCP/IP world (highlighted by the frequent Internet Engineering Task Force – IETF – Request for Comments – RFCs, and many published papers in various journals and conferences, proposing fixes or new solutions). What follows, are some examples of the vast research proposals:

TCP Vegas (Brakmo & Peterson, 1995; Hengartner, Bolliger, & Gross, 2000) observes the indication of congestion differently than the previous TCP variants mentioned above. In particular, TCP Vegas does not detect network congestion by loss events; rather it monitors the changes in the RTTs associated to the packets that it has sent previously. If the observed RTTs increase, TCP Vegas infers incipient network congestion and so it reduces the congestion window by one; otherwise, it increases the congestion window by one. There is also a RTT range, in which no changes on the congestion window happens (i.e. the sending rate is matching the network capacity).

TCP-Westwood+ (Mascolo et al., 2001; Grieco & Mascolo, 2003) is a new version of the TCP protocol, aimed at improving its performance under random or sporadic

losses. It introduces “faster” recovery to avoid over-shrinking the congestion window after a loss event (due to a timeout or a triple duplicate ACK), by taking into account the end-to-end estimation of the bandwidth available to TCP. The available bandwidth is estimated at the TCP source by measuring and performing low-pass filtering of the returning rate of ACKs. The estimated bandwidth is then used to adaptively decrease the congestion window and the slow-start threshold after a loss event.

Recently, there is an ongoing research towards enhancing the TCP congestion control mechanisms in order for TCP to fully exploit the network capacity of fast, long-distance networks (i.e. high-speed networks operating at 622 Mbit/s, 2.5 Gbit/s, or 10GBit/s, which have a high bandwidth-delay product). Research proposals in changing the way TCP adapts its congestion window include *HighSpeed TCP* (Floyd, 2003), *Scalable TCP* (Kelly, 2003), *FAST TCP* (Jin, Wei, & Low, 2004), *High TCP* (Leith & Shorten, 2004), as well as the *TCP Westwood+* (Altman et al., 2006) mentioned above.

2.4.4 TCP-friendly Congestion Control

With the widespread deployment of applications with stringent QoS requirements, like streaming multimedia and IP telephony, it is becoming increasingly important to ensure that this kind of applications can coexist with each other and with current TCP based applications (e.g., file transferring, Web-like, etc). Many current multimedia applications use the *User Datagram Protocol* (UDP), which does not offer any kind of congestion control, and do not change their sending rate regardless of the congestion state of the network. This can cause severe degradation in the performance of TCP-based applications, and may lead to network collapse. Thus recent proposals aim to incorporate some form of congestion control in multimedia applications, and make them “*TCP-friendly*”. By saying TCP-friendliness, it is meant that “the steady state send rate of the protocol will be roughly equal to that of a TCP connection experiencing similar network conditions” (Padhye, 2000).

To accomplish this goal, equation-based rate-controlled congestion control has been proposed in the literature (Handley, Floyd, Padhye, & Widmer, 2003). By that, it is meant that for traffic competing in the best-effort Internet with TCP, the appropriate response function is a model of TCP that characterises the steady-state sending rate of TCP as a function of the round-trip time and steady-state loss event rate. Thus using a TCP like response function makes the protocol "TCP-friendly". Further, the non-TCP flows should be able to satisfy requirements of the demanding application, but should not be unfair to the competing TCP flows. So, such non-TCP flows should use a control equation to govern their sending rate friendly from the TCP flows perspective. The equation relies on the loss event rate that is generally computed at the receiver and sent to the sender. The accuracy of the equation is fundamental for this alternative congestion control approach (see ongoing discussions at the *Datagram Congestion Control Protocol Working Group* (DCCP-WG, 2001)). However, given the complexity of such a modelling, a complete model appears still to be too far from being conceived.

Recently, a new unreliable transport protocol incorporating end-to-end congestion control (Floyd, Handley, & Kohler, 2006; Kohler, Handley, & Floyd, 2006) has been proposed, namely *Datagram Congestion Control Protocol* (DCCP). DCCP implements a congestion-controlled, unreliable flow of datagrams for use by applications, such as streaming media or on-line games. DCCP provides a choice of modular congestion control mechanisms. Two mechanisms are currently specified: TCP-like Congestion Control (Floyd & Kohler, 2006), and TCP-Friendly Rate Control (TFRC) (Floyd, Kohler, & Padhye, 2006).

2.4.5 Network-assisted Congestion Control

With network-assisted congestion control, routers provide explicit feedback to the sender regarding the congestion state in the network. This feedback may be as simple as a single bit indicating congestion at a link, or more complex as a multi-bit feedback giving to the source "full" information about the network state (e.g., the exact sending rate). Congestion information is typically conveyed from each router at

the path from the sender to the receiver, by marking/updating a field in a packet's header, to indicate congestion, and then fed back from the receiver to the sender as a form of notification.

It has become clear (Braden et al., 1998) that the existing TCP congestion avoidance/control mechanisms and its variants, while necessary and powerful, are not sufficient to provide good service in all circumstances. Basically, there is a limit to how much control can be accomplished from the edges of the network. Some additional mechanisms are needed in the routers to complement the endpoint congestion avoidance/control methods, as suggested by various researchers (Floyd & Jacobson, 1993; Braden et al., 1998; Ramakrishnan, Floyd, & Black, 2001). Note that the need for router control was realised early; e.g. see Jacobson (1988), where for future work the router side is advocated as necessary. A clear trend is observed: to progressively move the controls inside the network, closer to where it can be sensed.

By using network-assisted congestion control, TCP does not need to await a loss event – due to buffer overflow – to detect congestion and slow down properly. Instead, it is informed by the intermediate nodes (routers) when incipient congestion starts, and reacts accordingly.

2.4.5.1 Binary Feedback

The simplest method of assisting the TCP from the network point of view is to provide a binary feedback to the source about the network state. The use of *Explicit Congestion Notification* (ECN) was proposed (Ramakrishnan, Floyd, & Black, 2001) in order to provide TCP an alternative to packet drops as a mechanism for detecting incipient congestion in the network. The ECN proposal works together with the addition of *active queue management* (AQM) to the Internet infrastructure, where routers detect congestion before the queue overflows (see discussion of AQM in detail in Chapter 3).

Table 2.1 The ECN field in IP

ECN FIELD		
0	0	Not-ECT
0	1	ECT (1)
1	0	ECT (0)
1	1	CE

The ECN scheme requires both end-to-end and network support. An ECN-enabled router can *mark* a packet by setting a bit in the packet's header, if the transport protocol is capable of reacting to ECN. Specifically, the ECN proposal requires specific flags in both IP and TCP headers. Two bits are used in each header for proper signalling among the sender, routers, and the receiver.

In the IP header, the two bits (ECN field) results in four ECN codepoints (see Table 2.1). The *ECN-Capable Transport* (ECT) codepoints '10' and '01' are set by the data sender to indicate that the end-points of the transport protocol are ECN-capable. Routers treat both codepoints as equivalent; senders are free to use either of the two to indicate ECT. The not-ECT codepoint '00' indicates a packet that is not using ECN. The ECN codepoint '11' is set by a router to indicate congestion to the end nodes (i.e. *marks* the packet); this is called the *Congestion Experienced* (CE) codepoint. Upon the receipt by an ECN-capable transport of a single CE packet, the congestion control algorithms followed at the end nodes must be essentially the same as the congestion control response to a single dropped packet.

In the TCP header, two new flags are introduced. The *ECN-Echo* (ECE) flag is used by the receiver to inform the sender that a CE packet has been received. This is done in the ACK packet sent. Similarly, the sender uses the *Congestion Window Reduced* (CWR) flag to announce to the receiver that its congestion window has been reduced, as a consequence of the reception of the ECE ACK.

The use of ECN for notification of congestion to the end-nodes generally prevents unnecessary packet drops, and thus is appealing to be used in the Internet.

2.4.5.2 Multi-bit Feedback

As discussed in Section 2.4.1, if feedback control is binary, then the system does not generally converge to a single steady state; it rather reaches an “equilibrium”, in which it oscillates around the optimal state. Further, the congestion information fed back to the end-node, obviously, is not as rich as it could be using multiple-bit feedback. If multiple bits were used to convey the feedback, more precise information can be made available to the source. Of course, issues like creating/or not more overhead in the network is a subject that still needs investigation.

A recent proposal for enhancement of the current TCP, using multiple-bit feedback is an optional *Quick-Start* mechanism proposed by Floyd, Allman, Jain, and Sarolahti (2006). The aim of this mechanism is, in cooperation with routers, to determine an allowed sending rate at the start and at times in the middle of a data transfer (e.g., after an idle period), to allow connections to use higher sending rates – when there is significant unused capacity along the path and the sender and all the routers along the path approve the Quick-Start request. Quick-Start is being proposed as a mechanism that could be used in controlled environments. Each router along the path could approve the requested rate, reduce the requested rate, or indicate that the Quick-Start request is not approved (based on the ability of all routers along the path to determine if their respective output links are significantly underutilized or not). The Quick-Start option includes a request for a sending rate in bits per second, and a Quick-Start time-to-live to be decremented by every router along the path that understands the option and approves the request. The Quick-Start request includes, among other information, a four-bit rate request field, in which the request range is from 80Kbit/s to 1.3Gbit/s, using an encoding function.

Further, proposed solutions for TCP congestion avoidance problems in high speed networks, using multiple-bit feedback are the *eXplicit Control Protocol* (XCP), and the A*daptive Congestion Protocol* (ACP).

The XCP (Katabi, Handley, & Rohrs, 2002) generalizes the ECN proposal. Instead of the one-bit congestion notification used by the ECN, XCP-enabled routers inform senders of the degree of congestion at the bottleneck. Each XCP packet carries a

congestion header, which is used to communicate a flow's state to routers and feedback from the routers to the receivers. Decisions on how to update the sending rate of the sources are taken by the sources themselves, based on congestion signals that they receive from the network. These congestion signals are generated by the routers using a static control law.

An alternative proposal to XCP is ACP (Lestas, 2005), in which the decisions on how to update the sending rate of the sources are taken by the routers using a dynamic control law. That means that intelligence is transferred from the end nodes to the network. ACP aims to achieve max-min fairness and high network utilization at equilibrium; to clear the queues at equilibrium and to ensure small number of drops; to exhibit smooth responses and fast convergences.

2.5 Conclusions

Congestion control is still a critical issue, despite literally hundreds of proposed possible, and probably very good, solutions addressing the diverse needs of today's Internet. However, the difficulty in changing any deployed protocol, together with the "robust" behaviour (Internet as is, practically works!) of the ubiquitous Jacobson TCP congestion control mechanisms in most cases, are resisting the employment of new algorithms. In order to supplement the standard TCP based congestion control, network-assisted congestion control is introduced, that is, routers provide explicit feedback to the source regarding the congestion state in the network. This type of feedback can be either binary or multi-bit. Many research solutions towards this aim have been proposed.

The feedback signalling scheme in the current Internet is binary and implicit. With the standardization of ECN (Ramakrishnan, Floyd, & Black, 2001), the Internet feedback mechanism can become explicit and binary. This is the base of our research study; introduce intelligent, control methodology, as part of router support for congestion control, to supplement the standard TCP, to obtain satisfactory performance.

Chapter 3

Active Queue Management in TCP/IP Networks

3.1 Introduction

Network management and control is a complex problem, which is becoming even more difficult with the increased demand to use the Internet for time-sensitive applications with differing QoS requirements. The deficiencies of the implicit end-to-end feedback adopted by the TCP paradigm has led to the introduction of active queue management (AQM) mechanisms, as router support to the TCP congestion control. In this chapter we motivate the need for AQM and then the modelling and control approach followed by a number of well-known AQM schemes are discussed, and their limitations to meet the diverse needs of today's Internet are identified.

3.2 The Need for Active Queue Management

The TCP congestion avoidance/congestion control mechanisms have been very successful, as the Internet has evolved from a small-scale research network to today's interconnected millions of networks. However, the increased demand to use the Internet for time/delay-sensitive applications with differing QoS requirements, questions the efficiency and the feasibility of such an end-to-end implicit feedback

based congestion control. It has become clear (Braden et al., 1998) that the existing TCP congestion avoidance/control mechanisms and its variants, while necessary and powerful, are not sufficient to provide good service in all circumstances. Basically, there is a limit to how much control can be accomplished from the edges of the network. Some additional mechanisms are needed in the routers to complement the endpoint congestion avoidance/control methods, as suggested by various researchers (Floyd & Jacobson, 1993; Braden et al., 1998; Ramakrishnan, Floyd, & Black, 2001). Note that the need for router control was realised early; e.g. see Jacobson (1988), where for future work the router side is advocated as necessary. A clear trend is observed: to progressively move the controls inside the network, closer to where it can be sensed. Thus AQM mechanisms were proposed, which aim to provide high link utilization with low loss rate and queuing delay, while responding quickly to load changes.

Braden et al. (1998) distinguish between two types of router algorithms that are related to congestion control, that is queue management, and scheduling: "To a rough approximation queue management algorithms manage the length of packet queues by dropping packets (or sending feedback signal to regulate the rate) when necessary or appropriate, while scheduling algorithms determine which packet to send next and are used primarily to manage the allocation of bandwidth among flows" (Braden et al., 1998).

Queues are used to smooth spikes in incoming packet rates, and to allow the router sufficient time for packet transmission. When the incoming packet rate is higher than the router's outgoing packet rate, the queue size will increase, and eventually will exceed the available buffer space. In the existing TCP implementation, when the buffer is full, the packets that are just arriving are dropped; this dropping policy is known as *Tail-Drop* or *Drop Tail* (DT). DT is the most widespread dropping policy, due to its simplicity.

Although DT is simple to implement, and has been used for many years, it was shown to interact badly with TCP congestion control mechanisms and to lead to poor performance. In particular, studies have shown that DT can cause *global*

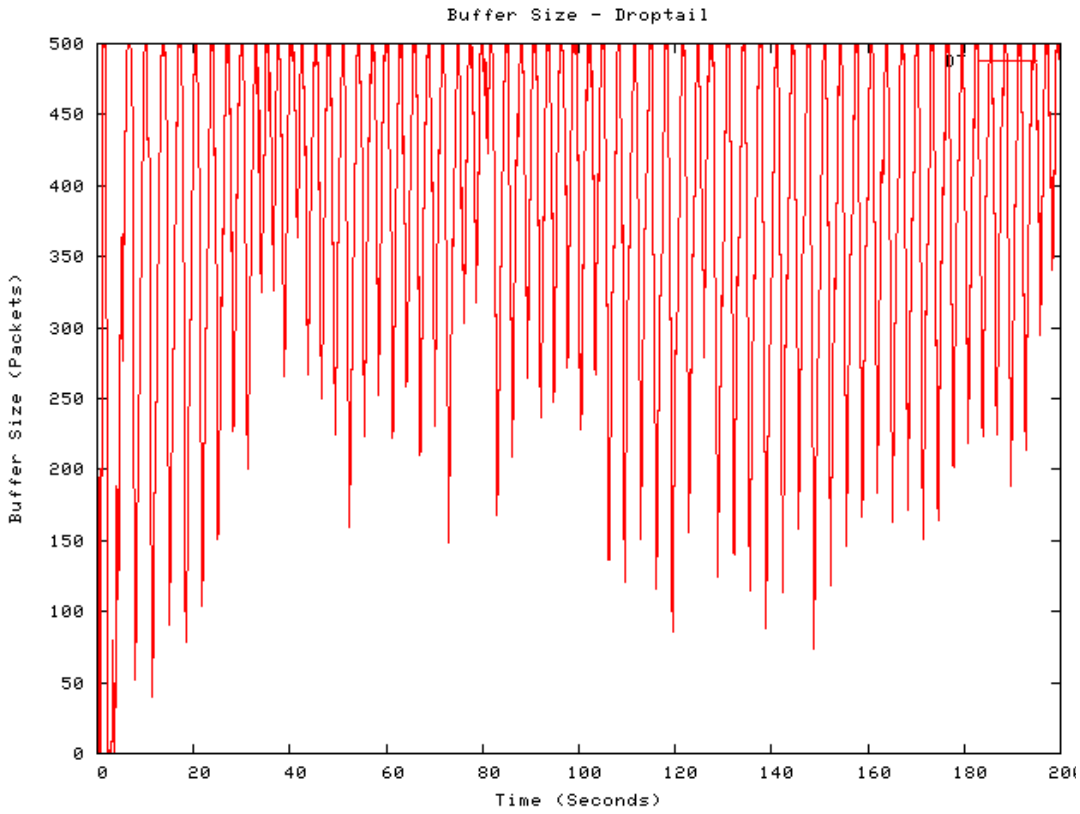


Figure 3.1 Drop Tail queue length dynamics

synchronisation, *lockouts*, and *full queues* (Pentikousis, 2001). *Global synchronisation* occurs when a significant number of TCP sources slows down at the same time and leads to underutilization of scarce network resources, like the bottleneck link. This phenomenon is likely to happen when a router drops many consecutive packets in a short period of time. With many TCP sources observing losses during the same period of time, an undesired cycle starts with periods of relatively low network utilization followed by heavy congestion. The *lockout* phenomenon may occur when DT allows a few connections to monopolize queue space (Ryu, Rump, & Qiao, 2003). Since the router sends back (implicitly) congestion signals to sources by means of packet losses only when the buffer has become full, a *full queue* phenomenon may persist for a long time under DT queue management. Full queues increase per-packet delays, and can lead to increased variation in delay (*jitter*).

Figure 3.1 illustrates the DT queue length dynamics over time at the bottleneck of a single-congested router network topology – shown in Figure 7.1 (see Chapter 7). The bottleneck link capacity is 15 Mbit/s with a propagation delay of *120 msec*. The number of TCP active flows, used in this simulation, is 100 long-lived File Transfer Protocol (FTP) flows, and 100 short-lived TCP flows (Web-like flows). It can be observed that the DT queue management scheme exhibits the *full queue* phenomenon, by keeping the buffer full, for a long period of time, and this has the negative effect of consecutive packet losses, and low useful throughput of the link. The frequent oscillations shown, indicates an undesired cycle with long periods of heavy congestion followed by relatively low network utilization.

One possible solution to overcome the drawbacks of the DT scheme is to drop packets before a queue becomes full so that a source can respond to congestion before buffers overflow. Another approach is to control the queue length, by regulating the flow of packets from the sources[*]. Therefore, in order to cope with the dynamic, time-varying nature of TCP/IP networks effectively, intelligent congestion control mechanisms for queue management are required at routers.

Thus the need for a robust enough controller to capture the dynamics, the highly bursty network traffic, and the nonlinearities of the controlled system leads to the introduction of *Active Queue Management* (AQM) mechanisms to assist the TCP congestion control to obtain satisfactory performance.

Due to the adherence to the current Internet standards next we focus on AQM mechanisms, which either drop or mark packets to indicate congestion, and also keep the TCP's window increase and decrease mechanism at the sources unchanged.

3.3 Active Queue Management Principles

The AQM approach can be contrasted with the DT queue management approach, employed by common Internet routers, where the discard policy of arriving packets

[*] For example, in this class of controllers, the router calculates the rate it will accept from a particular source over the next control interval. Various feedback schemes are proposed in the literature to transfer this feedback information to the source, as well as various algorithms to calculate the rate the router can accept over the next control interval.

is based on the overflow of the output port buffer. Contrary to DT, AQM mechanisms (Braden et al., 1998) start dropping or marking packets earlier in order to notify traffic sources about the incipient stages of congestion (TCP interprets dropped packets as congestion). AQM allows the router to separate policies of dropping packets from the policies for indicating congestion. In the case of dropping of packets the TCP congestion controller relies on the implicit feedback signal (generated by the lost packet as a timeout) to reduce the TCP congestion window. In the case of packet marking packets are not dropped, rather a bit is set on their header indicating congestion (hence termed *Explicit Congestion Notification,* ECN), and returned via the destination to the source. The use of ECN (Ramakrishnan, Floyd, & Black, 2001) was proposed in order to provide TCP an alternative to packet drops as a mechanism for detecting incipient congestion in the network. The ECN scheme requires both end-to-end and network support. An AQM-enabled gateway can *mark* a packet either by dropping it or by setting a bit in the packet's header, if the transport protocol is capable of reacting to ECN. The use of ECN for notification of congestion to the end-nodes generally prevents unnecessary packet drops.

Several schemes have been proposed to provide congestion control in TCP/IP networks (e.g., Floyd & Jacobson, 1993; Floyd, Gummadi, & Shenker, 2001; Hollot, Misra, Towsley, & Gong, 2002; Athuraliya, Li, Low, & Yin, 2001; Kunniyur & Srikant, 2004).

The main AQM performance characteristics include (Hollot, Misra, Towsley, & Gong, 2002):

- *Efficient queue utilization*: the queue should avoid overflow that results in lost packets and undesired retransmissions or emptiness that results in link underutilization.
- *Queuing Delay*: It is desirable to keep small both the queuing delay and its variations.
- *Robustness*: AQM scheme needs to maintain robust behaviour in spite of varying network conditions, such as variations in the number of TCP sessions, and variations in the propagation delay and link capacity.

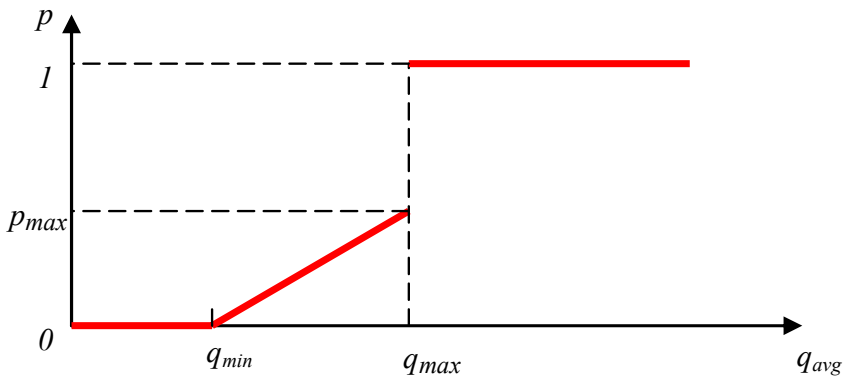


Figure 3.2 RED control law

A discussion of a number of representative, well-known, AQM schemes follows, in terms of modelling and control approach, and their limitations are identified.

3.4 Random Early Detection – A Linear Heuristic-based Technique

Random Early Detection (RED) (Floyd & Jacobson, 1993) was the first AQM algorithm proposed. It sets some minimum and maximum *drop/mark* thresholds in the router queues. In case the average queue size exceeds the minimum threshold, RED starts randomly *dropping/marking* packets, based on a linear heuristic-based control law (see Equation 3.1 and Figure 3.2), with a drop/mark probability depending on the average queue length, whereas if it exceeds the maximum threshold every packet is dropped.

$$p = \begin{cases} 0, & if\ q_{avg} < q_{\min} \\ \dfrac{q_{avg} - q_{\min}}{q_{\max} - q_{\min}} p_{\max}, & if\ q_{\min} \le q_{avg} \le q_{\max} \\ 1, & if\ q_{avg} > q_{\max} \end{cases} \qquad (3.1)$$

where $q_{\min}$ and $q_{\max}$ are the lower and upper thresholds, and $p_{\max}$ is the maximum drop/mark probability. The average queue length, q_{avg}, is updated at every packet arrival according to the *exponentially weighted moving average* (EWMA) method, as shown in Equation 3.2:

$$q_{avg}^{new} = (1-w) \times q_{avg}^{old} + w \times q_{inst} \tag{3.2}$$

where q_{inst} is the instantaneous queue length, and w is the averaging weight, $0 \le w \le 1$.

RED is designed to avoid "global synchronization", a case where all active sources reduce their sending rates at the same time that results in a fluctuation of link utilization, by introducing randomized packet dropping/marking.

The properties of RED and its variants have been extensively studied in the past few years. It is becoming clear that for successful implementation of RED based AQM (or its variants) in TCP/IP networks, there are still a number of unresolved issues. These include:

- Problems with performance of RED under different scenarios of operation and loading conditions. For example, the influence of: the moving average of the queue occupancy on the responsiveness of control; the loss function on the congestion control feedback mechanism; and the buffer size to the reaction time of the congestion controller is documented, see for example Kohler, Menth, and Vicari (2000). Note that numerous RED variants (e.g. Ott, Lakshman, & Wong, 1999; Feng, et al., 1999a, b) have been proposed, motivated by the difficulty in obtaining satisfactory performance under all network conditions. In Iannaccon et al. (2001) a comparative evaluation of RED with standard parameter settings, RED with optimal parameter settings (Ziegler, Fdida, & Brandauer, 2001), and Gentle RED (Floyd, 2000) showed no significant performance improvements with RED compared to DT queue management scheme that may justify deployment of RED in current backbone by ISPs; the performance of RED with standard parameters setting exhibits higher dependency on the traffic load dynamics than other mechanisms, indicating that fine tuning of the RED

parameters is not sufficient to cope with undesired RED behavior. It is worth noting that Gentle RED[†] addressed many of the RED deficiencies, but many serious deficiencies still remain (Iannaccon et al., 2001).

- Tuning of RED parameters has been an inexact science for sometime now, so much, so that some researchers have advocated against using RED, in part because of this tuning difficulty (May, Bolot, Diot, & Lyles, 1999; May, Bonald, & Bolot, 2000). Recently, some effort was undertaken to evaluate the performance of RED analytically. In May, Bonald, and Bolot (2000) the dependence of RED performance with respect to bursty traffic is studied. An approach to investigate RED in the presence of feedback traffic is presented in Peeters and Blondia (1999) where a source consists of a 3-state Markov model. Firoiu and Borden (2000) investigated the issue of recommendations of RED parameters and derived a set of recommendations of RED parameters for configuration of the RED queue size estimator: the frequency of queue sampling and the average weight. In Misra, Gong, and Towsley (2000) a more formal, control theoretic stand-point is adopted to also investigate the issue of recommending settings for the RED parameters. Ziegler, Fdida, and Brandauer (2001) derived quantitative models on how to set RED parameters with TCP traffic. The effective tuning of RED implies a "global" parameterisation that is very difficult, if not impossible to achieve, as it is shown in Feng (1999).

In (Ott, Lakshman, & Wong, 1999; May, Bolot, Diot, & Lyles, 1999; Misra, Gong, & Towsley, 2000) it is stated that one of RED's main weaknesses is that the average queue size varies with the level of congestion and with parameter settings. As a result, the mean queuing delay from RED is sensitive to the traffic load, as well as to the RED parameters. Consequently the throughput is also sensitive to the traffic load and to RED parameters. The authors of RED themselves have identified later on (Floyd, Gummadi, & Shenker, 2001) that: "RED does not perform well when the average queue becomes larger than *max threshold*, resulting in significantly

[†] Gentle RED (Floyd, 2000) modifies RED's drop/mark probability function in the case where the average queue size exceeds the maximum threshold; in that case, the drop/mark probability increases linearly between the maximum threshold and twice the maximum threshold.

decreased throughput and increased dropping rates. Avoiding this regime would require constant tuning of the RED parameters".

Adaptive-RED (A-RED) (Floyd, Gummadi, & Shenker, 2001), proposed by the one of the authors of RED (Floyd & Jacobson, 1993), attempts to solve the problem for the need of continuously tuning RED parameters by modifying a similar proposal by Feng, Kandlur, Saha, and Shin (1999b). In particular, A-RED adjusts the value of the maximum *drop/mark* probability to keep the average queue size within a target range half way between the minimum and maximum thresholds. Thus, A-RED maintains a desired average target queue length (TQL) twice the minimum threshold (if the maximum threshold is kept three times the minimum threshold). The adjustment of the maximum *drop/mark* probability is based on an additive fixed increase step when the average queue length exceeds the desired average queue, and on a multiplicative fixed decrease step when the average queue length goes below the desired average value, following a linear AIMD approach. Furthermore, A-RED also specifies a procedure for automatically setting the RED parameter of average queue weight as a function of the link capacity, following the approach in Ziegler, Fdida, and Brandauer (2001).

However, in Li, Zhang, Addie, and Clerot (2003) it is demonstrated that the fixed increase step size for the adjustment of the maximum drop/mark probability affects the performance of A-RED algorithm with loose convergence to the target average queue, as it is hard to quickly respond to traffic changes. The study in Li, Zhang, Addie, and Clerot (2003) shows that for different traffic conditions, different values of the increase step size would be preferred.

3.4.1 Control-Theoretic Design and Analysis of TCP/RED

Recently, some effort has been undertaken to evaluate the performance of RED-like AQM mechanisms analytically using control theory. For this purpose, the TCP/AQM flow dynamics are modelled and analysed in terms of feedback control theory. The design of such models is used to analyse the AQM system and provide guidelines/recommendations for correct configuration of AQM parameters that lead

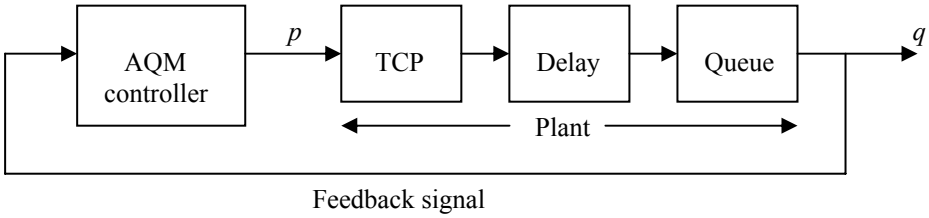


Figure 3.3 TCP/AQM feedback control system

to stable and robust operation, as well as to illustrate, in some cases, the difficulty of setting AQM parameters to stabilize TCP for the control approaches under study.

As shown in Figure 3.3, the TCP congestion control dynamics with an AQM scheme can be modelled as a feedback control system. It consists of (i) a plant - the controlled process – with TCP sources, receivers, and routers, (ii) an AQM controller – the controlling element – which generates as a control signal the packet drop/mark probability, p, (iii) the controlled variable queue length, q, and (iv), the feedback signal, which is a system output observed information.

In Firoiu and Borden (2000), RED implementation is studied and several structural problems are identified, such as large traffic oscillations. The system model Firoiu and Borden (2000) introduced is a deterministic nonlinear dynamic feedback system model, consisting of a derived plant function (queue law), averaging function, and RED control function. The exact form of the plant function depends on system parameters such as the number N of connections, the nature of the connections, and the round-trip delays. Potential problems of instability of the feedback system are identified, and therefore, a set of recommendations for configuration of the RED control law (frequency of queue sampling and the average queue weight) are derived.

Misra, Gong, and Towsley (2000) have developed a system of nonlinear differential equations for TCP/AQM dynamics using fluid-flow analysis that ignores the TCP slow start phase. Using the derived model, the role played by the RED configuration parameters on the behavior of the algorithm in a network is explained. It is believed that the RED averaging mechanism is a cause of tuning problems for RED.

A combined TCP and AQM model is designed and analyzed from a control theoretic standpoint in Hollot, Misra, Towsley and Gong (2001). Linearization is used to analyze a previously developed nonlinear model of the system (Misra, Gong, & Towsley, 2000). The forward-path transfer function of the plant depends on the number of connections, the round trip time, the link capacity, and a time delay factor. The analysis on an AQM system implementing RED is performed. Design guidelines are presented for choosing parameters that lead to stable operation of the linear feedback control system.

Low et al. (2002, 2003) studied the dynamics of TCP over RED queues through linearization around equilibrium points. The main conclusion of this paper is that "it is stability that largely determines the dynamics of TCP/RED", and not only because of noise traffic or its Additive Increase/Multiplicative Decrease (AIMD) probing. A general nonlinear model of TCP/RED is developed, and linearization of the model is done around the equilibrium in order to study local stability. The linear model generalizes the model of Hollot, Misra, Towsley, and Gong (2002). The results taken suggest that TCP/RED becomes unstable when the delay increases, or when the link capacity increases. The analysis done illustrates the difficulty of setting RED parameters to stabilize TCP: they can be tuned to improve stability, but only at the cost of large queues, even when they are dynamically adjusted.

A nonlinear modeling and analysis framework is provided in Ranjan, Abed, and La (2004). A deterministic nonlinear dynamical model for a simplified TCP network with RED control is used. This work goes beyond a simple linear stability analysis and studies regions where nonlinear instabilities occur due to the nonlinearity in the TCP throughput characteristic as a function of drop probability at the gateway. The model is shown to exhibit a rich variety of bifurcation behavior, leading to irregular operation. As system parameters are varied, the system dynamics are shown to transit between a stable fixed point and oscillatory and/or chaotic behavior (instability). In particular, the effects of various system and control parameters (such as, the averaging weight w, the lower queue threshold q_{min}, the number of connections, and the round trip delay) on average queue behavior are studied, as each of these

parameters is varied while the others are fixed. The results show that as the averaging weight is increased a chaos-type phenomenon appears. Also, the system becomes less stable as q_{min} is increased. On the other hand, the system stabilizes as the number of connections increases, though at the expense of increased delay. Furthermore, larger round trip propagation delays cause instability. The oscillatory behavior appearing in the system is due to the inherent nonlinearity of the interaction between RED mechanism and TCP.

A different approach is followed in Plasser and Ziegler (2004), where the above mentioned problems of a TCP/RED system are due mainly to the linearity of the drop/mark probability function of the RED algorithm, and not a matter of appropriate tuning of RED parameters.

Several papers have introduced models (as explained above) that aim to give appropriate configuration of RED parameters (such as averaging weight, minimum and maximum thresholds, maximum drop/mark probability) that can lead to stable and robust operation. However, as discussed earlier, it is a difficult task to find an appropriate configuration set that can be applied to broad network conditions. A major weakness of these models is that the selection of an appropriate parameter set is done for a specific operating point for which various system's parameters are assumed to be known (such as the number of flows and the round trip time).

As Plasser and Ziegler (2004) stated, even if the assumptions regarding the input parameters fit the specific scenario, the applicability of the RED algorithm would be restricted to a small range of the assumed values only. Therefore, the configured parameter set and the stability conditions introduced by the proposed models lack applicability to all possible real scenarios with varying dynamics of network conditions. As observed in Plasser and Ziegler (2004), during high load conditions a *disproportionately* higher drop/mark probability is required than in a low load condition, in order to keep the queue length in the same range, a requirement met only by a nonlinear drop/mark function. Such a nonlinear function fulfills the requirement of the queue length to remain between q_{min} and q_{max} for a much broader load conditions than the original linear RED function. Therefore, it is concluded that

the linear drop/mark probability function of RED itself is not robust enough for the highly bursty network traffic. The motivation should be to find a proper nonlinear function, rather than to find appropriate tuned RED parameters for a specific operating point for the original linear RED function.

3.5 Proportional Integral Control – A Linear Control Theory-based Technique

Hollot et al. use classical control system techniques to develop controllers well suited for TCP/AQM system (Hollot, Misra, Towsley, & Gong, 2002). Three key network parameters – the number of TCP sessions, the link capacity and the round-trip time – are related to the underlying feedback control system. It is determined that the queue averaging in RED algorithm is not beneficial. It also recommends an alternative AQM scheme, a *proportional-integral* (PI) control, on managing queue utilization and delay. The key feature is that PI control allows one to explicitly set the network queuing delay by introducing a desired queue length. Using a linearized TCP/AQM dynamics, the PI controller has been proposed not only to improve responsiveness of the TCP/AQM dynamics but also to stabilize the router queue length around the desired value. The latter can be achieved by means of integral control, while the former can be achieved by means of proportional control using the instantaneous queue length instead of using the average queue length.

The authors use a simplified version of a previously developed dynamic TCP model (Misra, Gong, & Towsley, 2000) – that uses fluid-flow and stochastic differential equation analysis, which ignores both slow start phase and timeout mechanism. This model is described by the coupled, nonlinear differential equations shown in Equation 3.3 (Hollot, Misra, Towsley, & Gong, 2002).

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)}{2}\frac{W(t-R(t))}{R(t-R(t))}p(t-R(t))$$

$$\dot{q} = \begin{cases} -C + \dfrac{N(t)}{R(t)}W(t), & q > 0 \\ \max\left\{0, -C + \dfrac{N(t)}{R(t)}W(t)\right\}, & q = 0 \end{cases} \tag{3.3}$$

where W is the average TCP window size, q is the average queue length, $N(t)$ is the number of TCP sessions, $R(t)$ is the round-trip time, which is equal to $\frac{q(t)}{C} + T_p$, C is the link capacity, and T_p is the propagation delay ($\dot{x}$ denotes the time-derivative). The first differential equation in (3.3) describes the TCP window control dynamic, and the second equation in (3.3) models the bottleneck queue length.

Using this simplified model, the linearization about an operating point is done to approximate these dynamics. To linearize (3.3) the authors have assumed that the number of TCP sessions is constant, that is, $N(t) \equiv N$. Given the vector of network parameters (N, C, T_p) the set of feasible operating points (W_o, q_o, p_o) is defined. The linearization about the operating point results in Equation 3.4 (Hollot, Misra, Towsley, & Gong, 2002).

$$\begin{aligned} \delta\dot{W}(t) &= -\frac{N}{R_o^2 C}\left(\delta W(t) + \delta W(t-R_o)\right) \\ &\quad - \frac{1}{R_o^2 C}\left(\delta q(t) + \delta q(t-R_o)\right) - \frac{R_o C^2}{2N^2}\delta p(t-R_o) \\ \delta\dot{q}(t) &= \frac{N}{R_o}\delta W(t) - \frac{1}{R_o}\delta q(t) \end{aligned} \tag{3.4}$$

where $\delta W = W - W_o$, $\delta q = q - q_o$, $\delta p = p - p_o$ represent the perturbed variables about the operating point. As indicated in Hollot et al. (2002) the linearization of the queue dynamics does not yield a pure integrator, but produces a leaky integrator with time constant R_o. This is explained by noting that the queue's arrival rate $\frac{NW}{R_o}$ is a

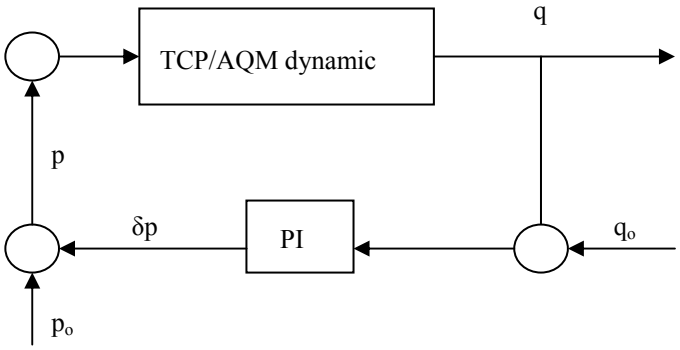


Figure 3.4 Implementation of the PI controller
(Hollot, Misra, Towsley, & Gong, 2002)

function of the round-trip time which, in turn, is a function of the queue length due to the queuing delay $\frac{q}{C}$.

A PI control law implementation with the TCP/AQM dynamic is shown in Figure 3.4 (Hollot, Misra, Towsley, & Gong, 2002). Implementing the PI controller in AQM-enabled routers, results in a difference equation, at time *t = kT, where* $T = \frac{1}{f_s}$ is the sampling period, as shown in Equation 3.5.

$$p(kT) = a\delta q(kT) - b\delta q\big((k-1)T\big) + p\big((k-1)T\big) \tag{3.5}$$

Design rules are given in (Hollot, Misra, Towsley, & Gong, 2002) for a PI controller for the linear control system to identify PI parameters (*a* and *b*). It is found that the PI controller stabilizes the feedback control system for all $N \geq N^-$ and all $R \leq R^+$, where N^- is considered to be a lower bound on the number of flows, and R^+ the upper bound on round-trip time. That is, by stabilizing against the largest expected R_o and C, and against the smallest expected N can lead to a robust AQM design.

However, the properties of this controller are not the desirable, since it depends on network dynamic parameters (such as the number of flows and the round-trip time). In Wang et al. (2004) an analysis for the lower bound of response time for PI, among others, is presented. As stated, since stability and response time are often in conflict

with each other in system performance, the PI scheme tries to find a tradeoff between them. If network parameters (especially the number of flows and the round-trip time) are known a priori, PI can adjust its control constant parameters to obtain the best response properly while guaranteeing stability. However, this is not the general case, since in a dynamic network it is difficult to obtain these parameters precisely. The analysis done shows that the response time of PI is dependent on their control constant parameters: buffer size, desired queue length, and desired stable packet drop/mark probability, p_o, which is an increasing function of the number of TCP flows and a decreasing function of round-trip time and link capacity. Under heavy congestion, PI suffers from a long response time (which is the opposite of what one wants during a congestion). In addition, with small buffer size, the responsiveness of PI will become worse as well.

3.6 Random Exponential Marking – An Exponentially Increasing Probability Function-based Technique

Athuraliya, Li, Low, and Yin (2001) proposed a new AQM scheme, namely *Random Exponential Marking* (REM). The key idea behind this AQM design is to stabilize both the input rate around link capacity and the queue length around a small target.

REM maintains a control parameter called "*price*" as a congestion measure. This is used to determine the *drop/mark* probability. "Price" is updated, periodically, based on two key factors:

- *rate mismatch*: the difference between input rate at the router and link capacity
- *queue mismatch*: the difference between queue length and the TQL.

The price is incremented if the weighted sum of these mismatches is positive; otherwise, it is decremented. The weighted sum of the two mismatches is positive when either the input rate exceeds the link capacity or there is excess backlog to be cleared, and negative otherwise.

In particular, the price $p(t)$ in period *t* is updated according to Equation 3.6.

$$p(t+1) = \max\left(0,\ p(t) + \gamma\left(\alpha\left(q(t) - q_o\right) + x(t) - c(t)\right)\right) \tag{3.6}$$

where $\gamma > 0$ and $\alpha > 0$ are small design constants, $q(t)$ is the aggregate buffer occupancy in period *t*, q_o is the target queue length (TQL), $x(t)$ is the aggregate input rate in period *t*, and $c(t)$ is the available bandwidth in period *t*. The difference $q(t) - q_o$ measures the queue mismatch, and the difference $x(t) - c(t)$ measures the rate mismatch. The design constant α trades off utilization and queuing delay during the transient period (it determines the prominence given to the queue length when determining the level of congestion), whereas the design constant γ controls the responsiveness of REM to changes in network conditions (it determines the speed of convergence of the algorithm).

Since the difference $x(t) - c(t)$ measures the rate at which the queue length grows, it can be approximated as $q(t+1) - q(t)$. Thus, Equation 3.6 can be reduced to Equation 3.7.

$$p(t+1) = \max\left(0,\ p(t) + \gamma\left(q(t+1) - \left(1-\alpha\right)q(t)\ - \alpha\, q_o\right)\right) \tag{3.7}$$

It turns out that the PI controller as expressed in (3.5) (Hollot, Misra, Towsley, & Gong, 2002) and REM as expressed in (3.7) are equivalent. Of course, REM uses the price, updated in (3.7) to further determine the drop/mark probability using an exponentially increasing technique (it can be said that REM has PI-type control structure and exponentially dropping/marking method for loss-probability).

In particular, the queue drops/marks each arrival packet with a probability that is calculated periodically and it is *exponentially increasing* in the current price. This is illustrated in Equation 3.8.

$$prob(t) = 1 - \varphi^{-p(t)} \tag{3.8}$$

where $prob(t)$ is the drop/mark probability in period *t*, $\varphi > 1$ is a design constant (it determines the range of loss or marking probability), and $p(t)$ is the price in period *t*.

A critical issue of concern is the proper selection of values for the design parameters of REM, namely α, γ, and φ. Athuraliya (2002) gives recommendations ($\alpha = 0.1$, $\gamma = 0.001$, $\varphi = 1.001$) based on simulation experiences. However, these design control parameters depend on network parameters, such as the number of sources and their delays, as well as capacities. Hence, as stated by Athuraliya (2002), given a set of network parameters, REM parameters can be tuned to optimize performance. However, this requires the knowledge of network parameters a priori, which is not practical.

Definitely, REM control parameters can significantly influence the overall performance of the proposed AQM scheme, as this can be understood by looking at (3.7) and (3.8). Thus the correct configuration of REM control parameters is still an issue for further investigation, concerning the dynamic, time-varying nature of TCP/IP networks.

3.7 Adaptive Virtual Queue-based Technique

Kunniyur and Srikant (2004) proposed an *Adaptive Virtual Queue* (AVQ) -based dropping/marking scheme for AQM. AVQ uses a modified token bucket model as a virtual queue to regulate link utilization, rather than the actual queue length. The AVQ scheme detects congestion solely on the arrival rate of the packets at the link.

In particular, the AVQ algorithm maintains a virtual queue whose capacity (i.e. the *virtual capacity*) is less than the actual capacity of the link, and whose buffer size is equal to the buffer size of the real queue. When a packet arrives in the real queue, the virtual queue is also updated to reflect the new arrival. Packets in the real queue are dropped/marked when the virtual buffer overflows. The virtual queue capacity at each link is then adapted to ensure that the total flow entering each link achieves a desired utilization of the link. Thus the virtual queue capacity is updated according to the differential equation 3.9.

$$\frac{d\widetilde{C}}{dt} = \alpha \left(\gamma C - \lambda \right) \tag{3.9}$$

where C is the link capacity, $\widetilde{C}$ is the virtual queue capacity, λ is the arrival rate at the link, γ is the desired link utilization, and $\alpha > 0$ is the smoothing parameter. As Kunniyur and Srikant (2004) stated, "the rationale behind (3.9) is that dropping/marking has to be more aggressive when the link utilization exceeds the desired utilization and should be less aggressive when the link utilization is below the desired utilization".

A critical issue of concern is the proper selection of values for the design parameters of AVQ, namely, γ and α. The desired utilization γ determines the robustness of the scheme. It allows the trade-off between high levels of utilization and small queue lengths. The design parameter α determines how fast one adapts the dropping/marking probability at the link to the changing network conditions. A design rule is given by Kunniyur and Srikant (2004) for choosing the parameter α using a control theoretic approach to ensure system stability.

Specifically, a fluid-model representation of the TCP flow control is used as proposed by Kunniyur and Srikant (2003), along with the AVQ scheme. The nonlinear TCP/AVQ model is then linearized to obtain conditions for local stability in terms of the round-trip delay, the number of users, the utilization of the link, and the smoothing parameter α of the AVQ scheme. Thus for a given feedback delay, utilization, and a lower bound on the number of users the design parameter α can be specified. However, in their study, only the single-router case accessed by TCP sources with the same RTT is considered. Also, the slow-start and timeouts are neglected in modelling the TCP dynamics, which strongly affect today's TCP in case of short-lived flows.

Definitely, the AVQ design parameter α can much influence the overall performance of the proposed AQM scheme, as this can be understood by looking at (3.9). Using the design rule proposed by Kunniyur and Srikant (2004) – for selecting the proper value of the design parameter α – which is based on the a priori knowledge of network parameters, like RTT and number of TCP flows can be impractical due to the dynamic nature of the network and the difficulty in obtaining these dynamic parameters precisely.

3.8 Limitations of Existing AQM Mechanisms

Some limitations of the existing AQM mechanisms that have been identified follow firstly and then some general remarks.

At first, as the RED-based algorithms control the macroscopic behaviour of the queue length (looking at the average) they often cause sluggish response and fluctuation in the instantaneous queue length. As a result, a large variation in end-to-end delays is observed.

A-RED attempts to tune the RED parameters for a robust behavior but fails to do so in various dynamic cases due to the fact that A-RED retains RED's basic linear structure. Thus, fine tuning of the RED parameters is not sufficient to cope with the undesired RED behavior.

The PI controller behaves in a similar way by exhibiting sluggish response to varying network conditions. This can be explained due to the fact that as the fixed/static PI parameters are dependent on network parameters, like the number of flows and round-trip time, it is difficult to get a stable operation in a broad range of dynamic varying traffic conditions. In addition, the PI controller is based on a simplified linear model which does not consider the slow start phase of TCP and timeout events. These modelling assumptions are not consistent with what real/live measurements indicate (Christiansen et al., 2001; Guo & Matta, 2001; Khalifa & Trajkovic, 2004; Padhye, Firoiu, Towsley, & Kurose, 2000), and should be taken into account. In particular, the TCP traffic constitutes 90 percent of all traffic with 50-70 percent of this TCP traffic being short-lived connections both in size and in lifetime (the so called *mice*) (Christiansen et al., 2001; Guo & Matta, 2001). In case of short-lived flows, TCP is strongly affected by slow start phases, with segment losses mostly being timeout losses (Khalifa & Trajkovic, 2004). Also, real TCP traces used in (Padhye, Firoiu, Towsley, & Kurose, 2000), where a stochastic model for the steady-state throughput of long-lived bulk transfer TCP flows is developed, contained more timeout loss events than fast retransmit events. Therefore capturing the effect of the timeout mechanism is important from a modelling prospective. Thus, the applicability of the derived model is questioned, when considering today's

Internet dynamics. The PI controller shows difficulties to accommodate itself to the complex, nonlinear, time-varying network status, with bursty, short-lived flows, and also unresponsive flows. This modelling approach is common, due to the difficulty in modelling the effect of slow start and/or timeout events; thus it is questionable whether the proposed models can capture the rich dynamics, and thus makes the analysis overly conservative.

Moreover, the TCP/AVQ model derived, also neglects the slow start phase of TCP and timeout events, making the AVQ scheme show difficulties in handling the dynamics and the nonlinearities of TCP. Also, the AVQ control parameters are dependent on network parameters, like the round trip delay and the number of flows. Thus, it is difficult, as discussed above, to get a stable operation in a broad range of dynamic varying traffic conditions.

The REM controller follows an equivalent to PI controller price control function, thus it is also found to exhibit sluggish response to varying network conditions. The correct configuration of REM control parameters is still an issue for further investigation, concerning the dynamic, time-varying nature of TCP/IP networks.

Some general remarks follow next. The existing AQM mechanisms still require a careful configuration of non-intuitive control parameters. Most of the AQM schemes have been designed by taking into account a simple network containing a single-congested router. Thus, the stability study of the controlled system, done by using a control-theoretic analysis, considers a single-router accessed by TCP sources with the same RTT. Further, most of the related simulation studies assumed idealized traffic, which differs significantly from real IP traffic. For example, most simulations have been performed assuming persistent (long-lived) connections, constant RTTs, and a limited number of connections. This traffic environment provides similar traffic conditions, and is very different from the dynamic, bursty traffic nature in real IP networks.

Furthermore, the dynamics of TCP/AQM models are mostly studied with the aid of linearization around equilibrium points of the nonlinear model developed, in order to study TCP/AQM stability around equilibrium. However, linearization fails to track

the system trajectories across different regions dictated by the nonlinear equations derived. As stated in Guirguis, Bestavros, & Matta (2003), linearization "assumes, and hence requires that the system always stays within a certain operating regime". Moreover, the equations modeled are dependent on various network parameters, such as the number of flows and the round trip delays, which in current Internet vary substantially. Therefore, the linearization around a specific operating point and the dependence on varying network parameters makes it difficult to get a stable and robust operation, in the case of TCP/IP networks with dynamic load and delay changes.

Hence, a major weakness of the proposed models is that the configuration of control parameters is done for a specific operating point for which various system's parameters are assumed to be known, and certain important dynamics are ignored. As stated in Plasser & Ziegler (2004), even if the assumptions regarding the input parameters fit the specific scenario, the applicability of the AQM algorithm would be restricted to a small range of the assumed values only. Therefore, the configured parameter set and stability conditions introduced by the proposed models lack applicability to all possible real scenarios with varying dynamics of network conditions.

In addition, even if the linearized system is made stable at equilibrium, there is no guarantee that the nonlinear system will remain stable (Plasser & Ziegler, 2004), especially if the deviations from the equilibrium are at times large. As stated in Low et al. (2003), instability is undesirable, and can cause three problems: (i) it increases variations in source rate and delay, (ii) it subjects short-lived transfers to unnecessary delay and loss, and (iii) it can lead to underutilization of network links if queues oscillate between empty and full.

3.8.1 Illustrative Example of Limitations

Possible limitations of the existing AQM schemes have been addressed in Section 3.8. A detailed demonstration of the performance of a number of representative, well-known, AQM schemes can be found in Chapter 7, where an extensive

simulative evaluation is done. In this section, we only give an illustrative example of how the PI AQM mechanism requires careful configuration of non-intuitive control parameters, and shows weaknesses to detect and control congestion under dynamic traffic changes, and a slow response to regulate queues.

Specifically, we conducted an experiment to investigate the influence of the network parameters (number of TCP flows and round trip time) on the behavior of the PI controller. As indicated in Section 3.5, the PI parameters (*a* and *b*, see Equation 3.5) are selected so as to stabilize the feedback control system for all $N \geq N^{-}$ and all $R \leq R^{+}$, where N^{-} is considered to be a lower bound on the number of flows, and R^{+} the upper bound on round-trip time. However, the PI parameters are fixed/static and it is difficult to get a stable operation in a broad range of dynamic varying traffic conditions. This is illustrated in Figure 3.5 and 3.6.

In particular, Figure 3.5 shows the queue length evolution of the PI controller in the case of *a single-bottleneck* network topology – shown in Figure 7.1 (see Chapter 7). The bottleneck link capacity is *15 Mbit/s* with a propagation delay of *120 msec*. The number of TCP active flows, used in this simulation, is 200 long-lived File Transfer Protocol (FTP) flows, and 200 short-lived TCP flows (Web-like flows). The TCP senders are grouped equally into 4 groups, with each group having a propagation delay increased by *5 msec*, starting from *5 msec* up to *35 msec* (that is, we create heterogeneous delays in the network). We also provide some time-varying dynamics by stopping half of the TCP/FTP flows at time *t = 70 sec* and resuming transmission at time *t = 150 sec.*

The default values of the PI parameters used are those given in Hollot, Misra, Towsley, and Gong (2002); that is: *a = 0.00001822, b = 0.00001816* (these values correspond to $N^{-} = 60$ *flows,* $R^{+} = 0.246$ *sec, C = 15 Mbit/sec*, and a sampling frequency of *170 HZ*).

From Figure 3.5, the poor performance of the PI controller can be seen in the case of an increased traffic load and high RTT. It cannot manage the queue, leading to sustained packet losses due to overflow. Also, at the time of the traffic changes, the

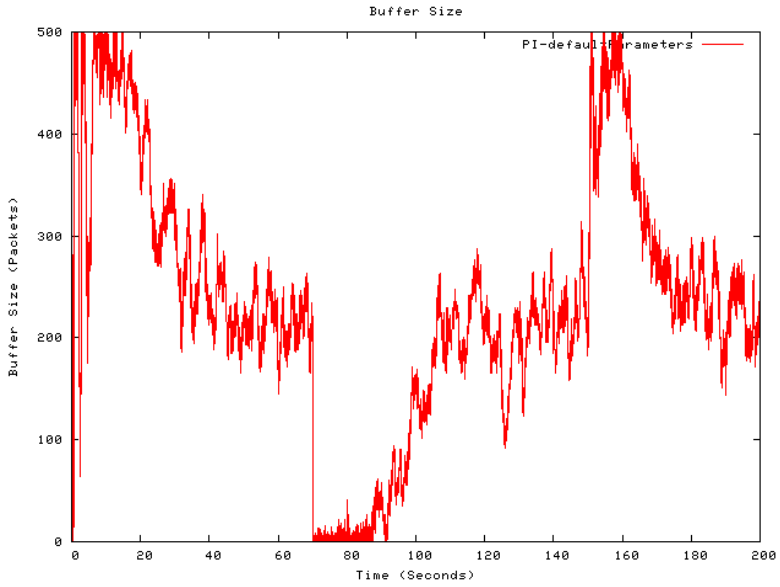


Figure 3.5 The PI Controller queue length dynamics with default parameter values

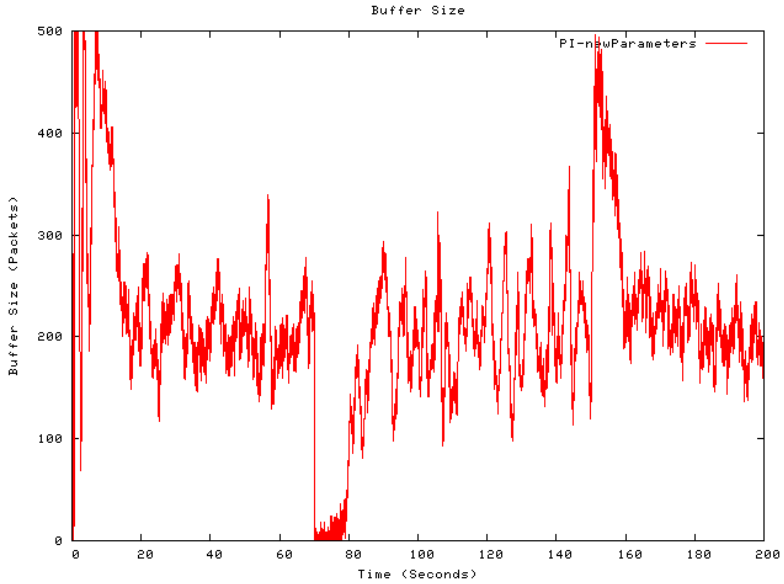


Figure 3.6 The PI Controller queue length dynamics with new parameter values

PI controller shows a sluggish response to maintain the queue at the desired queue length.

Then, we have manually changed the PI parameters (by following the design rules explained in Hollot, Misra, Towsley, and Gong (2002)), by considering the particular

network conditions for this experiment, that is, the increased number of TCP flows, that its lower bound can be approximated to 200, as well as the high RTT that can be approximated to *0.420 sec*), and the new result is shown in Figure 3.6. As it can be seen, the new values have improved the performance of the PI controller, as compared to the Figure 3.5, even though the sluggish response and large buffer fluctuations still exist. This experiment demonstrates a major weakness of the PI controller: having fixed parameters that are dependent on network parameters, like the number of flows and round-trip time. Hence, in the case of dynamic, varying conditions the PI controller fails to exhibit a robust behaviour.

3.9 Conclusions

The dynamic, time-varying nature of TCP/IP networks necessitates the design of robust, possibly intelligent, control methodologies to capture the dynamics, highly bursty network traffic, and nonlinearities of the controlled system, and obtain satisfactory performance. AQM mechanisms have been introduced to supplement the TCP based congestion control.

A number of representative AQM schemes in TCP/IP networks are studied; the modelling and control approach these follow are discussed, and their limitations are identified, including:

- The linearity of the control functions of existing AQM mechanisms that cannot capture effectively the nonlinearities of the TCP network.
- The dependency of AQM control parameters on dynamic network parameters, like the number of flows and the round trip propagation delays.
- The linearization of the existing models to allow analysis and design of AQM-based controllers, often making stability bounds overly conservative, and sluggish performance when dynamic changes occur.

- The accuracy of the existing TCP/AQM models, as they ignore the slow start phase of TCP and/or timeout events that are prominent conditions in today's Internet with the existence of short-lived TCP/Web flows.

Based on the above identified limitations it is evident that by using a nonlinear drop/mark probability function[‡], which does not require knowledge of dynamic system/network parameters, an effective and robust AQM system can be designed to drive quickly the system to be controlled into the steady-state. This should be contrasted with the linear drop/mark probability function that itself is not robust enough for the highly bursty network traffic and cannot capture the dynamics and nonlinearities of TCP/IP networks. For example, during high load conditions a *disproportionately* higher drop/mark probability is required than in a low load condition, in order to keep the queue length in the same range, a requirement met only by a nonlinear drop/mark function.

Thus, the complexity of these problems and the difficulties in implementing conventional controllers to eliminate those problems, as summarized in section 3.8, motivate the need to investigate intelligent control techniques to derive nonlinear control law, such as fuzzy logic, as a solution to controlling systems in which dynamics and nonlinearities need to be addressed. The capability to qualitatively capture the attributes of a control system based on observable phenomena is a main feature of fuzzy logic control and has been demonstrated in various research literature and commercial products. The main idea is that if the fuzzy logic control is designed with a good (intuitive) understanding of the system to be controlled, the limitations due to the complexity system's parameters introduce on a mathematical model can be avoided. A common approach in the networking literature is to either ignore such complex parameters in the mathematical model (e.g., ignoring the slow-start phase in the nonlinear model derived by Misra, Gong, and Towesly (2000)), or to simplify the model (e.g., ignoring the timeout mechanism and linearization of the model derived by Hollot, Misra, Towsley, and Gong (2001)) to such an extent (in order to obtain tractable model for controller design and/or stability results), which

[‡] In this study we use a nonlinear fuzzy logic based control methodology to derive a nonlinear drop/mark probability function

render the designed controllers and their derived stability bounds overly conservative.

Chapter 4

Differentiated Services Congestion Control

4.1 Introduction

The Differentiated Services (Diff-Serv) approach proposes a scalable means to deliver IP QoS based on handling of traffic aggregates. It operates on the premise that complicated functionality should be moved toward the edge of the network with very simple functionality at the core. The Diff-Serv framework enables QoS provisioning within a network domain by applying rules at the edges to create traffic aggregates and coupling each of these with a specific forwarding path treatment in the domain through use of a codepoint in the IP header. The Diff-Serv Working Group (WG) of IETF (DIFFSERV, 1998) has defined the general architecture for differentiated services and has focused on the forwarding path behaviour required in routers. The WG has also discussed the functionality required at Diff-Serv domain edges to select and condition traffic according to the rules. Further, AQM mechanisms are needed at the core of the Diff-Serv domain to provide bandwidth assurance, with low loss and bounded delay to various (aggregated) service classes.

4.2 Differentiated Services Architecture

Today's Internet provides a best-effort service to all of its applications; that is, it serves all users in a best-effort manner, and thus does not make any promises about

the QoS an application will receive. An application will receive whatever level of performance (for example, end-to-end packet delay and loss) that the network is able to provide at that moment. Thus the IETF, recently, has been under active discussion in various working groups to identify new architectural components that can be added to the Internet architecture to provide QoS in the Internet.

The *Integrated Services* (Int-Serv) architecture (Braden, Clark, & Shenker, 1994) is developed to provide individualized QoS guarantees to individual application sessions. Int-Serv allows sources and receivers to exchange signaling messages, which establish additional packet classification and forwarding state on each node along the path between them (Braden et al., 1997; Herzog, 2000). In the absence of state aggregation, the amount of states on each node scales in proportion to the number of concurrent reservations, which can be potentially large on high-speed links. Thus, Int-Serv failed to be adopted for widespread use, due to these scalability problems.

The IETF proposed a more evolutionary approach that did not require significant changes to the Internet infrastructure and provided *differentiation of services* (Blake et al., 1998). In particular, the Diff-Serv architecture aims to provide *scalable* and *flexible* service differentiation; that is, the ability to handle different “classes” of traffic in different ways within the Internet, without the need for per-flow state and signaling at every hop.

Service differentiation is desired to accommodate heterogeneous application requirements and user expectations, and to permit differentiated treatment, and pricing of Internet service. By *service* it is meant (Blake et al., 1998) “some significant characteristics of packet transmission. These characteristics may be specified in quantitative or statistical terms of throughput, delay, jitter, and loss, or may otherwise be specified in terms of some relative priority of access to network resources”.

This architecture achieves scalability by aggregating the traffic classification state, which is conveyed by means of IP-layer packet marking using the *Differentiated Services field* (DS Field) (Nichols, Blake, Baker, & Black, 1998). Packets are

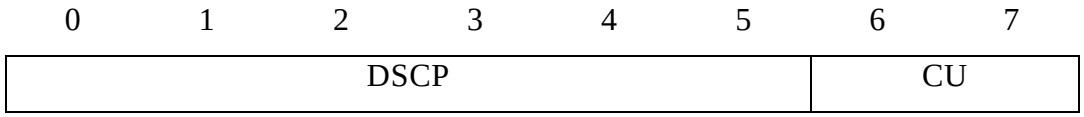


Figure 4.1 The DS Field structure

classified and marked to receive a particular *per-hop forwarding behavior* (PHB) on nodes along their path. Sophisticated *classification*, *marking*, *policing*, and *shaping* operations need only be implemented at network boundaries (at the edges of the network) or hosts, which thus reduces the operational complexity in the network core and makes it more scalable. Network resources are allocated to traffic streams by service provisioning policies, which govern how traffic is marked and conditioned upon entry to a differentiated services-capable network, and how that traffic is forwarded within that network.

The DS Field in the IP header, mentioned above, is intended to supersede the existing definitions of the IPv4 TOS (*type of service*) octet (Postel, 1981), and the IPv6 Traffic Class octet (Deering & Hinden, 1998). Six bits of the DS Field are used as a *codepoint* (DSCP) to select the PHB a packet experiences at each node. A two-bit *currently unused* (CU) field is reserved (see Figure 4.1). In the packet forwarding path, differentiated services are realized by mapping the DSCP contained in a field in the IP packet header to a particular forwarding treatment, or PHB, at each network node along its path. PHBs are expected to be implemented by employing queue management disciplines on a network node's output interface queue.

4.2.1 Diff-Serv Functional Elements

The Diff-Serv architecture is composed of a number of functional elements implemented in network nodes, including a small set of per-hop forwarding behaviors, packet classification functions, and traffic conditioning functions including metering, marking, shaping, and policing. This architecture achieves scalability by implementing complex classification and conditioning functions only at network boundary nodes, and by applying per-hop behaviors to aggregates of traffic, which have been appropriately marked using the DS Field in the IP packet

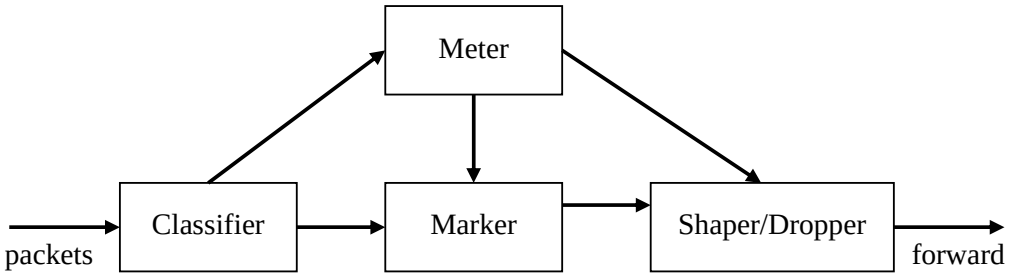


Figure 4.2 Logical view of packet classification and traffic conditioning at a Diff-Serv boundary node

header. Per-hop behaviors are defined to permit a reasonably granular means of allocating buffer and bandwidth resources at each node among competing traffic streams.

4.2.1.1 Classifiers

Packet classifiers select packets in a traffic stream based on the content of some portion of the packet header. The Behavior Aggregate classifier classifies packets based on the DSCP only. The Multi-Field classifier selects packets based on the value of a combination of one or more header fields, such as source and destination IP address, source and destination port numbers, DS Field, and protocol ID.

4.2.1.2 Traffic Profiles

A traffic profile provides rules for determining whether a particular packet is in-profile or out-of-profile. The concept of in- and out-of-profile can be extended to more than two levels, i.e. multiple levels of conformance with a profile may be defined and enforced.

4.2.1.3 Traffic Conditioners

A traffic conditioner may contain the following elements: meter, marker, shaper, and dropper (see Figure 4.2). A traffic stream is selected by a classifier, which steers the packets to a logical instance of a traffic conditioner. A meter is used to measure

the traffic stream against a traffic profile. The state of the meter with respect to a particular packet (e.g., whether a packet is in- or out-of-profile) may be used to affect marking, dropping or shaping action. A marker sets the DS Field of a packet to a particular codepoint, adding the marked packet to a particular Diff-Serv behavior aggregate. A shaper/dropper delays/discards some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. Traffic conditioners are usually located within Diff-Serv ingress and egress boundary nodes.

4.2.1.4 Per-Hop Behaviors

A per-hop behavior (PHB) is a description of the externally observable forwarding behavior (i.e., loss, delay, jitter) of a Diff-Serv node applied to a particular Diff-Serv behavior aggregate. The PHB is the means by which a node allocates resources to behavior aggregates, and it is on top of this basic hop-by-hop resource allocation mechanism that useful differentiated services may be constructed from. PHBs are implemented in nodes by means of some buffer management mechanisms on a network node's output interface queue. It is recommended (Nichols, Blake, Baker, & Black, 1998) that PHB implementations do not introduce any packet re-ordering within a flow.

The two PHBs being standardized are the Expedited Forwarding (EF) (Jacobson, Nichols, & Poduri, 1999; Davie et al., 2002), and the Assured Forwarding (AF) (Heinamen, Baker, Weiss, & Wroclawski, 1999).

- *Expedited Forwarding* (EF) PHB:

The EF PHB specifies a forwarding behavior that is intended to provide low delay, low jitter and low loss services by ensuring that the EF aggregate is served at a certain configured rate. This is done through a Service Level Agreement (SLA) during the connection setup. The departure rate of the aggregate's packets from any Diff-Serv node must equal or exceed a configurable rate. The EF traffic should receive this rate independent of the intensity of any other traffic attempting to transit the node.

If the EF PHB is implemented by a mechanism that allows unlimited preemption of other traffic (e.g., a priority queue), the implementation must include some means to limit the damage EF traffic could cause on other traffic (e.g., a token bucket rate limiter). Traffic that exceeds this limit must be discarded.

- *Assured Forwarding* (AF) PHB:

The AF PHB specifies a forwarding behavior in which packets are expected to see a very small amount of loss. The AF PHB group is a means to offer different levels of forwarding assurances for IP packets, and it provides delivery of IP packets in four independently forwarded AF classes (AF1, AF2, AF3, and AF4). Each AF class is, in each Diff-Serv node, allocated a certain amount of forwarding resources (buffer space and bandwidth), and should be serviced to achieve the configured service rate (bandwidth). Within each AF class, IP packets are marked with one of three possible drop precedence values (e.g., AF11, AF12, AF13). In case of congestion, the drop precedence of a packet determines the relative importance of the packet within the AF class. A congested Diff-Serv node tries to protect packets with a lower drop precedence value from being lost by preferentially discarding packets with a higher drop precedence value (e.g., packets non-conforming to contract); thus it differentiates flows with different drop preference levels.

Also, a Diff-Serv node does not reorder IP packets of the same flow when they belong to the same AF class, no matter if they are in- or out-of-profile (i.e., regardless of their drop precedence).

An AF implementation must attempt to minimize congestion within each class. This requires an active queue management algorithm. The queuing discipline mechanism must be insensitive to the short-term traffic characteristics of the flows using an AF class. Also, all packets within a single AF class and precedence level must be treated identically.

The AF PHB group could also be used to implement a low loss and latency (end-to-end delay) service by setting, for example, a low maximum limit to the buffer space available for an AF class.

4.2.2 Diff-Serv Service Classes

Currently, there is an ongoing work between active members of the networking community as to give recommended configuration guidelines for Diff-Serv Service Classes (Babiarz, Chan, & Baker, 2006). Service classes are defined based on the different traffic characteristics and required performance of the applications/services. This approach allows the mapping of current and future applications/services of similar traffic characteristics and performance requirements into the same service class.

A service class represents a set of traffic that requires specific delay, loss, and jitter characteristics from the network. It is essentially a statement of the required characteristics of a traffic aggregate that can be realized by the use of defined PHB.

Examples of recommended service classes are as follows (see also Table 4.1):

- *Telephony service class*: It is best suited for applications that require real-time, very low delay variation and packet loss and are of constant rate, such as IP telephony, and circuit emulation over IP applications. This type of service class should use EF PHB, and should also be configured to use a priority queuing system. Normally, traffic in this service class does not respond dynamically to packet loss; thus AQM should not be applied to EF marked packet flows.
- *Multimedia Conferencing service class*: It is best suited for applications that require real-time, very low delay service for rate adaptive traffic (i.e., have the ability to change the encoding rate), such as H.323/V2 and later versions of video conferencing service. This type of service class should use the AF PHB to provide a bandwidth assurance for AF41, AF42, and AF43 marked packets to ensure that they get forwarded. AQM should be used primarily to switch video encoding rate under congestion, changing from high rate to lower rate.

Table 4.1 Example of Diff-Serv Service Classes

Service Class	DSCP	AQM
Telephony	EF	No
Multimedia Conferencing	AF41, AF42, AF43	Yes per DSCP
Multimedia Streaming	AF31, AF32, AF33	Yes per DSCP
Low Latency Data	AF21, AF22, AF23	Yes per DSCP
High Throughput Data	AF11, AF12, AF13	Yes per DSCP

- *Multimedia Streaming service class*: It is best suited for applications that require near-real-time packet forwarding of variable rate elastic traffic sources. This type of service class should use the AF PHB to provide a minimum bandwidth assurance for AF31, AF32, and AF33 marked packets to ensure that they get forwarded. In such service class, an application has the capability to react to packet loss by reducing its transmission rate, such as streaming video and audio, web casts, etc. Thus, AQM should be used primarily to reduce forwarding rate to the minimum assured rate during network congestion.
- *Low Latency Data service class*: It is best suited for elastic and responsive applications, such as TCP short-lived flows (e.g., data processing applications, web-based transactions, etc). This type of service should use the AF PHB to provide a minimum bandwidth assurance for AF21, AF22, and AF23 marked packets to ensure that they get forwarded. Since this type of service is elastic and responds dynamically to packet loss, AQM should be used primarily to control TCP flow rates, when congestion occurs, by dropping packets from TCP flows that have large burst size.
- *High Throughput Data service class*: It is best suited for elastic applications that require timely packet forwarding of variable rate traffic sources, and more specifically is configured to provide good throughput for TCP long-lived flows, such as FTP, electronic-mail, etc. This type of service should use the AF PHB to

provide a minimum bandwidth assurance for AF11, AF12, and AF13 marked packets to ensure that they get forwarded in timely manner. Since this type of service is elastic and responds dynamically to packet loss, AQM should be used primarily to control TCP flow rates, when congestion occurs, by dropping packets from TCP flows that have higher rates first.

4.2.2.1 Aggregation of Diff-Serv Service Classes

Concurrently, there is an ongoing work to further aggregate (Chan, Babiarz, & Baker, 2006) the recommended configured Diff-Serv service classes (Babiarz, Chan, & Baker, 2006). It is believed that some network segments may be configured in such a way that a single forwarding treatment satisfies the traffic characteristics and performance requirements of two or more service classes. For such cases, it may be desirable to aggregate two or more service classes into a forwarding treatment. The treatment aggregates recommended in Chan, Babiarz, & Baker (2006) are designed to aggregate the service classes in such a manner as to protect real-time traffic, on the assumption that real-time sessions are protected from each other by admission control at the edge.

The performance requirements (tolerance to loss, delay and jitter) from the application/user are used as guidance on how to map the service classes into treatment aggregates. Examples of recommended aggregates of the service classes, defined in Table 4.1, are as follows (see also Table 4.2):

- *Real Time treatment aggregate*: All real-time (inelastic) service classes are aggregated. This treatment aggregate may include, among others, the Telephony and Multimedia Conferencing service classes. Traffic in each service class that is going to be aggregated into the treatment aggregate should be conditioned prior to aggregating. As such, there is a predictable and enforceable upper bound on the traffic that can enter such a queue, and to provide predictable variation in delay it must be protected from bursts of elastic traffic.

Table 4.2 Example of Aggregation of Diff-Serv Service Classes

Treatment Aggregate	Service Class	Treatment Aggregate Behavior	DSCP name
Real Time	Telephony, Multimedia Conferencing	EF	EF, AF41, AF42, AF43
Assured Elastic	Multimedia Streaming, Low Latency Data, High Throughput Data	AF	AF31, AF32, AF33, AF21, AF22, AF23, AF11, AF12, AF13

- *Assured Elastic treatment aggregate*: All elastic traffic that uses the AF model is aggregated. This treatment aggregate may include, among others, Multimedia Streaming, Low Latency Data, and High Throughput Data service classes. The DSCPs of the original service classes remain an important consideration and should be preserved during aggregation. Traffic bearing these DSCPs is carried in a common queue or class with a AF PHB. Since the traffic is elastic and responds dynamically to packet loss, AQM should be used primarily to reduce forwarding rate to the minimum assured rate, during network congestion.

4.3 Differentiated Services Congestion Control

The Diff-Serv architecture aims to provide aggregated QoS by using differentiated services aware congestion control algorithms. Recently, active queue management mechanisms (e.g. RED and its variants) have been proposed (Blake et al., 1998; Clark & Fang, 1998) within the framework of the Diff-Serv architecture (Braden et al., 1998) to preferentially drop non-conforming against conforming packets. Thus, the focus is on the development of differential dropping/marking algorithms for network core routers.

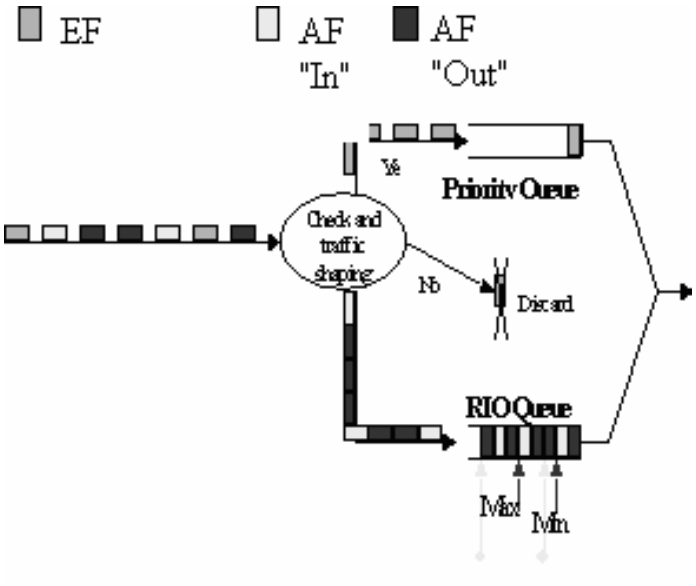
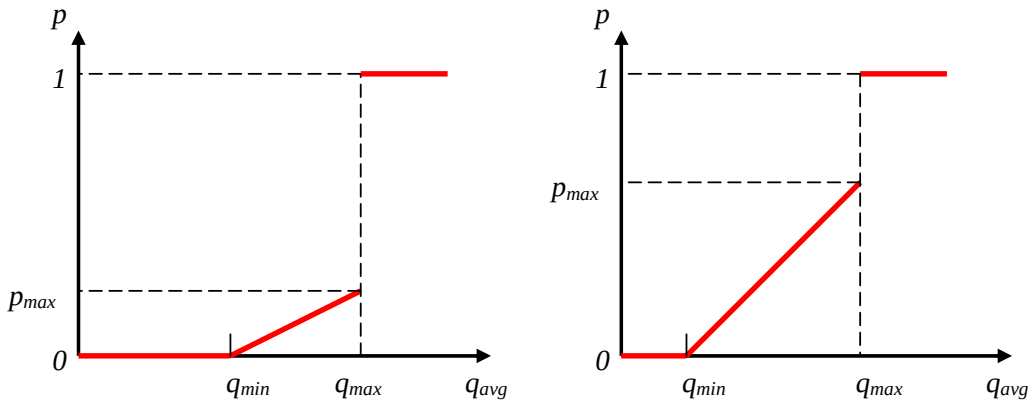


Figure 4.3 Diff-Serv scenario with RED queue for control

The most popular algorithm used for Diff-Serv implementation is RED (Floyd & Jacobson, 1993). The RED implementation for Diff-Serv defines that we have different thresholds for different drop precedence levels. The lowest drop precedence packets have the lowest minimum and maximum thresholds and therefore they are dropped/marked with larger probability than packets of a higher drop precedence level.

In Figure 4.3 we see a simple Diff-Serv scenario where RED is used for AQM based congestion control. A leaky bucket traffic shaper is used to check if the packets comply with the SLA. EF packets are discarded if they do not comply with the SLA. For the case of the AF class, both in- and out-of-profile packets share a RIO (Clark & Fang, 1998) queue. RIO stands for RED In/Out queue, where “In” and “Out” mean packets are in or out of the connection conformance agreement. EF packets use a separate high priority queue. There are different minimum and maximum thresholds (see Figure 4.4) for “In” (high priority) and “Out” (low priority) packets.

RIO uses the same mechanism as in RED, but it is configured with two different sets of parameters, one for “In” packets, and one for “Out” packets. Upon each packet arrival at the router, RIO checks whether the packet is tagged as “In” or



"In" – low-level of drop precedence "Out" – high-level of drop precedence

Figure 4.4 RIO control law

"Out". If it is an "In" packet, RIO calculates the average queue length of "In" packets only; otherwise (i.e., the packet is tagged as "Out") RIO calculates the total average queue length (i.e., of both "In" and "Out" arriving packets). The probability of dropping/marking an "In" packet depends on the average queue length of "In" packets, whereas the probability of dropping/marking an "Out" packet depends on the total average queue length.

The discrimination against "Out" packets is created by carefully choosing the parameters of minimum and maximum thresholds, and maximum drop/mark probability. As illustrated in Figure 4.4, RIO is more aggressive in dropping "Out" packets than "In" packets. It drops "Out" packets much earlier than it drops "In" packets; this is achieved by choosing the minimum threshold for "Out" packets smaller than the minimum threshold for "In" packets. It also drops/marks "Out" packets with a higher probability, by setting the maximum drop/mark probability for "Out" packets higher than the one for "In" packets.

However, as RIO is actually the RED implementation for Diff-Serv, it still suffers from the undesired RED behavior, as this was discussed in Chapter 3. May, Bolot, Jean-Marie, and Diot (1999) state, based on analytic evaluation of the loss probability, that the "choice of different RIO parameter values can have a clear impact on performance". RIO also retains RED's basic linear structure that itself is

not robust enough for the bursty network traffic and cannot capture the dynamics and nonlinearities of TCP/IP networks. Furthermore, RIO's decision for dropping/marking a packet of any level of drop precedence is not based on the total buffer occupancy; this may be a drawback if we want to have a bounded delay for the queue as a whole, and under any congestion level.

Apart from RED, the standard PI AQM (Hollot, Misra, Towsley, & Gong, 2002) was proposed for Diff-Serv AQM based congestion control. In particular Chait et al. (2002) proposed a two-level AQM controller for providing differential marking probabilities at the Diff-Serv core. The PI AQM scheme proposed by Hollot, Misra, Towsley, & Gong (2002) is used to preferentially drop/mark high-level of drop precedence than low-level, by introducing two set points (TQLs) for the core queue, which correspond to the two levels of drop precedence used, respectively. The drop/mark probability for both levels is computed by two PI AQM controllers, using the same parameter values, except for the TQL. The TQL of the low-level of drop precedence is set higher than the TQL of the high-level of drop precedence, in order to preferentially drop/mark packets of high drop precedence, during congestion. However, as the two-level PI controller is actually the PI implementation for Diff-Serv congestion control, it still suffers from the undesired PI behavior, as this was discussed in Chapter 3 (e.g., the dependency of PI control parameters on dynamic network parameters, like the number of flows and the round trip propagation delays, the linearity of the control law, etc).

4.4 Conclusions

Differentiated Services provide an approach to IP QoS that is modular, incrementally deployable, and scalable while introducing minimal per-node complexity (Blake et al., 1998). From the end user's point of view, QoS should be supported end-to-end between any pair of hosts. However, this goal is not immediately attainable. It will require interdomain QoS support, and many untaken steps remain on the road to achieve this (Nichols & Carpenter, 2001). Nevertheless, there is a need to use AQM for Diff-Serv congestion control at the core, to realize

multiple levels of drop/mark precedence, and thus offer an adequate differentiation among different traffic levels of priority. Further, there is a need to implement a low loss and delay service by setting, for example, a low maximum limit to the buffer space available for a traffic class or aggregate of classes. This can be achieved by the use of an effective AQM scheme, implemented at the core routers of the Diff-Serv domain, offering bandwidth assurance (high utilization, with low loss and bounded delay); thus addressing QoS issues to different traffic levels of precedence in TCP/IP networks. In this thesis, we investigate the suitability of Fuzzy Logic to provide effective control for differentiated services. At the same time, we investigate whether a generic fuzzy logic based AQM control methodology can be adopted to different TCP/IP environments, such as best-effort and Diff-Serv architectures.

Chapter 5

Fuzzy Logic

5.1 Introduction

Fuzzy logic is a logical system, which is an extension and generalization of multivalued logic systems. In recent years, fuzzy logic control has been found to be very useful in the realms of industrial systems and consumer products, as well as in communication data networks. Fuzzy logic control has strengths in controlling highly nonlinear, complex systems, which are commonly encountered in product design, manufacturing and control. Fuzzy logic provides a set of mathematical methods for representing information in a way that resembles natural human communication, and for handling this information in a way that is similar to human reasoning. By using fuzzy logic, a designer is able to blend qualitative linguistic expressions favored by human experts in the structure of control systems. A fuzzy logic controller can be conceived as a nonlinear controller whose input-output relationship is described in linguistic terms that can be better understood and easily modified (tuned). It is independent of mathematical models of the system to be controlled, thus achieving inherent robustness and reducing design complexity. Even for complicated processes whose models are not easily obtainable, one can still design a controller encapsulating the knowledge of the process operator or human expert.

5.2 Fuzzy Logic Principles

Fuzzy logic is one of the tools of what is commonly known as Computational Intelligence (CI). CI is an area of fundamental and applied research involving numerical information processing. While these techniques are not a panacea (and it is important to view them as supplementing proven traditional techniques), there is a lot of interest not only from the academic research community (e.g. Pitsillides & Sekercioglu, 2000; Sekercioglu, Pitsillides, & Vasilakos, 2001), but also from the telecommunications industry (e.g. Azvine & Vasilakos, 2000).

Fuzzy Logic Control (FLC) (Passino & Yurkovich, 1998) denotes the field in which fuzzy set theory (Zadeh, 1965) and fuzzy inference are used to derive control laws. A fuzzy set is defined by a membership function that can be any real number in the interval [0, 1], expressing the grade of membership for which an element belongs to that fuzzy set. The concept of fuzzy sets enables the use of fuzzy inference, which in turn uses the knowledge of an expert in a field of application to construct a set of "IF-THEN" rules. Fuzzy logic becomes especially useful in capturing human expert or operator's qualitative control experience into the control algorithm, using linguistic rules.

The idea of FLC was initially introduced by Zadeh (1973) and first applied by Mamdani (1974) in an attempt to control systems that are difficult to model mathematically. FLC may be viewed as a way of designing feedback controllers in situations where rigorous control theoretic approaches are too difficult and time consuming to use, due to difficulties in obtaining a formal analytical model, while at the same time some intuitive understanding of the process is available. The control algorithm is encapsulated as a set of linguistic rules, leading to algorithms describing what action should be taken based on system behaviour observations. FLC has been applied successfully for controlling numerous systems in which analytical models are not easily obtainable or the model itself, if available, is too complex and possibly highly nonlinear (e.g. Yasunobu & Miyamoto, 1985; Morales, Polycarpou, Hemasilpin, & Bissler, 2001).

Therefore, FLC concentrates on attaining an intuitive understanding of the way to control the process, incorporating human reasoning in the control algorithm. It is independent of mathematical models of the system to be controlled, thus achieving inherent robustness and reducing design complexity. This is in contrast with conventional control approaches that concentrate on constructing a controller with the aid of an analytical system model that in many cases is uncertain, nonlinear, and subject to noises. The capability to qualitatively capture the attributes of a control system based on observable phenomena is a main feature of FLC and has been demonstrated in various research papers as well as in commercial products. Thus, if the fuzzy logic control is designed with a good (intuitive) understanding of the system to be controlled, the limitations due to the complexity the system's parameters introduce on a mathematical model can be avoided. A common approach in classical control theory is to either ignore such complex parameters in the mathematical model, or to simplify the model to such an extent (in order to obtain some stability results), which render the designed controllers and their derived stability bounds overly conservative.

5.2.1 Fuzzy Sets

Fuzzy logic starts with the concept of a fuzzy set. A *fuzzy set* is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership. Thus, fuzzy sets are generalizations of ordinary/classical sets. In ordinary/classical sets, objects called elements of the set can claim only full membership or no membership at all. In fuzzy sets, *partial membership is allowed*.

A typical example is people height. In this case, it is unrealistic to say that, for example, two individuals who differ in height by less than a few centimetres belong to different classical sets, such as *short* and *tall* sets. Thus, the concept of *degree of membership* is proposed by Zadeh (1965), where the degree of membership is changed gradually from *false (0)* to *true (1)* rather than abruptly (see Figure 5.1).

In general, in fuzzy logic, *the truth of any statement becomes a matter of degree.* Reasoning in fuzzy logic is just a matter of generalizing the familiar *true-false*

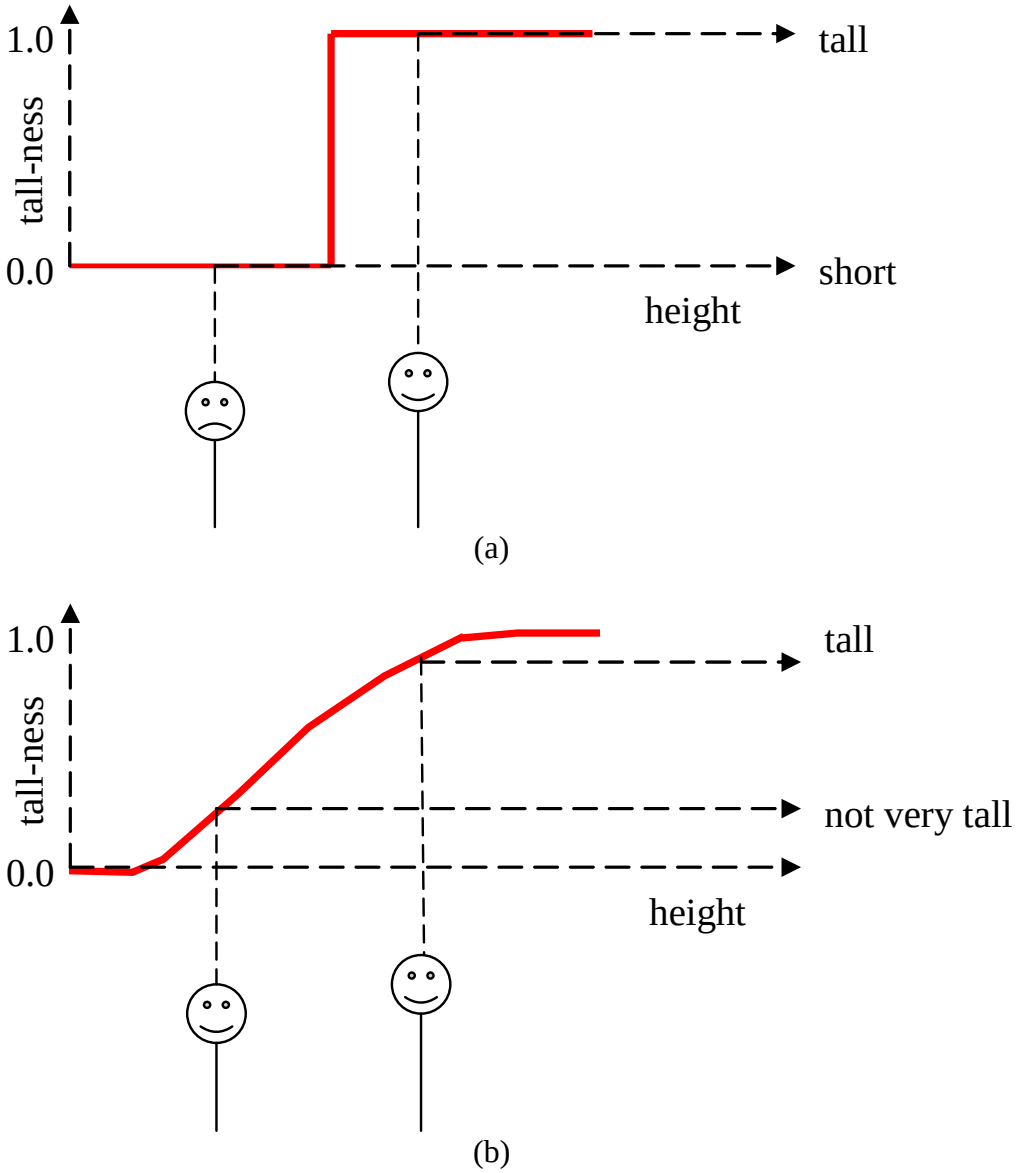


Figure 5.1 Example of (a) classical versus (b) fuzzy sets

(Boolean) logic. Figure 5.1(a) shows the truth values for “tall-ness”. In Figure 5.1(b) the truth value of “tall-ness” is shown, when we are allowed to respond with *multiple values.* Thus, fuzzy logic is a *multivalued* logic. Any value of height has a *partial membership* in the corresponding fuzzy set. The characteristic function that defines the fuzzy set is known as a *membership function.*

In fuzzy logic theory, the range of values for a given input or output space is often called the *universe of discourse* (Lee, 1990). For greater flexibility in fuzzy

controller implementation, the universes of discourse are “normalized” to a certain interval (e.g., [-1, 1] or [0, 1]) by means of constant scaling factors.

5.2.2 Membership Functions

A *membership function* is a curve that defines how each point in the *universe of discourse* is mapped to a membership value (or degree of membership) between 0 and 1, and is often given the designation μ.

In general, if X is the universe of discourse and its elements are denoted by x, then a fuzzy set A in X is defined as a set of ordered pairs (see Equation 5.1).

$$A = \{x, \mu_A(x) \quad | \ x \in X\} \tag{5.1}$$

where $\mu_A(x)$ is called the *membership function* of x in A. The membership function maps each element of X to a membership value between 0 and 1. Notice that a fuzzy set is simply a crisp set of pairs of elements of the universe of discourse coupled with their associated membership values.

There are various types of membership functions. The simplest membership functions are formed using straight lines. Of these, the simplest is the *triangular* membership function (see Figure 5.2). Further, the *trapezoidal* membership function has a flat top, and is just a truncated triangle curve (see Figure 5.3). These straight line membership functions have the advantage of simplicity. Other types of membership functions may include the Gaussian- and bell-shaped curves.

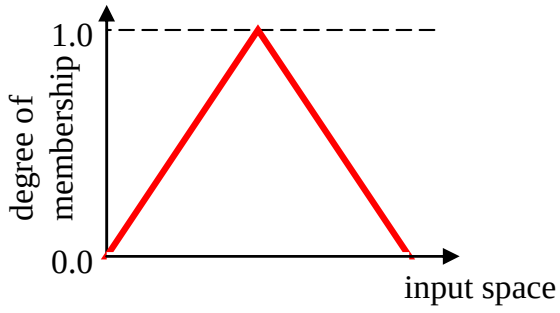


Figure 5.2 Triangular-type membership function of a fuzzy set

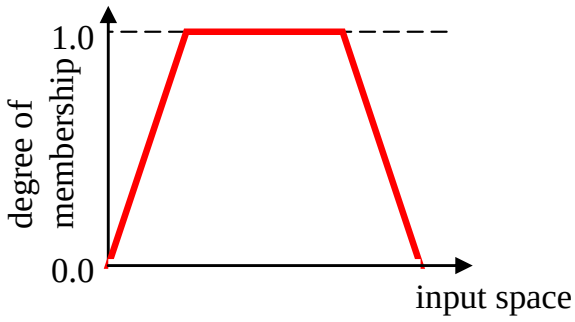
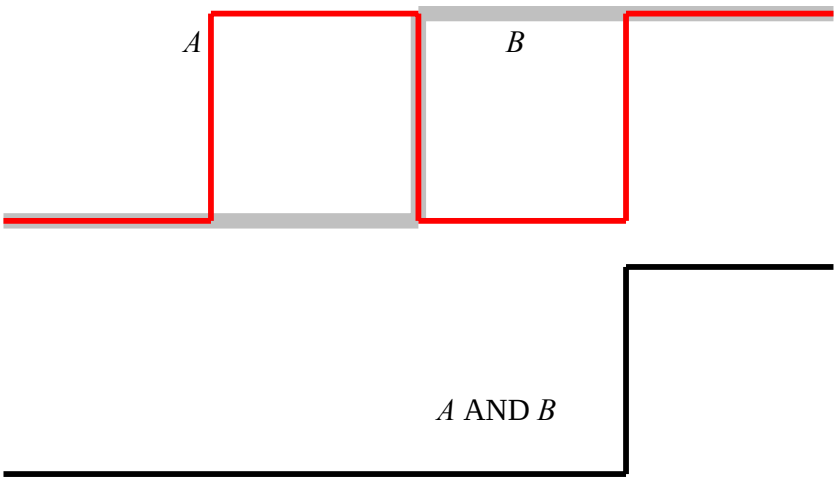
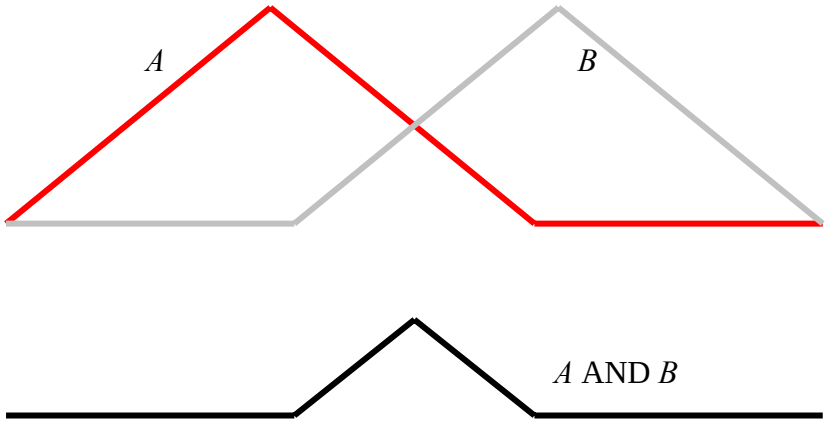


Figure 5.3 Trapezoidal-type membership function of a fuzzy set



(a) Two-valued (Boolean) logic



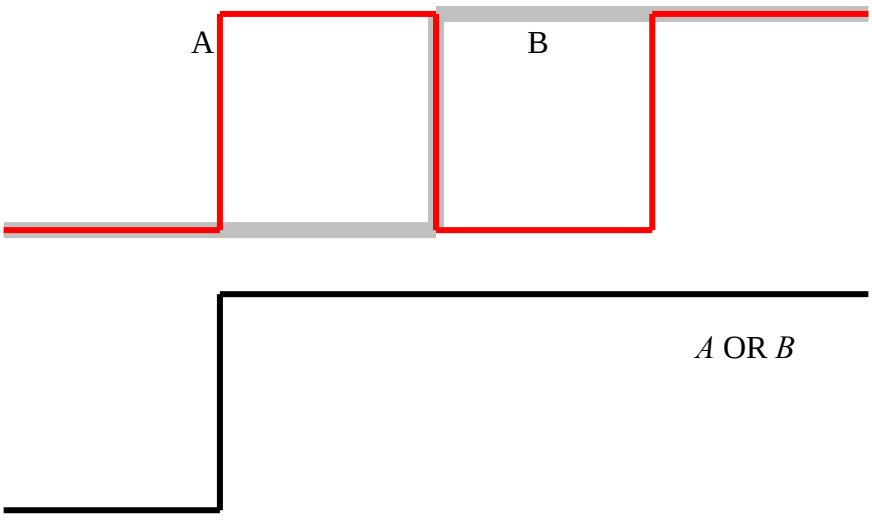
(b) Multivalued (fuzzy) logic

Figure 5.4 Intersection logical operation

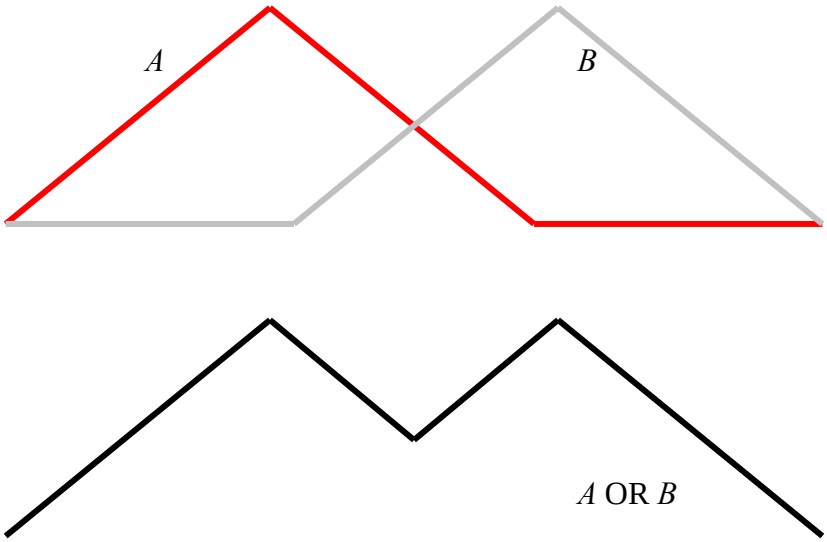
5.2.3 Logical Operations

Following the conventional fuzzy logic operations, initially defined by Zadeh, three basic operations fulfil the needs of most typical fuzzy logic based systems. Let A and B be fuzzy sets on a mutual universe of discourse with membership functions $\mu_A(x)$ and $\mu_B(x)$, respectively.

- *Fuzzy Intersection (AND):* The *min* operator represents the intersection of the two fuzzy sets A and B. That is, the elements of A and B are operated one-by-



(a) Two-valued (Boolean) logic



(b) Multivalued (fuzzy) logic

Figure 5.5 Union logical operation

one and the minimum of them is taken as the output (see Equation 5.2). This is illustrated in Figure 5.4, where the multivalued (fuzzy) logic operation of intersection is contrasted with the relative operation using two-valued (Boolean) logic.

$$\mu_{A\cup B}(x) = \min\left(\mu_A(x),\, \mu_B(x)\right) \tag{5.2}$$

- *Fuzzy Union (OR):* The *max* operator represents the union of the two fuzzy sets *A* and *B*. That is, the elements of *A* and *B* are operated one-by-one and the maximum of them is taken as the output (see Equation 5.3). This is illustrated in Figure 5.5, where the multivalued (fuzzy) logic operation of union is contrasted with the relative operation using two-valued (Boolean) logic.

$$\mu_{A\cap B}(x) = \max\left(\mu_A(x),\, \mu_B(x)\right) \tag{5.3}$$

- *Fuzzy Complement (NOT):* The fuzzy complement is obtained by using Equation 5.4.

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{5.4}$$

The above defined fuzzy logic operators are the classical ones used, because of their computational simplicity. Other logical operations, however, can also be defined.

5.2.4 IF-THEN Rules

Fuzzy sets and fuzzy logic operators are the so-called "subjects" and "verbs" of fuzzy logic, respectively. IF-THEN rule statements are used to formulate the conditional statements that comprise fuzzy logic.

A single fuzzy IF-THEN rule can have the form:

$$\textit{if } x \textit{ is } A \textit{ and } y \textit{ is } B \textit{ then } z \textit{ is } C \tag{5.5}$$

where *A, B* and *C* are *linguistic values* defined by fuzzy sets on the ranges (universes of discourse) *X, Y,* and *Z,* respectively. These linguistic values are part of their corresponding *linguistic variables*. While an algebraic variable takes numbers as values, a *linguistic variable* takes words as values. Using linguistic variables, such a variable, for example called "height", would assume (linguistically) values like "short", "tall", "very tall", etc. The linguistic values that describe the linguistic

variable “height” are defined by fuzzy sets that are represented by certain membership functions.

The IF-parts of the rule (5.5) are called the *antecedents* (or *premises*), while the THEN-part is called the *consequent*. The antecedent as well as the consequent of a rule can have multiple parts.

A fuzzy system contains generally a certain number of rules specifying the system behaviour against the input variables. This collection of fuzzy relational expressions representing the qualitative knowledge of the human operator forms the *rule base*. A fuzzy system performs reasoning on every rule in this rule base toward a final inference. However, the operations performed on these rules are simple, which is advantageous regarding computational processing.

5.2.5 Inference Process

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. Mamdani’s fuzzy inference method is the most commonly seen fuzzy methodology (Mamdani & Assilian, 1975) for controlling systems. Mamdani-type inference expects the output membership functions to be fuzzy sets. The steps needed to be followed are explained below. To illustrate the necessary steps needed to be taken, let us use the following fuzzy system as our example:

- Two input linguistic variables are used: the “queue length”, and the “rate of change of queue”.
    - The linguistic variable “queue length” is composed of three linguistic values: “empty”, “moderate” and “full”.
    - The linguistic variable “rate of change of queue” is composed of three linguistic values: “decreasing”, “zero” and “increasing”.
- One output linguistic variable is used: the “drop probability”, which is composed of four linguistic values: “zero”, “low”, “medium” and “high”.

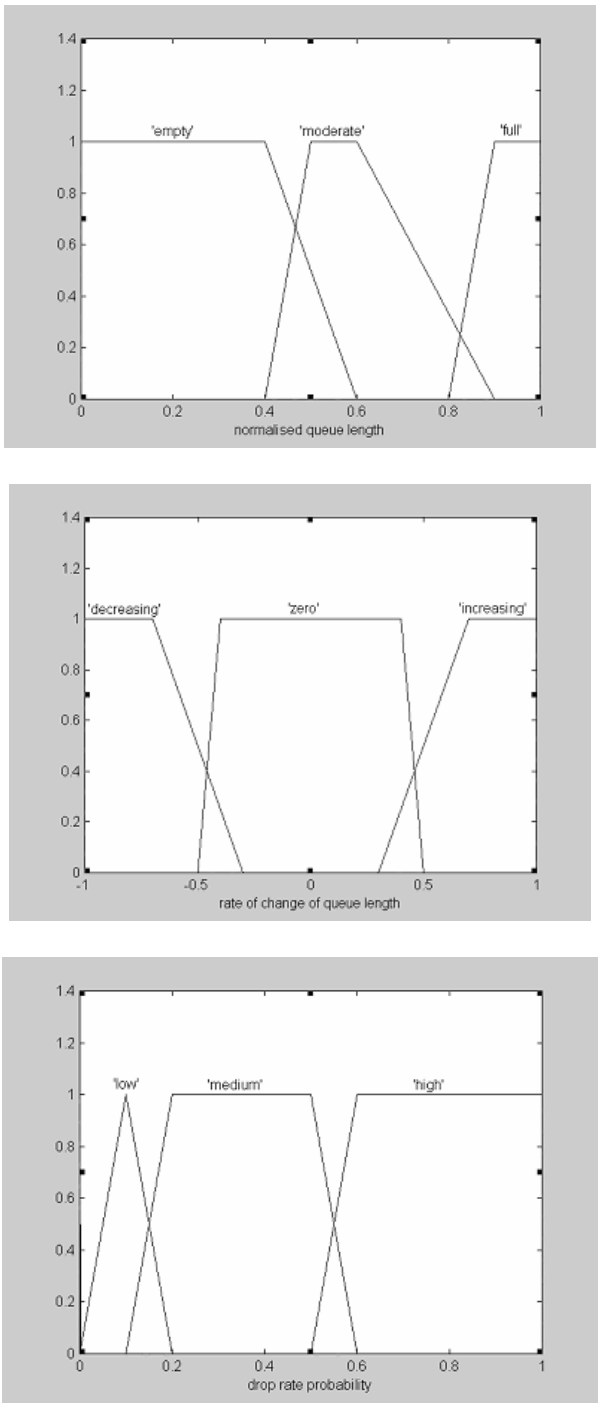


Figure 5.6 Membership functions of the linguistic values representing the linguistic variables of the fuzzy system used as an example

- The rule base of such a fuzzy system can be constructed by the following example rules:

  Rule 1: *if queue is empty then drop_probability is zero*

  Rule 2: *if queue is moderate and rate_of_change_of_queue is decreasing then drop_probability is zero*

  Rule 3: *if queue is moderate and rate_of_change_of_queue is zero then drop_probability is low*

  Rule 4: *if queue is moderate and rate_of_change_of_queue is increasing then drop_probability is medium*

  Rule 5: *if queue is full and rate_of_change_of_queue is decreasing then drop_probability is medium*

  Rule 6: *if queue is full and rate_of_change_of_queue is zero then drop_probability is high*

  Rule 7: *if queue is full and rate_of_change_of_queue is increasing then drop_probability is high*

The linguistic values of the three variables are defined by fuzzy sets with their membership functions shown in Figure 5.6.

5.2.5.1 Fuzzification of the Input Variables

The fuzzification of the input variables is carried out by taking the inputs of the fuzzy control system and determining the degree to which they belong to each of the appropriate fuzzy sets via their membership functions. Each input is a crisp numerical value limited to the universe of discourse of the associated input linguistic variable, and the output is a fuzzy degree of membership in the qualifying linguistic set (always in the interval between 0 and 1); that is, each input is “fuzzified” over all the qualifying membership functions required by the rules. This procedure is called “fuzzification” of the input, and amounts to either a table lookup or a function evaluation.

For example, if the normalized input variable “queue”, of the fuzzy system used as an example, has a current crisp value of 0.45, to what extent is the “queue” really

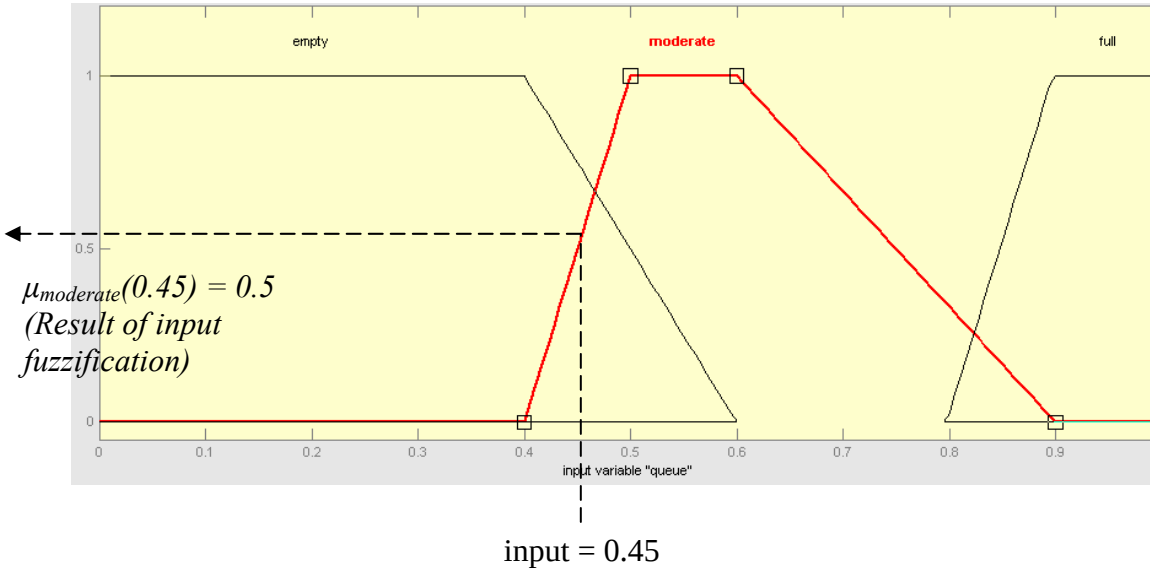


Figure 5.7 Fuzzification of the input variable

"moderate"? Given the graphical definition of "moderate queue" (see Figure 5.6), the "queue" being rated as 0.45, corresponds to $\mu_{\text{mod}erate}(0.45) = 0.5$ for the "moderate queue" membership function (see Figure 5.7). For such input value of queue, the corresponding degree of membership for the "empty queue" value is $\mu_{empty}(0.45) = 0.8$, and for the "full queue" is $\mu_{full}(0.45) = 0.0$.

5.2.5.2 Application of Fuzzy Operators

Once the inputs have been fuzzified, we know the degree to which each part of the antecedent has been satisfied for each rule. If the antecedent of a given rule has more than one part, the fuzzy operator used in the IF-part of the rule (as described in Section 5.2.3) is applied to obtain one number that represents the result of the antecedent for that rule (in the interval between 0 and 1). This number will then be applied to the output function. The input to the fuzzy operator is the membership values derived from the fuzzification of the input variables. The output is a single truth value. This is the degree of support for the rule.

For example, let us take rule number 3, of the fuzzy system used as an example, reproduced here:

Rule 3: *if queue is moderate and rate_of_change_of_queue is zero then drop_probability is low*

The IF-part of the rule has two antecedents. The fuzzy logic operator AND, forming the *Fuzzy Intersection logic operation* between the two antecedents, is represented by the *min* operator; that is, a particular element of *"moderate"* linguistic value of the linguistic variable "queue", and the one of "zero" linguistic value of the linguistic variable "rate of change of queue" are compared, and the minimum of them is taken as the output. For a particular instance, let us assume that the two inputs have values of 0.45 and 0.43, respectively (i.e., the normalized queue is 0.45, and the normalized rate of change of queue is 0.43). The two different pieces of the IF-part yielded the fuzzy membership values 0.5 and 0.8, respectively. The fuzzy AND operator simply selects the minimum of the two values, 0.5 (see Equations 5.6), and the fuzzy operation for rule 3 is complete (see Figure 5.8). Thus, *we are 0.5 (or 50%) certain that this rule applies to the current situation.* The rule indicates that if its antecedent part has a degree of truth (certainty) then the action indicated by its consequent part should be taken.

$$\mu_{\mathrm{mod}\,erate}^{queue}(0.45) = 0.5\,,\ \mu_{zero}^{rate-queue}(0.43) = 0.8$$

$$\mu_{antecedent} = \min\left(\mu_{\mathrm{mod}\,erate}^{queue}(0.45),\ \mu_{zero}^{rate-queue}(0.43)\right) = 0.5 \qquad (5.6)$$

where $\mu_{antecedent}$ is the minimum membership value of all antecedents of the IF-part of a particular rule.

5.2.5.3 Implication

A consequent is a fuzzy set represented by a membership function, which weights appropriately the linguistic characteristics that are attributed to it. The consequent is *reshaped* using a function associated with the antecedent (a single number). The input of the implication method is a single number given by the antecedent, and the output is a fuzzy set. Implication is implemented for each rule. The most popular, well-known implication method, which is also used by the fuzzy logic AND operator, is the *min* (minimum) that *truncates* the output fuzzy set. The justification

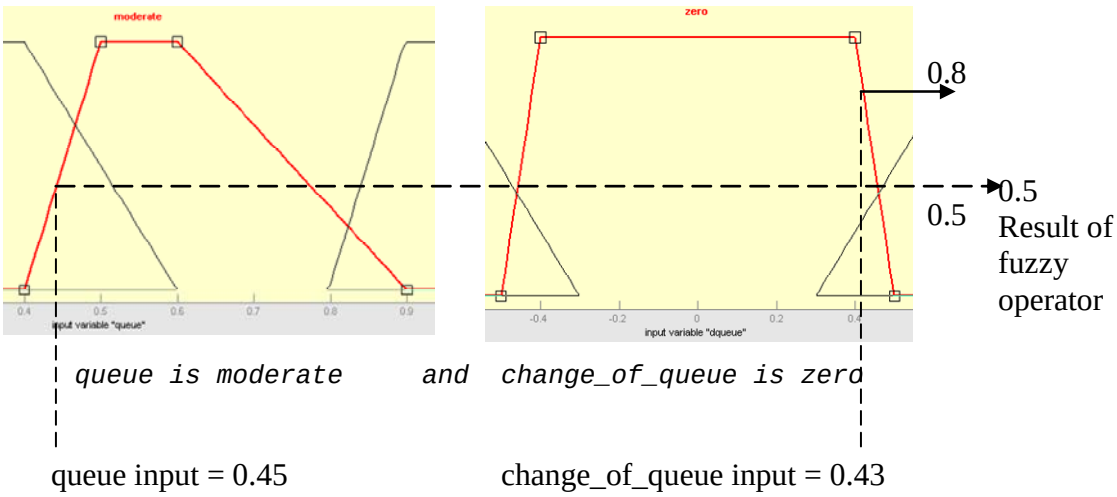


Figure 5.8 Application of the Fuzzy Operator AND (min)

of using the *minimum* operator to represent the implication is that *we can be no more certain about our consequent than our antecedent (premise)*.

Thus, the implication method shapes the consequent (the output fuzzy set) of a particular rule on the basis of the antecedent. What we achieve, is to get the implication of applying the result of the antecedents constituting the IF-part to the consequent in the IF-THEN rule.

Equation 5.7 shows how the *reshaped* (*implied*) output fuzzy set of a particular rule is constructed. The implication method used is the *min*. That is, all possible values in the universe of discourse of the specific output fuzzy set are compared with the generated antecedent single value, and the minimum is taken. Thus, the result is to get a truncated output fuzzy set, which is the outcome of the specific implication from the antecedents to the consequent:

$$\mu_{implied}(x_i) = \min\left(\mu_{antecedent}, \mu_{consequent}(x_i)\right) \tag{5.7}$$

where $\mu_{antecedent}$ is the minimum (if the *min* represents the fuzzy logic operator AND) membership value of all antecedents of the IF-part of a particular rule, and $\mu_{consequent}(x_i)$ is the membership value of the i^{th} element belonging to the universe of discourse of the consequent – output fuzzy set, which is updated whenever

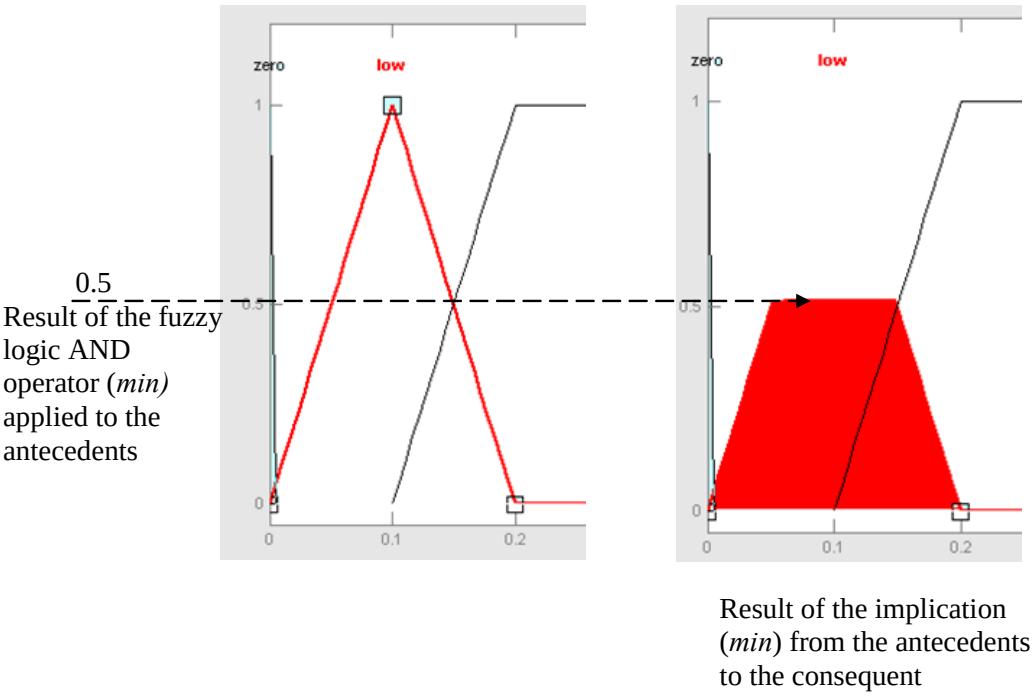


Figure 5.9 Application of the implication method (min)

$\mu_{antecedent} < \mu_{consequent}(x_i)$ to form the implied output fuzzy set with membership value $\mu_{implied}(x_i)$. This is illustrated in Figure 5.9, where we apply the *min* implication method to the rule number 3 of the fuzzy system used as an example. Remember that this rule is the following:

```
Rule 3: if queue is moderate and rate_of_change_of_queue is
        zero then drop_probability is low
```

Recall from the Equation 5.6 that the $\mu_{antecedent}$ = 0.5. Therefore, all possible membership values, in the interval between 0 and 1, of the output fuzzy set "low" are limited – truncated to the value 0.5. This clearly shows the implication of applying the result of the antecedents constituting the IF-part to the consequent in the IF-THEN rule. We see that the $\mu_{implied}(x_i)$ is in general a time-varying function that quantifies how certain the specific rule is that the output of the fuzzy logic system should take on certain values. It has a certain degree of truth that the output of the fuzzy system should lie in a region around *low* values.

5.2.5.4 Aggregation

Since decisions are based on the testing of all of the rules in the *rule base* of a fuzzy system, the rules must be combined in some manner in order to make a decision. *Aggregation* is the method by which the fuzzy sets that represent the resulted outputs of each rule are combined into a *single fuzzy set*. Aggregation only occurs once for each output variable. The input of the aggregation method is the list of output functions returned by the implication method for each rule. The output of the aggregation method is one fuzzy set for each output variable. It is important to notice that the order in which the rules are executed is unimportant. The most popular and well-known aggregation method, which is also used by the fuzzy logic OR operator, is the *max*, which has the properties discussed in Section 5.2.3. If this is the method used in the aggregation process, the resulting fuzzy set contains the maximum membership values among those generated by the implication process.

Equation 5.8 shows how the aggregated fuzzy set is obtained from the implied output fuzzy sets of all rules.

$$\mu_{aggregated}(x_i) = \max\left(\mu^1_{implied}(x_i),\ \mu^2_{implied}(x_i),\ ...,\ \mu^N_{implied}(x_i)\right) \tag{5.8}$$

where $\mu_{aggregated}(x_i)$ is the i^{th} membership value of the aggregated fuzzy output set, and $\mu^j_{implied}(x_i)$ is the membership value of the i^{th} element belonging to the universe of discourse of the consequent – output fuzzy set, obtained in the implication process of the rule number j ($1 < j < N$). This is illustrated in Figure 5.10, where we apply the *max* aggregation method to the fuzzy system used as an example. For a particular instance, let us assume that the two inputs have values of 0.45 and 0.43, respectively (i.e., the normalized queue is 0.45, and the normalized rate of change of queue is 0.43). We apply the *min*-operation for the fuzzy logic AND operator, and the *min-max* inference method, that is, the *min*-operation for the implication of each rule is selected, and the *max*-operation for the aggregation of all the resulted-implied output fuzzy sets is used. The result is to get a single aggregated output fuzzy set that needs however to be transformed into a single output crisp value.

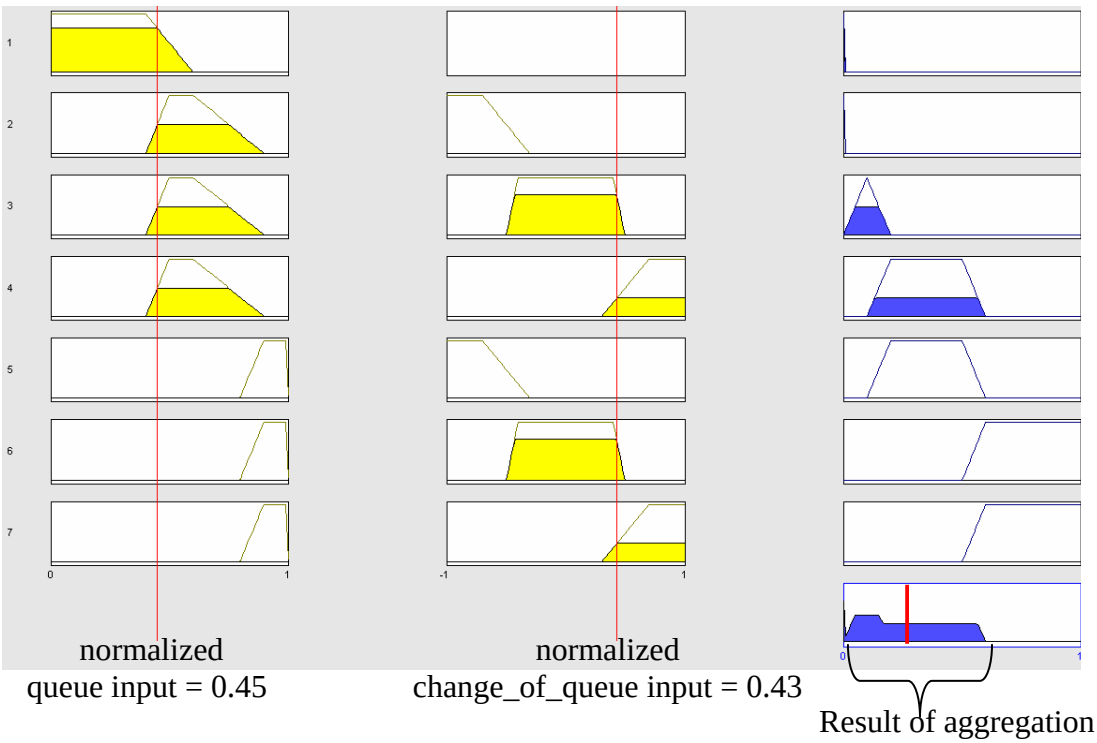


Figure 5.10 Application of aggregation method (min)

5.2.5.5 Defuzzification

The input for the defuzzification process is a fuzzy set (the aggregated output fuzzy set) and the output is a single number. As the aggregated fuzzy set encompasses a range of output values, a single output value must be resolved from the set. Looking at the example of the aggregation process illustrated in Figure 5.10, we can observe that two rules contribute to the production of the aggregated output fuzzy set, namely rule number 3 and 4. The implied output fuzzy sets of these rules have defined the linguistic values “low” and “medium”, respectively. Thus, the aggregated fuzzy set constitutes of a range of the maximum membership values of those implied fuzzy sets. What would be the final crisp output result? The answer to this question is found at the last step of the fuzzy reasoning process: the process of defuzzification, which converts the fuzzy reasoning output, which is a fuzzy set, into a crisp value that represents the whole inference process outcome.

There are various methods for defuzzification purposes. The most popular method is the *centroid* method, which returns the centre of area under the curve that

represents the aggregated output fuzzy set. Equation 5.9 gives, in the case of a continuous aggregated fuzzy set, the definition of the centroid.

$$p = \frac{\int_S y\mu_C(y)\,dy}{\int_S \mu_C(y)\,dy} \tag{5.9}$$

where $\mu_C(y)$ is the membership degree of y in the aggregated output fuzzy set C. The limits of integration correspond to the entire universe of discourse S of the output variable p. If discrete values are used, then Equation (5.9) may be simplified as shown in Equation 5.10.

$$p = \frac{\sum_{i=1}^{k} y_i \mu_C(y_i)}{\sum_{i=1}^{k} \mu_C(y_i)} \tag{5.10}$$

where the output universe of discourse S is discretized to k values.

The Centroid method is the most widely used method because it exhibits effective properties including: 1) the defuzzified values tend to move smoothly around the output fuzzy region, and 2) it is relatively easy to compute.

The Centroid method for defuzzification is illustrated in Figure 5.11, where we apply this method to the fuzzy system used as an example. Recalling the resulted aggregated output fuzzy set in Figure 5.10, the output crisp value of the fuzzy system is found, which is equal to 0.27. This value is at the start of the range of the "medium" value and very close to the range of the "low" value of the output linguistic variable.

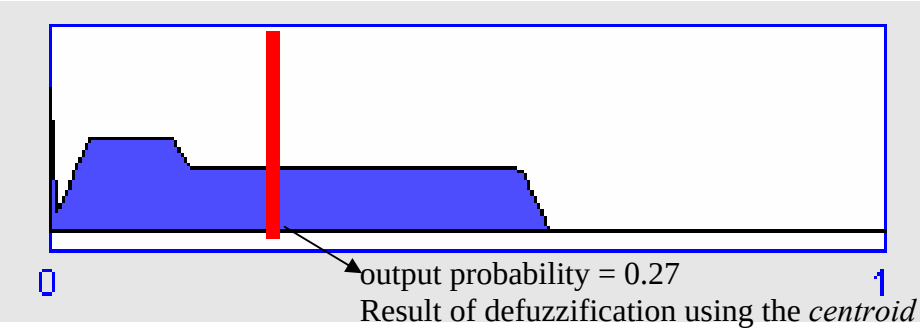


Figure 5.11 Application of defuzzification method (centroid)

5.2.6 Fuzzy Logic Control System

Having introduced the concepts that form the fuzzy logic theory, we summarize in this section the use of these concepts in designing a fuzzy logic Mamdani-type control system (Mamdani & Assilian, 1975). A block diagram of a fuzzy logic control system is given in Figure 5.12, where we show a fuzzy logic controller embedded in a closed-loop control system. The fuzzy logic controller is composed of the following four main components:

- A *rule base* that holds the knowledge of how best to control the system in the form of a set of IF-THEN rules. It contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.
- An *inference mechanism* (also called a "*fuzzy inference engine* (FIE)"), which emulates the expert's decision making in interpreting and applying knowledge about how best to control the plant. It basically evaluates which control rules are relevant at the current time, and then decides what the input to the plant should be.
- A *fuzzification interface*, which converts the fuzzy logic controller's inputs into information that the inference mechanism can use to activate and apply rules.
- A *defuzzification interface*, which converts the conclusions reached by the inference mechanism into crisp input(s) for the plant.

Basically, the fuzzy logic controller can be viewed as an artificial decision maker that operates in a closed-loop system in real time (Passino & Yurkovich, 1998). It gathers plant output data $y(t)$, compares it to the reference input $r(t)$, and then decides what the plant input should be to ensure that the performance objectives will be met.

Fuzzy logic control system design essentially amounts to (1) choosing the fuzzy logic controller input(s) and output(s), (2) choosing the preprocessing that is needed for the controller input(s) and possibly postprocessing that is needed for the output(s) (i.e, normalisation of the input and output values), and (3) designing each of the four components of the fuzzy logic controller shown in Figure 5.12.

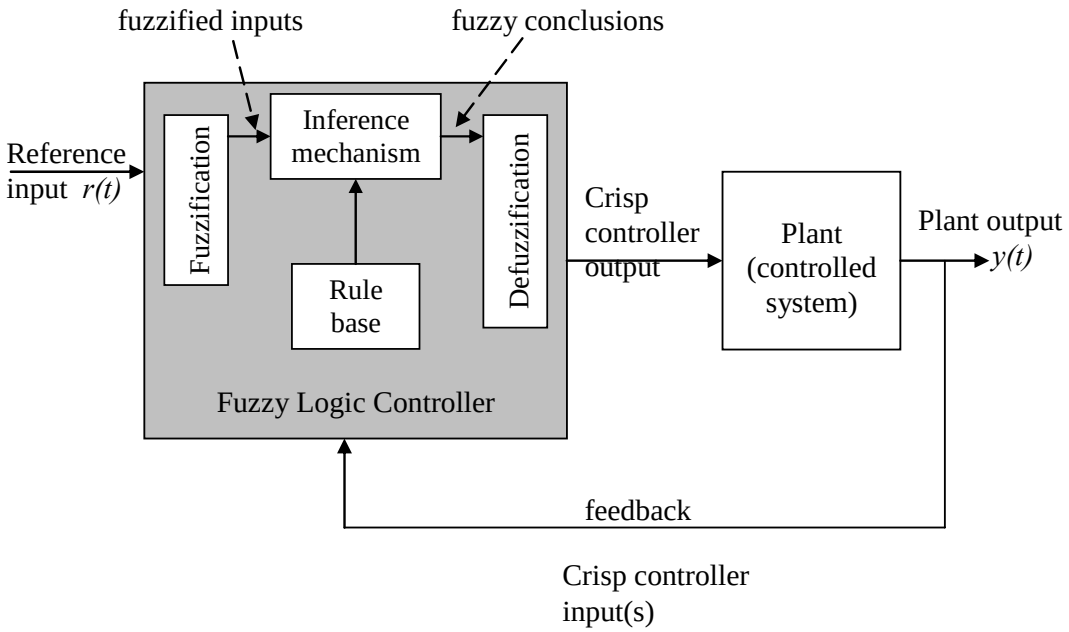


Figure 5.12 Fuzzy logic control system

A fuzzy logic control system (see Figure 5.12) is a nonlinear mapping between its inputs and output(s). The inputs and output(s) are crisp – real numbers. The fuzzification block converts the crisp inputs to fuzzy sets, the inference mechanism uses the fuzzy rules in the rule base to produce fuzzy conclusions (e.g., the implied fuzzy sets), and the defuzzification block converts these fuzzy conclusions into crisp output(s).

This nonlinear mapping from input to output implemented by the fuzzy logic controller is called the *control surface*. This mapping can be visualized by a nonlinear surface plot, where the controller's output is plotted against its inputs. Figure 5.13 shows such a control surface for the fuzzy logic system used as an example in this Section (i.e., a plane with three dimensions). Notice that the surface represents in a compact way all the information in the fuzzy logic controller. The rippled surface is created by the rules and the membership functions. There is a type of interpolation between the rules that is performed by the fuzzy logic controller. The output is an interpolation of the effects of the rules that are activated at the current time.

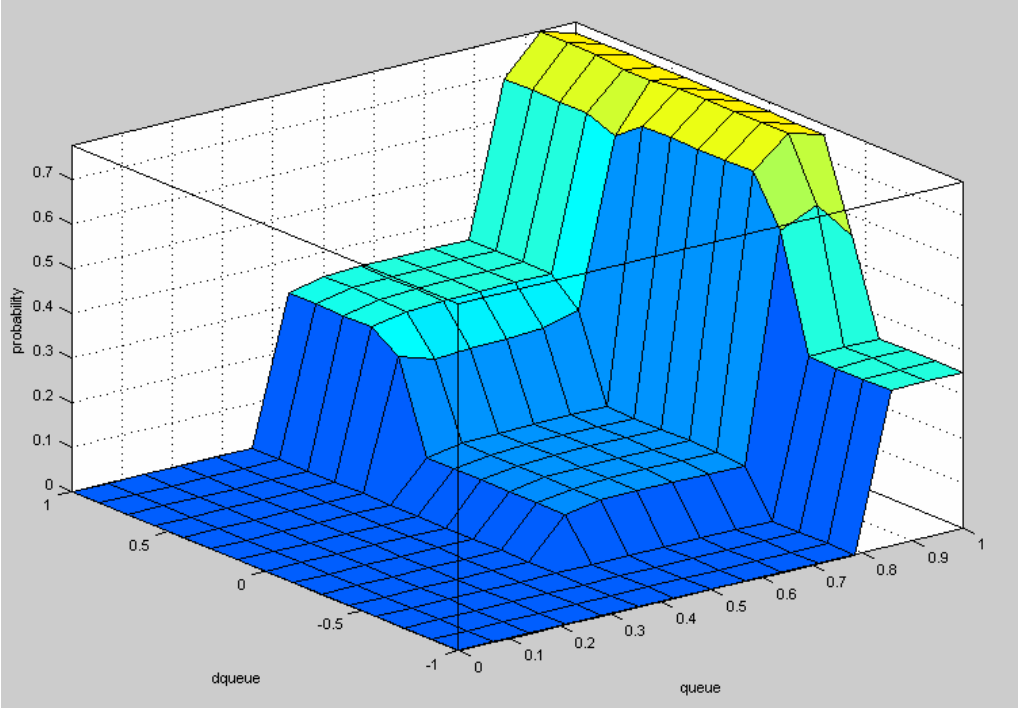


Figure 5.13 Nonlinear control surface of the fuzzy system used as an example

5.3 Application of Fuzzy Logic in Networks

Fuzzy Logic Control has been successfully used in a wide variety of applications in engineering, science, business, medicine and other fields. For instance, in engineering some potential applications areas include the following:

- *Robotics:* Position control and path planning
- *Process control:* Temperature, pressure, level control, and failure diagnosis
- *Manufacturing systems:* Scheduling and deposition process control
- *Automobiles:* Brakes, transmission, suspension, and engine control
- *Aircraft/Spacecraft:* Flight control, engine control, navigation, and failure diagnosis
- *Power industry:* Motor control, power distribution, and load estimation

Further, the use of fuzzy logic based control methods in communication data networks is becoming an effective alternative, non-conventional way of designing such controllers without relying on formal models of the controlled system.

In recent years, a number of research papers using fuzzy logic investigating solutions to congestion control issues in networking, especially in *Asynchronous Transfer Mode* (ATM) networks, have been published. Given the complexity of ATM networks, rich variety of traffic sources that operate on them, and difficulty of obtaining formal models for in depth analysis, it is not surprising to see that FLC was adopted by many researchers. For example, Sekercioglou, Pitsillides, and Egan (1994), Douligeris and Develekos (1995), Pitsillides, Sekercioglou, and Ramamurthy (1997), Pitsillides and Sekercioglu (2000), since early 90's, have successfully used the concept of FLC for congestion control in ATM, as an alternative to the conventional counterparts. A survey of these techniques is given by Sekercioglu, Pitsillides, and Vasilakos (2001).

Based on the vast experience of successful implementations of FLC in the design of control algorithms, as indicated above, the reported strength of fuzzy logic in controlling complex and highly nonlinear systems has recently started to be used in the IP world as well. To the best of our knowledge, fuzzy logic, in the concept of *active queue management* in TCP/IP networks, has been firstly introduced for providing congestion control by Pitsillides, Rossides, Chrysostomou, et al. (early 2000 and onwards). This novel research demonstrated that the application of fuzzy control techniques to the problem of congestion control in TCP/IP networks is worthy of further investigation. The main reasoning can be attributed to the difficulties in obtaining a precise enough mathematical model (amicable to analysis), using conventional analytical methods, while some intuitive understanding of congestion control is available.

Lately, we are witnessing an increase of research papers focusing on the use of fuzzy logic in various fields of the IP world. Fengyuan, Yong, and Xiuming (2002) have proposed a fuzzy controller for AQM in IP networks, with input variables the error on the queue length, and the rate of change of the error, while the output is the

increment step of the packet drop/mark probability. Their choice of the range of possible values (universe of discourse) of the rate of change of the error, as well as for the output variable, however, are dependent on the actual scenarios that have been depicted, and not a generic normalisation of all possible values that can appear. Specifically, as the authors stated in their paper, after running simulation experiments the maximum range of the rate of change of the error is set at half the buffer size. Further, they have chosen, using the same procedure, the range of the increment step size of the drop/mark probability to be $[-8.75 \times 10^{-5}, 8.75 \times 10^{-5}]$. Thus, this procedure is much dependent on specific scenarios, and questions the universality of the chosen ranges to be applicable in any dynamic network/traffic environment. This advocates the need of choosing the right inputs and output with generic normalised universes of discourse. Chrysostomou et al. have successfully addressed this problem by suitably choosing the right generic variables with normalised universes of discourse applicable in any possible network/traffic topology (see Chapter 6 for details). Further, Fengyuan, Yong, and Xiuming (2002) have used only the single-bottleneck link topology to evaluate their proposed controller, while a tandem network with multiple congested routers (which gives a more realistic picture of today'sTCP/IP networks) is not considered at all.

Wang, Li, Sohraby, and Peng (2003) proposed a fuzzy controller for AQM, with only one input – the queue length, while the output is the drop probability. However, this scheme is implemented by keeping the RED's algorithm semantics (they use the same threshold-based method as that in RED, i.e., when the queue length is less than a minimum threshold the probability is zero; when the queue length is between the minimum and maximum threshold the drop probability is computed, and when the queue length is greater than the maximum threshold then the same *gentle* mechanism in RED is used). In this way, the difficulty of RED parameters' tuning is not avoided. Also, by keeping a single input variable, the system dynamics are not captured more accurately than using multiple inputs. Another limitation of keeping only the queue length as input is the direct coupling between queue length and loss feedback that results in load dependent queue levels, as this is well-known by the study of the RED's behavior.

Aul, Nafaa, Negru, and Mehaoua (2004) proposed a fuzzy controller for AQM, with input variables the error on the queue length, and the rate of change of the error, while the output is the drop probability. No normalization of the input variables is done, as this can be seen by the control surface they provide (thus, questioning the suitability of the controller in a generic network topology), and also a limited performance evaluation is offered (a single scenario is used) using a single-bottleneck network topology that is not sufficient to demonstrate the wider applicability of the scheme.

Di Fatta, Hoffmann, Lo Re, and Urso (2003) proposed a fuzzy PI-controller for AQM, where the gains are tuned by a genetic algorithm. Although, this approach is indeed interesting, the results provided are not convincing for the suitability of the specific scheme as an effective candidate of AQM, as the proposed scheme shows almost identical (closed) behaviour with the PI counterpart scheme[*]. Thus, not much gain in terms of performance is demonstrated.

Fuzzy logic control has also been used, beside AQM, in other fields concerning today's Internet. Siripongwutikorn, Banerjee, and Tipper (2002) have proposed an adaptive bandwidth control algorithm based on fuzzy control to maintain the aggregate loss QoS. Habetha and Walke (2002) developed a new clustering scheme concerning mobility and load management, based on fuzzy logic. Wang et al. (2004) presented a fuzzy-based dynamic channel-borrowing scheme to maximize the number of served calls in a distributed wireless cellular network. Savoric (2003) proposed a fuzzy explicit window adaptation scheme that can decrease the advertised receiver window in TCP acknowledgements if necessary in order to avoid congestion and packet losses. Oliveira and Braun (2004) proposed a technique for packet loss discrimination using fuzzy logic over multihop wireless networks.

[*] The PI scheme is compared with our proposed approach later.

5.4 Conclusions

Fuzzy Logic Control defines a nonlinear control law by employing a set of fuzzy "IF-THEN" rules. The "IF" part describes the fuzzy inputs and the "THEN" part of a fuzzy rule specifies a control action (law) applicable within the fuzzy region from the "IF" part. FLC can be viewed as an alternative, non-conventional way of designing feedback controllers where it is convenient and effective to build a control algorithm without relying on formal models of the controlled system and control theoretic tools. For example, obtaining a formal (mathematical) model may prove infeasible for control system design (e.g., linearization of the nonlinear model may result in poor controlled system). The control algorithm is encapsulated as a set of commonsense linguistic rules. FLC has been applied successfully to the task of controlling systems for which analytical models are not easily obtainable or the model itself, if available, is too complex and highly nonlinear.

In this thesis, we adopt fuzzy logic control due to the significant property of attaining an intuitive understanding of the way to control the process, through incorporating human reasoning in the control algorithm. It is independent of mathematical models of the system to be controlled, thus achieving inherent robustness and reducing design complexity. This is in contrast with conventional control approaches that concentrate on constructing a controller with the aid of an analytical system model that in many cases is uncertain, nonlinear, and subject to noises. The capability to qualitatively capture the attributes of a control system based on observable phenomena is a main feature of FLC and has been demonstrated in various research papers as well as in commercial products. Thus, if the fuzzy logic control is designed with a good (intuitive) understanding of the system to be controlled, the limitations due to the complexity the system's parameters introduce on a mathematical model can be avoided. A common approach in classical control theory is to either ignore such complex parameters in the mathematical model, or to simplify the model to such an extent (in order to obtain some stability results, or to make model tractable for the controller design), which render the designed controllers and their derived stability bounds overly conservative. Further, the non-

intuitive tuning of conventional controllers makes it very difficult, due to the dependency on system dynamic parameters; thus are much affected by the dynamics, and the nonlinearities of the controlled system.

Of course, fuzzy logic control has its own limitations. Much work remains for the analytical study of fuzzy logic, particularly in the area of universally applicable global stability and performance analysis. Most proposed fuzzy logic controllers in literature do not have any stability analysis because of the difficulty in analysis. This is mainly due to the existence of the nonlinearity in the control structure that makes it difficult to conduct theoretical analysis to explain why fuzzy logic controllers in many instances achieve better performance than the conventional counterparts, especially for highly nonlinear processes.

However, as elegantly pointed out by Mamdani (1993), overstressing the necessity of mathematically derived performance evaluations may be counter productive and contrary to normal industry approach (e.g. prototype testing may suffice for accepting the controlled systems performance).

Nevertheless, the reported strength of FLC in controlling nonlinear systems (including many diverse commercial products) using linguistic information motivates the investigation of such intelligent control techniques as a solution to controlling TCP/IP networks in the form of intelligent AQM.

Chapter 6

Fuzzy Explicit Marking (FEM): An Intelligent Nonlinear Fuzzy Logic-based AQM Control Methodology in TCP/IP Best-Effort Networks

6.1 Introduction

The use of fuzzy logic control as an alternative to conventional based AQM mechanisms for congestion control is motivated from certain well-known limitations identified in the AQM literature. It is widely accepted that we need to design a nonlinear drop/mark probability, and that the control law must be easily tuneable over a wide range of operating conditions, (note that the requirement by certain schemes of non-intuitive dynamic system/network parameters for tuning makes this task very difficult). Also, the resultant nonlinear control law must exhibit desirable control properties, such as to drive quickly the system to be controlled into the steady-state (good transient and steady state response), and be robust.

Our aim is to adopt a methodology that deals with the uncertainties and high variability appearing in the network, and exhibits fast system response and robust behavior in spite of varying network conditions, without the need to (re)tune

complex control parameters. Our objective is to investigate the suitability of fuzzy logic based control that can exhibit such desirable properties.

6.2 The Need for the Alternative

As discussed in Chapter 3, the current Internet feedback mechanism for congestion control is binary and implicit and the network provides a best effort service. However, the existing TCP congestion avoidance/control mechanisms and its variants, while necessary and powerful, are not sufficient to provide good service in all circumstances (Braden et al., 1998). Therefore, network-assisted mechanisms have been introduced (e.g., ECN) to provide a more responsive feedback mechanism. The pressing need to better capture the dynamics and the highly bursty network traffic, and nonlinearities of TCP has lead to the design of AQM mechanisms as router support to the TCP congestion control.

While many AQM mechanisms (e.g., Floyd & Jacobson, 1993; Floyd, Gummadi, & Shenker, 2001; Hollot, Misra, Towsley, & Gong, 2002; Athuraliya, Li, Low, & Yin, 2001; Kunniyur & Srikant, 2004) have recently been proposed in the best effort TCP/IP environment, these require careful configuration of non-intuitive control parameters that are dependent on network/traffic parameters, and show weaknesses to detect and control congestion under dynamic traffic changes, and exhibit a slow response to regulate queues (Chrysostomou et al.).

Thus the complexity of these problems and the difficulties in implementing conventional controllers to eliminate those problems, as identified in Chapter 3, motivate the need to investigate intelligent control techniques, such as fuzzy logic, as a solution to controlling systems in which dynamics and nonlinearities need to be addressed. This work supplements the standard TCP to obtain satisfactory performance in a best-effort environment.

Fuzzy logic control has been widely applied to control nonlinear, time-varying systems, in which they can provide simple and effective solutions. Hence, the main objective is to investigate the *alternative* of using fuzzy control approach for AQM

control law rather than conventional approaches. Thus, the problem we solve is the following: *formulate and evaluate an intelligent TCP/AQM control methodology that performs adequately over a wide variety of network/traffic conditions, capturing the nonlinearities and dynamics of the process itself.* The main idea is to design a new probability function for packet dropping/marking, and the major interest is on the requirement of a n*onlinear control law* derived by a fuzzy logic based control methodology.

By using a nonlinear drop/mark probability function, which does not require knowledge of dynamic system/network parameters for tuning, an effective and robust AQM system can be designed to drive quickly the system to be controlled into the steady-state. This should be contrasted with the linear drop/mark probability function that itself is not robust enough for the highly bursty network traffic and cannot capture the dynamics and nonlinearities of TCP/IP networks over widely different operating conditions. For example, during high load conditions a *disproportionately* higher drop/mark probability is required than in a low load condition, in order to keep the queue length in the same range, a requirement met only by a nonlinear drop/mark function. On the other hand, the existing AQM schemes (Floyd & Jacobson, 1993; Floyd, Gummadi, & Shenker, 2001; Hollot, Misra, Towsley, & Gong, 2002; Athuraliya, Li, Low, & Yin, 2001; Kunniyur & Srikant, 2004) fail to achieve such an important requirement, either due to the linearity of the control law they obey, or/and the dependency of their control parameters on network/traffic parameters (such as the number of flows, RTTs). The control parameters are designed for "worst-case", which typically leads to degraded performance under normal dynamic situations. This has been identified and elaborated in Chapter 3.

6.3 Fuzzy Logic Control Methodology Design Goals

The *nonlinear fuzzy logic-based control methodology* (FLCM) is designed to operate in TCP/IP best-effort networks, and specifically in the IP routers' output port buffer. However, the aim is to achieve such design goals, as to be feasible to use the FLCM in other network architectures, as e.g. TCP/IP Diff-Serv networks; that is, to

design a generic control methodology that can be easily adopted in other network environments as well.

The proposed FLCM is an AQM approach for delivering an improved and more predictable, inherently robust congestion control implementation in TCP/IP networks. The fuzzy logic approach allows the use of linguistic knowledge to capture the dynamics of a nonlinear probability function, and by using multiple inputs can capture the dynamic state of the network more accurately. It supports the explicit congestion notification (ECN) in order to mark packets, when decided, instead of dropping them. This will prevent unnecessary packet drops. Drop of a packet happens only in the case of buffer overflow.

The principal aim of the proposed nonlinear FLCM is to achieve the following, very significant for an AQM scheme, goals:

- Dynamic and effective fast system response with robustness to the time-varying, dynamic nature of the controlled system
- High link utilization (based on the useful throughput)
- Minimal packet losses
- Bounded-regulated queue fluctuations and delays (mean and variation).

The bounded mean queuing delay and delay variation can be achieved by regulating the queues of the output port buffers of IP routers at predefined levels. This will have as a consequence to have low losses and to maintain high utilization as well. By having a nonlinear control law, based on fuzzy logic, the aim is to effectively deal with the uncertainties and high variability appearing in the network, and thus exhibit fast system response and robust behavior in spite of varying network conditions.

6.4 Fuzzy Explicit Marking System Model

The proposed FLCM in TCP/IP best-effort networks, called *Fuzzy Explicit Marking* (FEM) controller provides a new nonlinear probability function based on fuzzy logic for packet *marking*. Linguistic rules are used to represent how to control the plant, that is, to *mark* packets in TCP/IP networks. FEM controller has been implemented with *marking* capabilities, so that FEM-like routers have the option of either dropping a packet or setting its ECN bit in the packet header, instead of relying solely on packet drops (for the rest of the Thesis, by *marking* a packet it is meant setting its ECN bit).

In order to process the inputs of the fuzzy logic based FEM controller to get the output reasoning the following steps are needed:

- Identify the inputs and their ranges
- Identify the output and its range
- Construct the rule base that the system will operate under
- Create the degree of fuzzy membership function for each input and output
- Decide how the action will be executed for each rule
- Combine the rules and defuzzify the output.

There is no systematic procedure to design the fuzzy controller (Passino & Yurkovich, 1998). The most used approach is to define membership functions of the inputs and output, together with a rule data base and to test the controller. The fuzzy controller is nonlinear and it is very difficult to examine analytically the influence of certain parameters. Because of that, we rely on the use of heuristic expertise and study of the plant dynamics about how to best configure the control law. The focus is on the achievement of the design goals indicated in Section 6.3, whilst keeping the design of the controller as *simple* and *generic* as possible.

6.4.1 Selecting FEM Controller Inputs and Output

Our aim is to ensure that the controller will have the proper information available to be able to make good decisions, and will have proper control inputs to be able to steer the controlled system in the directions needed, so that it achieves a high-performance operation, as pointed out in Section 6.3.

Since multiple inputs are usually used not only to capture the dynamic state of the controlled system more accurately, but also to offer better ability to linguistically describe the system dynamics (Passino & Yurkovich, 1998), we utilize a two-input, single-output (the simplest of the Multiple Input Single Output (MISO) model based) fuzzy controller on the buffer of each output port of a router in TCP/IP networks.

An obvious information that is available to the controller is the queue length, as the controller operates on the router buffer queues. Thus, it can be considered as an input, as it can give a clear view of the local congestion status at a given time, like the level of queuing delay. As one of our design goals is to put bounds on the queue fluctuations and delay, we aim to regulate the queue at a specified level. This leads to the introduction of a constant reference input, the desired-*target queue length* (TQL) that can be set by the network operator at a level that can achieve acceptably low mean queuing delay and variation. With the aim of keeping the instantaneous queue size at a specified level, this can provide tighter control and can avoid losses and unacceptable levels of fluctuation around the reference point. This is in contrast with RED-based algorithms that since they control the macroscopic behavior of the queue length (i.e., average), they often cause sluggish response and fluctuation in the instantaneous queue length, and as a result an important variation in delay has been observed (Kohler, Menth, & Vicari, 2000). Thus, the first controller input is chosen to be the *error on the instantaneous queue length* at a given time. The mismatch of the instantaneous queue length and the TQL means deviation from equilibrium state and the necessity of updating the packet *mark* probability.

The selection of the *rate of change in error* – as a second input on the controller, which is found in many research papers concerning feedback systems, needs careful setting of appropriate range of values. As pointed out by Passino and Yurkovich

(1998) the range of possible values for such an input could be determined by experimenting with various input values to the controlled system to determine the normal range of values that the *change in error* can reach during one sampling interval, in order to effectively define partitions over that input space; thus this kind of input and, consequently, the choice of its normalizing scaling gain, is application-dependent (see earlier discussion in Chapter 5, Section 5.3). However, considering the dynamics, the time-varying behavior and the nonlinearities of the controlled system under investigation (i.e., TCP/IP networks), makes it difficult to determine a correct and effective range of values of such a controller's input; thus resulting in a less generic controller.

The same considerations apply for a possible choice of the *increment* step of the packet *mark* probability as the controller's output. The effective range of such an increment needs a vast experimentation that makes it hard to determine such a range.

This advocates the need to choose the right inputs and output with generic normalised universes of discourse, applicable in any network/traffic environment.

Thus, the decision is to use the *error on the instantaneous queue length* for two *consecutive sampling intervals*; that is, the current error on queue length and the error on queue length with a specified delay (at the previous sampling interval). Sampling at every packet arrival, just like RED does, is an overkill and provides no perceptible benefit.

By measuring the queue at two consecutive sampling intervals (the current and the past), we attempt to estimate the future behavior of the queue (can be interpreted as a prediction horizon).

Notice that the difference between the input arrival rate at a buffer and the link capacity at discrete time intervals can be approximated and visualized as the rate at which the queue length grows when the buffer is non-empty. Thus, as it is usually easier to sample queue length than rate in practice, we track the change of the queue length for two consecutive discrete time intervals. That is, we compare the instantaneous queue length with its previous sampled value. The system converges

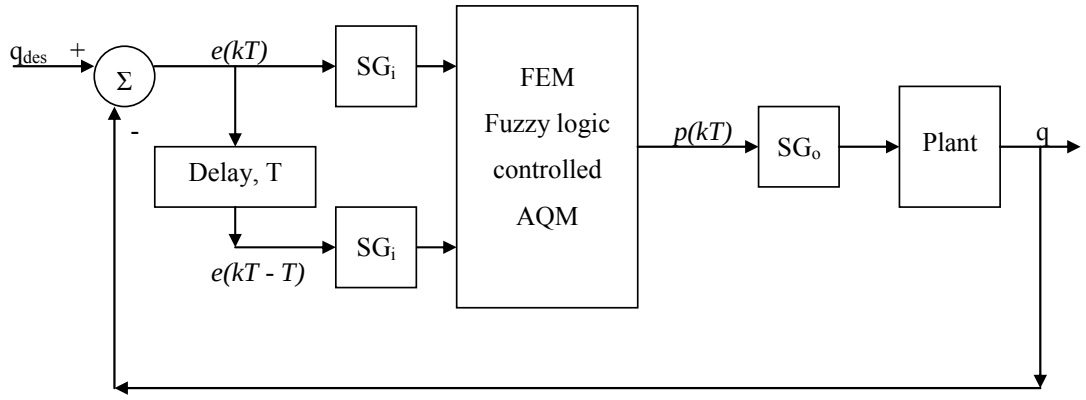


Figure 6.1 Fuzzy logic based AQM system model

only when both sampled queue lengths reach the TQL (i.e. the errors on the queue length go to zero). The errors converging to zero imply that the input rate has been matched to the link capacity, and there is no growth or drain in the router queue level. This has the effect of decoupling the congestion measure from the performance measure by keeping as congestion indices the queue length and the input rate (which is approximated with queue growth rate, as discussed above). Thus, the calculation of the *mark* probability is not *directly* related to the actual queue length.

Further, the output of the controller is clearly a nonlinear *mark* probability that is given as input of the controlled system in order to decide whether to *mark* a particular packet.

After all the inputs and the output are defined for the FEM controller, we can specify the fuzzy control system shown in Figure 6.1, where all quantities are considered at the discrete instant *kT*:

- *T* is the sampling period
- $e(kT)$ is the error on the controlled variable queue length, $q(kT)$, at each sampling period *kT*, defined in Equation 6.1.

$$e(kT) = q_{des} - q(kT) \tag{6.1}$$

where q_{des} is the specified desired TQL.

- $e(kT-T)$ is the error on queue length with a delay T (at the previous sampling period)
- $p(kT)$ is the packet *mark* probability
- SG_i and SG_o are the input and output scaling gains, respectively

In fuzzy control theory, the range of values for a given controller input or output is often called the "universe of discourse". Often, for greater flexibility in fuzzy controller implementation, the universe of discourse for each process input is "normalized" to the interval [-1, +1] by means of constant scaling factors (Passino & Yurkovich, 1998). For our fuzzy controller design, the scaling gains SG_i and SG_o, shown in Figure 6.1, are employed to normalize the universe of discourse for the controller inputs error $e(kT)$ and $e(kT-T)$, and for the controller output $p(kT)$, respectively (e.g., $SG_i * e(kT)$) is an input value to the fuzzy controller). The gain SG_i is chosen so that the range of values of $SG_i * e(kT)$ and $SG_i * e(kT-T)$ lie on [-1, 1], and SG_o is chosen by using the allowed range of inputs to the plant in a similar way. The range of values of the controller's output lies between 0 and 1 (i.e., $p(kT) \in [0,1]$).

In order to achieve a normalized range of the FEM input variables from *-1* to *1*, the input scaling gain SG_i is set to be equal to $-1/(q_{des}-QueueBufferSize)$, if the instantaneous queue length, q_{inst}, is greater than the TQL (q_{des}); otherwise SG_i is equal to $1/q_{des}$ (see Equation 6.2):

$$SG_i = \begin{cases} -\dfrac{1}{q_{des} - BufferSize}, & q_{inst} > q_{des} \\ \dfrac{1}{q_{des}}, & otherwise \end{cases} \tag{6.2}$$

The SG_i values are taken by considering the lower and upper bounds of the queue length. When the instantaneous queue length takes its maximum value (i.e., is equal

to the buffer size), then the error on the queue length (see Equation 6.1) should have its minimum value of q_{des}–*BufferSize.* On the other hand, when the instantaneous queue length takes its minimum value, that is, zero, then the error on the queue length has its maximum value that is equal to q_{des}.

The output scaling gain SG_o is determined so that the range of outputs that is possible is the maximum, as well as to ensure that the input to the plant will not saturate around the maximum. Following the approach of Floyd, Gummadi, and Shenker (2001) SG_o is dynamically set to a value indicating the maximum *mark* probability (e.g. initially set to 10%) in response to changes of the instantaneous queue length, q_{inst}. That is,

$$\begin{aligned} &\textit{if } q_{inst} > 1.1 * TQL \quad \textit{THEN} \quad \textit{increase } SG_o \textit{ by } 0.01 \\ &\textit{if } q_{inst} < 0.9 * TQL \quad \textit{THEN} \quad \textit{decrease } SG_o \textit{ by } 0.9 \end{aligned}$$

The dynamic selection of SG_o based on formal adaptive control theory is a subject of future research.

6.4.2 Control Knowledge - Linguistic Description

To specify rules for the rule base, we need to provide a description of how best to control the plant. We seek to take this "linguistic" description and load it into the fuzzy controller. Hence, linguistic expressions are needed for the inputs and the output, and the characteristics of the inputs and the output. We will use "linguistic variables" (that is, constant symbolic descriptions of what are in general time-varying quantities) to describe fuzzy system inputs and output. For FEM controller,

- "*queue-error*" describes $e(kT)$
- "*previous-queue- error*" describes $e(kT-T)$
- "*mark-probability*" describes $p(kT)$

The linguistic variables take on "linguistic values" that change dynamically over time and are used to describe specific characteristics of the variables. Linguistic

values are generally descriptive terms such as "*positive-big*", "*zero*" and "*negative-small*".

The linguistic variables and values provide us a language to express our ideas about the control decision-making process in the context of the framework established by our choice of FEM controller inputs and output. In order to determine the linguistic values of the input and output variables, we need to define partitions over the input and output space that will adequately represent the linguistic variables. Since the inputs of the FEM controller deals with the queue evolution, which is dynamic and time-varying in nature, we need to have as "many" operating regions – state partitions as possible, in order to capture as much detail of the dynamics and the nonlinearities of the TCP/IP plant. We also need to keep the controller as simple as possible by not increasing the number of linguistic values – state partitions beyond a number, which offers insignificant improvement on the plant performance. The same applies for the output of the FEM controller, the *mark* probability.

Firstly, we need to quantify certain dynamic behaviors with linguistics. For example, each of the following statements quantifies a different configuration of the queue:

- The statement "*queue-error is positive-big*" can represent the situation where, recalling Equation 6.1, the queue is much below the desired TQL.
- The statement "*queue-error is negative-small*" can represent the situation where, recalling Equation 6.1, the queue is driven slightly above the desired TQL, but not too close to the TQL to justify quantifying it as "*zero*", and not too far away to justify quantifying it as "*negative-big*".
- The statement "*queue-error is zero*" can represent the situation where, recalling Equation 6.1, the queue is very closed to the desired TQL, and thus has reached the "equilibrium" state (a linguistic quantification is not precise, thus we are willing to accept any value of the *queue-error* around the TQL as being quantified linguistically by "*zero*" since this can be considered a better quantification than "*positive-small*" or "*negative-small*").

- The statement "*queue-error is negative-big* and *previous-queue-error is negative-small*" can represent the situation where the queue is much above the TQL, and it has the trend of "moving" away, in the "upward" direction, from the specified desired queue length.
- The statement "*queue-error is positive-big* and *previous-queue-error is negative-small*" can represent the situation where the queue is much below the TQL, and it has the trend of "moving" away, in the "downward" direction, from the specified desired queue length.

The model of the FEM control system, comprising the control rules and the values of the linguistic variables, is obtained through an offline intuitive tuning process that starts from a set of the initial insight considerations and progressively modifies the number of linguistic values of the system until it reaches a level of adequate performance. The design objective is to keep the controller as simple as possible to start with, and only increase complexity, by adding more linguistic values, if required. The right number of linguistic values is essential to describe the nonlinear behavior of a system accurately enough. A formal sensitivity analysis to the choice and number of rules is beyond the scope of this thesis, but our experimentation has shown that it is not very sensitive. Adding more rules, as expected, increases the accuracy of the approximation, which yields an improved control performance. But beyond a certain point the improvement is marginal. Guidelines for selecting the rules are discussed in the next section.

The linguistic values chosen for the *queue-error* and *previous-queue-error* input linguistic variables are the following:

- *negative-very-big*
- *negative-big*
- *negative-small*
- *zero*
- *positive-small*

- *positive-big*
- *positive-very-big*

The corresponding linguistic values chosen for the *mark-probability* output linguistic variable are the following:

- *zero*
- *tiny*
- *very-small*
- *small*
- *big*
- *very-big*
- *huge*

6.4.3 Specifying the Knowledge – Rule Base

Using the linguistic quantification described in Section 6.4.2, we next specify a set of rules (i.e., create a rule base) that captures our knowledge about how to control the plant.

By choosing the simplest MISO controller, we have avoided the exponential increase of the rule base, and thus increase the complexity of the controller, when the number of input variables increases.

A careful design of the rule base is done based on two goals:

- *Completeness:* Completeness of rules means that all kinds of situations of system behavior are taken into consideration, i.e., all kinds of combinations of input variables results in an appropriate output value.
- *Consistency:* The rule base is consistent if it does not contain any contradiction. A set of rules is inconsistent if there are at least two rules with the same antecedents-part and different consequent-part.

The knowledge-base for the fuzzy controller is generated from IF-THEN control rules of the form:

IF queue-error is E_i *AND previous-queue-error is* E_j *THEN mark-probability is* P_m

where *queue-error* and *previous-queue-error* denote the *linguistic variables* associated with the two controller inputs, the error on queue length for two consecutive sampling periods ($e(kT)$ and $e(kT-T)$), *mark-probability* denotes the the linguistic variable associated with the controller's output ($p(kT)$), E_b denote the b^{th} *linguistic value* associated with the input linguistic variables, and P_b denotes the b^{th} *consequent linguistic value* associated with the output linguistic variable.

As a generic example consider the case where one fuzzy rule could be

IF queue-error is negative-big AND previous-queue-error is negative-small

THEN mark-probability is big.

A set of such rules forms the "rule base", which characterizes how to control a dynamical system.

The fuzzy control rules are given empirically to determine the control signal according to the errors on the queue length. This relationship between the inputs and the output is mainly based on intuitive understanding and considerations (using expert knowledge) of the concept of congestion control, and tuned manually offline from system behavior observation such as packet *marking*, delay occurrences and throughput curves. For example, if the current error on queue length is *negative-big* and the past error on queue length is *negative-small* then the output control signal should be *big* enough in order that the system can respond quickly to drive the controlled system's output downward closed to the TQL. Another example could be the situation where the current error on queue length is *positive-big* and the past error on queue length is *negative-small* then the output control signal should be *zero* in order to let the queue length "move" upwards towards the TQL. Thus, the design of the rule matrix can be straightforward. The fuzzy rules are easy to understand and highly suitable for representation of the decision-making process a fuzzy-based AQM controller has to possess.

Table 6.1 FEM Linguistic rules – Rule base

p(kT)		**Q_{error} (kT - T)**						
		NVB	NB	NS	Z	PS	PB	PVB
Q_{error} (kT)	NVB	H	H	H	H	H	H	H
	NB	B	B	B	VB	VB	H	H
	NS	T	VS	S	S	B	VB	VB
	Z	Z	Z	Z	T	VS	S	B
	PS	Z	Z	Z	Z	T	T	VS
	PB	Z	Z	Z	Z	Z	Z	T
	PVB	Z	Z	Z	Z	Z	Z	Z

The philosophy behind the knowledge base of the FEM scheme is that of being aggressive when the queue length deviates from the TQL (where congestion starts to set in and quick relief is required), but on the other hand being able to smoothly respond when the queue length is around the TQL. All other rules can represent intermediate situations, thus providing the control mechanism with a highly dynamic action.

Using the above approach, we can continue to write rules for FEM controller for all possible cases. With two inputs and seven linguistic values for each of these, there are at most $7^2 = 49$ possible rules; that is, all possible combinations of antecedent linguistic values for the two inputs. The designed rule base has a compact size that can achieve a fast reasoning. Note that for a given time, not all of the rules are activated (that is, a nonzero activation level exists, when the fuzzy AND operation on the antecedents-part of a rule gives a nonzero value); thus making the rule base simple and fast. This is discussed in the next section.

The complete set of rules that define the *knowledge base* of the FEM controller is given in Appendix A. A convenient way to list all possible rules is to use a tabular representation (see Table 6.1[*]). These rules reflect the particular view and experiences of the designer, and are easy to relate to human reasoning processes and gathered experiences.

[*] Table content notations: negative/positive very big (NVB/PVB), negative/positive big (NB/PB), negative/positive small (NS/PS), zero (Z), huge (H), very big (VB), big (B), small (S), very small (VS), tiny (T).

Note that the actual number of rules is reduced, since when the current error on queue length is *negative-very-big*, then the output control signal is always *huge*, irrespective of the status of the past error on queue length. The same applies when the current error on queue length is *positive-very-big*, then the output control signal is always *zero*.

6.4.4 Fuzzy Quantification of Knowledge – Inference Process

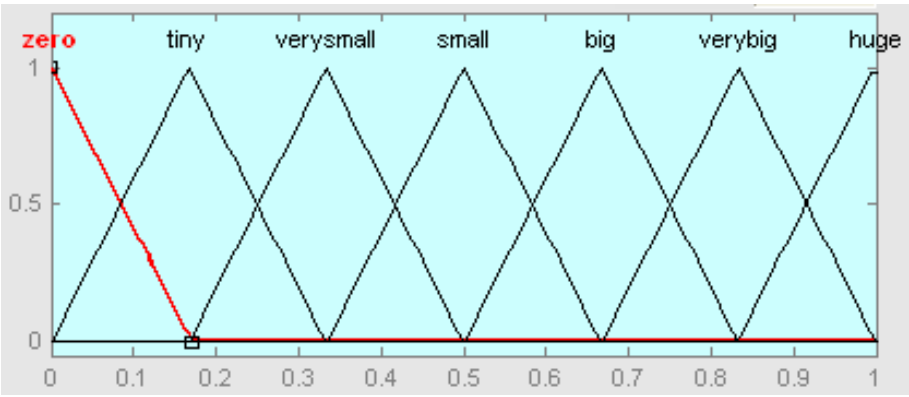
The FEM controller is a Mamdani-based model. Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology (Mamdani & Assilian, 1975). We also adopt this approach due to its simplicity and investigate its suitability through extensive simulations.

6.4.4.1 Selected Membership Functions

Firstly, we need to quantify the meaning of the linguistic values using membership functions. The membership functions of the linguistic variables are determined (*deep structure*), and are the result of a more intuitive and pragmatic choice and not of an analytic approach (that this works – see Chapter 7 – is one of the main advantages of fuzzy logic controllers compared to the conventional counterparts). Due to computational simplicity the membership function of a linguistic variable is often triangular or trapezoidal shaped, thus they are used in our FEM control model. These types of shapes are a standard choice used in many industry applications due to their simple expressions. The chosen membership functions representing the linguistic values for both the inputs and the output of the FEM controller are shown in Figure 6.2.

The amount of *overlapping* between the membership functions' areas is significant. The left and right half of the triangle membership functions for each linguistic value is chosen to provide membership overlap with adjacent membership functions. Our method is simple in that we use *symmetric-and-equally spaced* membership functions, where the sum of the grade of membership of an input value,

(a) linguistic input variables



(b) linguistic output variable

Figure 6.2 Membership functions of the linguistic values representing the input variables "normalized error on queue length for two consecutive sample periods", and the output variable "mark probability"

concerning the linguistic values of a specific input variable, is always one (see Equation 6.3).

$$\sum_{k=1}^{m} \mu_k(x_i) = 1 \tag{6.3}$$

where $\mu_k(x_i)$ is the membership value of the input value x_i taken from the membership function of the linguistic value k, ($1 < k < m$, where m is the number of linguistic values of a linguistic variable), of the input variable of concern.

The overlapping of the fuzzy regions, representing the continuous domain of each control variable, contributes to a well-behaved and predictable system operation; thus the fuzzy system can be very robust.

If there were no overlap, no more than one rule could be activated at the same time; thus no adequate inference applies. Further, we remark that the choice of equal-width intervals entails no loss of generality (Chen, Pham, & Weiss, 1995).

The membership functions at the outer edges in Figure 6.2 deserve special attention. For the input variables we see that the outermost membership functions "saturate" at a value of one (thus, the use of a trapezoidal-like shapes). This makes intuitive sense as at some point we just group, for example, all large values together in a linguistic description such as "*positive-very-big*". The membership functions at the outermost edges appropriately characterize this phenomenon since they characterize "greater than" (for the right side) and "less than" (for the left side) situations.

For the output variable, the membership functions at the outermost edges cannot be saturated for the FEM controller to be properly defined. The basic reason for this is that in fuzzy-based decision-making processes we seek to take actions that specify an *exact value* for the controlled system's input.

As the membership functions are designed to be overlapping-symmetric-equally spaced, this results in having at most two membership functions overlapping, thus we will never have more than four rules on/activated at a given time. This offers computational simplicity on the implementation of the FEM controller, a design objective.

6.4.4.2 Implication-Aggregation-Defuzzification

After the linguistic rules are set (called *surface structure*), and the membership functions of the linguistic values are determined (called *deep structure*), we need to further define the following: the fuzzy AND operator used in the antecedents part of the rule base is chosen to be the *min* (minimum) operation, which indicates that *we can be no more certain about the conjunction of the two statements* (that belong to the IF-part of each rule) *than we are about the individual terms that make them up.*

For the implication of the antecedents-part to the consequent-part of each rule, we use the *min* (minimum) operation that *truncates* the output fuzzy set. The justification of using the *minimum* operator to represent the implication is that *we can be no more certain about our consequent than our antecedent*.

Further, the *max* (maximum) operation to all implied output fuzzy sets is used in the aggregation process. The resulted fuzzy set contains the maximum membership values among those generated by the implication process. This single aggregated output fuzzy set needs however to be transformed into a single output crisp value, using a defuzzyfication method.

The calculated output control signal of the nonlinear fuzzy controller, shown in Equation 6.4, uses the center of gravity – the most common defuzzification method - (Passino & Yurkovich, 1998) or centroid of area of the aggregated fuzzy output set *C:*

$$p_k = \frac{\int y\mu_C(y)dy}{\int \mu_C(y)dy} \tag{6.4}$$

where, $\mu_C(y) = \max\left(\mu_1(y), \mu_2(y), ..., \mu_N(y)\right)$ is the membership degree of y in the aggregated fuzzy set C (which is found using the *max-operation* over all N implicated output fuzzy sets), and N is the number of linguistic rules.

The limits of integration correspond to the entire universe of discourse Y of output *mark* probability values, to which y belongs. To reduce computations, we discretize the output universe of discourse Y to m values, $Y = \{y_1, y_2, ..., y_m\}$, which gives the discrete fuzzy centroid (see Equation 6.5).

$$p_k = \frac{\sum_{j=1}^{m} y_j \times \mu_C(y_j)}{\sum_{j=1}^{m} \mu_C(y_j)} \tag{6.5}$$

Note that the use of symmetric triangular and trapezoidal membership functions makes the computation of the equation easy.

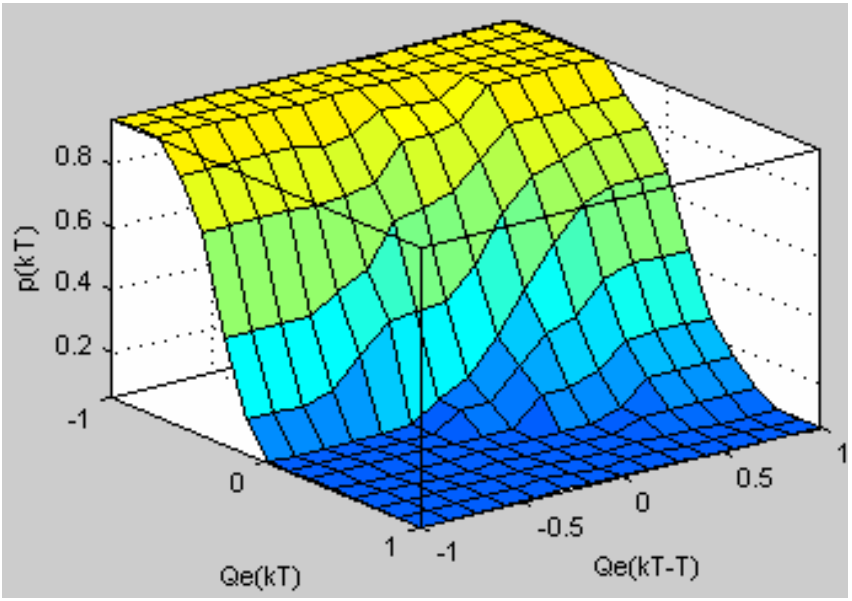


Figure 6.3 Control-decision surface of the fuzzy inference engine of FEM controller. The nonlinear control surface is shaped by the rule base and the linguistic values of the linguistic variables.

6.4.4.3 FEM Nonlinear Control Surface

The nonlinearity that is implemented by the FEM controller, called the *control-decision surface*, is shaped by the constructed rule base and the linguistic values of the inputs and output variables (see Figure 6.3). This surface represents in a compact way all the information in the fuzzy controller. It is useful to notice that there is a type of interpolation between the rules that is performed by the FEM controller that is illustrated in Figure 6.3. The output is actually an interpolation of the effects of the four rules that are mostly on/activated. For a certain queue length, different *mark* probabilities are calculated depending on the past error on queue length. An inspection of this nonlinear control surface and the linguistic rules shown in Table 6.1 provides hints on the operation of FEM. The *mark* probability behaviour under the region of equilibrium (i.e., where the error on the queue length is close to zero) is smoothly calculated. On the other hand, the rules are aggressive by increasing the probability of packet *marking* sharply in the region beyond the equilibrium point (and where congestion starts to set in and quick relief is required). Thus the inference process of FEM controller dynamically calculates the *mark* probability

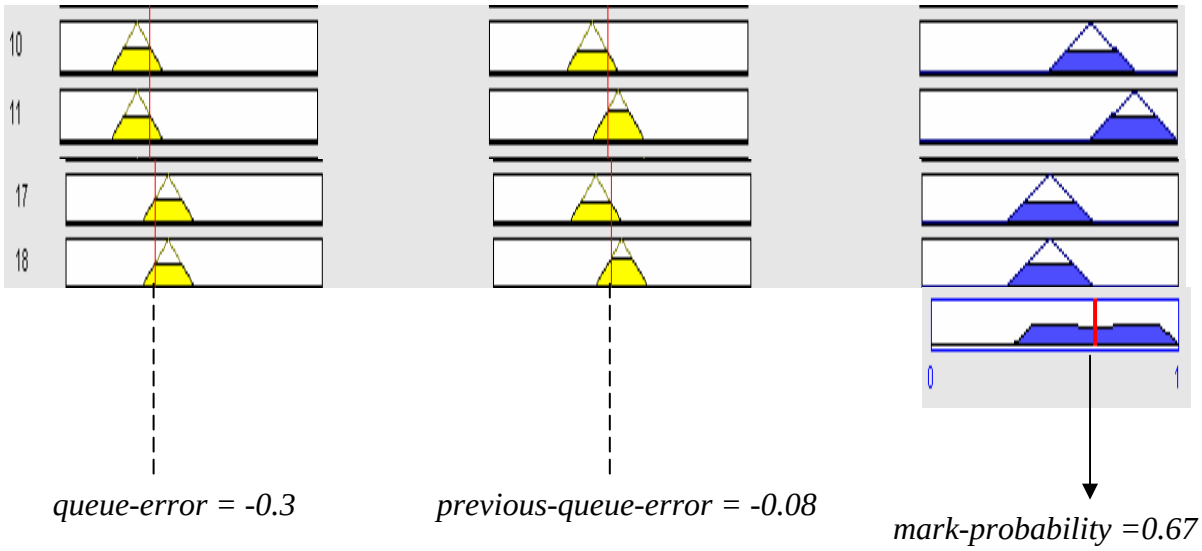


Figure 6.4 Example of computing FEM output

behaviour based on the two inputs. The dynamic way of calculating the *mark* probability by the inference process comes from the fact that according to the error of queue length for *two* consecutive sample periods, a different set of fuzzy rules and so inference apply. Based on these rules and inferences, the *mark* probability is more responsive than other AQM approaches, (as for e.g. Floyd & Jacobson, 1993; Floyd, Gummadi, & Shenker, 2001; Hollot, Misra, Towsley, & Gong, 2002; Athuraliya, Li, Low, & Yin, 2001; and Kunniyur & Srikant, 2004) due to the human reasoning and the inbuilt non linearity.

6.5 Illustrative Example of Computing Controller Output

An illustrative example on the operation of the proposed fuzzy logic based control methodology is given in Figure 6.4. Assume that at the end of two successive sample intervals (past and current) the normalised errors on the queue length are calculated as *-0.08* and *-0.3*, respectively; that is there is a trend of the queue "moving" away, in the "upward" direction, from the specified desired queue length.

Note that the current error is a member of the fuzzy sets "*negative-big*" and "*negative-small*" with equal membership values of 0.5 each. On the other hand, the

past error is a member of the fuzzy sets "*negative-small*" and "*zero*" with membership values of 0.4 and 0.6, respectively. Their membership of any other fuzzy sets is equal to zero. Each rule is visited and the minimum of membership values of the inputs to the corresponding linguistic values are found. Then the corresponding (for each rule) output linguistic value is truncated (using the *min* operation) accordingly. That is, with the numerical values of input variables used in this example, only rules 10, 11, 17, and 18 of Appendix A contribute in the calculation of the output. For further clarification the above mentioned rules are shown below:

10. *IF queue-error is negative-big and the previous-queue-error is negative-small THEN mark-probability is big*

11. *IF queue-error is negative-big and the previous-queue-error is zero THEN mark-probability is very-big*

17. *IF queue-error is negative-small and the previous-queue-error is negative-small THEN mark-probability is small*

18. *IF queue-error is negative-small and the previous-queue-error is zero THEN mark-probability is small*

In rules 10 and 17, the output fuzzy sets of "*big*" and "*small*", respectively, are truncated by the antecedents' minimum membership value of 0.4, whereas in rules 11 and 18 the output fuzzy sets of "*very-big*" and "*small*", respectively, are truncated by the antecedents' minimum membership value of 0.5.

We then take the aggregated of all four implied fuzzy sets using the *max* operator and by using the center-of-gravity defuzzification method, a numerical value for the output of the controller is computed as 0.67, which is considered as rather "big" *mark* probability, in order that the system can respond quickly to drive the queue length downward closed to the TQL.

6.6 Sensitivity of Fuzzy Logic Control Methodology to External Parameters Settings

The design of the fuzzy logic control methodology, as explained in the previous sections is based on intuitive understanding and considerations (using expert knowledge) of the concept of congestion control, and tuned manually offline from system behavior observation. At the same time it is kept as simple and generic as possible.

Since the fuzzy controller is nonlinear, it is very difficult to examine analytically the influence of certain parameters. The general properties of FLC are widely discussed in the literature. We rather investigate the effects that the external parameters (TQL, sampling interval, and scaling gain) of the fuzzy controller have on the controlled system's behavior.

Let us recall what the significance of these parameters is:

- The desired-target queue length (TQL), is a constant reference input that can be set by the network operator at a level that can achieve bounded mean queuing delay and variation.
- The sampling interval (T) is used to periodically compute a new/updated *mark* probability, that is, a new input for the controlled system, in order to adequately respond to possible changes in the behavior of the controlled system. In general, the smaller the sampling interval, the better. It means that the feedback signal arriving at the hosts is more up-to-date. On the other hand, a small sampling interval means more processing overhead.
- The input scaling gain (SG_i) values are taken by considering the lower and upper bounds of the queue length, in order to achieve a normalized range of the input controller's variables from *-1* to *1*. Thus, it is a constant factor (see Equation 6.2) that can not be changed over any network/traffic conditions. The output scaling gain (SG_o) is determined so that the range of outputs that is possible is the maximum. Following the approach of Floyd, Gummadi, and Shenker (2001) SG_o

is dynamically set to a value indicating the maximum *mark* probability (e.g. initially set to 10%).

To investigate the sensitivity of the fuzzy logic control methodology to the above parameters we use the case of *a single-bottleneck* network topology – shown in Figure 7.1 (see Chapter 7), with the following setup. We set the link capacities of the sources to the first-hop router (C_1, d_1) to be 100 Mbps. The number of TCP active flows used is 100 long-lived File Transfer Protocol (FTP) flows, and 100 short-lived TCP flows (Web-like flows). The senders are grouped equally into 4 groups, with each group having a propagation delay increased by *5 msec*, starting from *5 msec* up to *35 msec* (that is, we create heterogeneous delays in the network). The bottleneck link capacity is *15 Mbit/s* with a propagation delay of either *30 msec* or *120 msec* (C_2, d_2). The last link to the destination (C_3, d_3) is set to (*200 Mbps, 5 msec*). We also provide some time-varying dynamics by stopping half of the TCP/FTP flows at time $t = 70$ *sec* and resuming transmission at time $t = 150$ *sec* (the whole simulation time is set to *200 sec*), in order to examine the effects of decreasing/increasing the number of flows.

We have conducted two kinds of experiments: one with a bottleneck link propagation delay of *30 msec* and another with a bottleneck link propagation delay of *120 msec*; thus we further investigate the effect of delays. For both kinds of experiments, we vary one of the three parameters under study, and keep the other two fixed.

In particular, we vary the TQL for 10%, 20%, 40% and 60% of the buffer size (we don't examine larger values of TQL because this will violate our goal of having low mean delay). For a buffer size of 500 packets (and assuming a packet of 1000 bytes), the TQL is varied starting from 50, 100, 200, and 300 packets. For a fixed value TQL is chosen to be 40% of the buffer size, that is, 200 packets.

The sampling interval, T, is at first fixed at *6 msec*, which corresponds to a sampling frequency of about *160 Hz* (we keep the value selected by Hollot, Misra, Towsley, & Gong, 2002, for convenient comparison). Also, this choice is further confirmed by Athuraliya (2002) that indicates that a value of the sampling interval

between *1 msec* and *10* msec seems to work well for REM controller (Athuraliya, Li, Low, & Yin, 2001). This sampling interval is by far large enough to complete a fuzzy inference cycle, and small enough to respond to fluctuations of incoming flows. Note that Dualibe, Jespers, and Verleysen (2000) report on a programmable analogue fuzzy controller implemented in CMOS technology, capable of performing 5.26 million fuzzy inferences per second. Thus, we vary the sampling period for values *1 msec, 6 msec,* and *10 msec*.

The output scaling gain SG_o, following the approach of Floyd, Gummadi, and Shenker (2001), is initially set to a value of *0.1* (10%) that indicates the maximum *mark* probability. We vary the SG_o with initial values of *0.02, 0.1,* and *0.5*.

For both kinds of experiments (i.e., bottleneck link propagation delay of *30 msec* and *120 msec*), we show the instantaneous queue length evolution with respect to time (see Figure 6.5, 6.6 & 6.7, where we show the sensitivity of Fuzzy Logic Control Methodology to the setting of external parameters of TQL, sampling interval, and SG_o, respectively). We further obtain the main statistical results of mean queuing delay and its standard deviation, the utilization of the bottleneck link (regarding the useful throughput), and the loss rate (see Table 6.2, 6.3 & 6.4, for sensitivity of Fuzzy Logic Control Methodology to external parameters of TQL, sampling interval, and SG_o, respectively).

The results obtained show that the proposed methodology is not very sensitive to the external parameters of the target queue length, the sampling interval, and the output scaling gain over the tested scenarios. There is a very small variation on the various statistical results, which do not influence the overall behaviour of the controlled system. Based on these results, the proposed methodology is shown to be acceptably robust against different parameters, rendering it tolerant to a wide range in parameter selection. Of course, looking at Table 6.4, we see an increase of the loss rate for both kinds of experiments, when the SG_o is initially set to 0.02. Even though the loss ratio is small in general, the increase we observe, as opposed to much lower ratios for other SG_o initial values used, indicates that a further investigation of the dynamic tuning of the output scaling gain, based on formal adaptive control theory,

Table 6.2 Summary of statistical results –
Sensitivity of Fuzzy Logic Control Methodology to External Parameter of TQL

T=0.006 sec, SG_0=0.1,
Bottleneck propagation delay (Bpd)=30 or 120 msec
TQL=50: expected mean delay = 26.67 msec
TQL=100: expected mean delay = 53.33 msec
TQL=200: expected mean delay = 106.67 msec
TQL=300: expected mean delay = 160 msec

	Mean Delay (msec)		**Std-Deviation (msec)**		**Utilization (%)**		**Loss Rate (%)**	
	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms
TQL=50	29.92	24.83	9.45	12.69	99.9	99.23	0.0197	0.004
TQL=100	55.11	50.92	9.86	14.03	99.92	99.47	0.017	0.007
TQL=200	105.96	100.62	12.26	21.83	99.92	99.46	0.0186	0.0075
TQL=300	159.53	151.27	14.68	27.23	99.92	99.58	0.012	0.005

can improve the responsiveness and the accuracy of control. This is recommended as a topic for future research.

Table 6.3 Summary of statistical results –
Sensitivity of Fuzzy Logic Control Methodology to
External Parameter of Sampling Interval

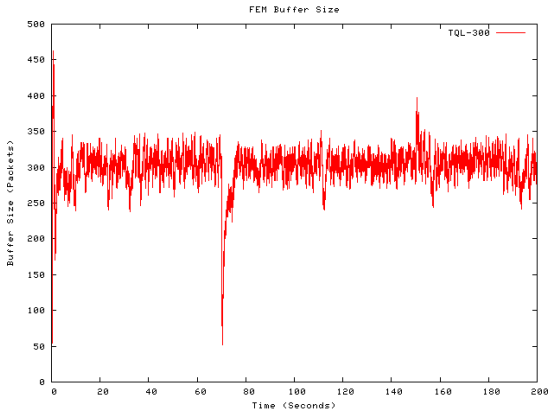
TQL=200: expected mean delay = 106.67 msec, SG_0=0.1,
Bottleneck propagation delay (Bpd)=30 or 120 msec

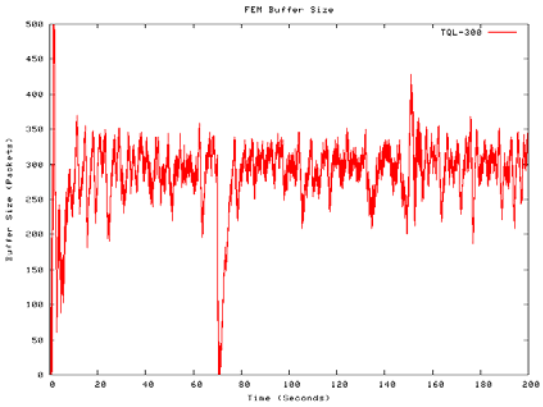
	Mean Delay (ms)		**Std-Deviation (ms)**		**Utilization (%)**		**Loss Rate (%)**	
	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms
T=0.001 sec	109.22	103.84	15.93	21.8	99.86	99.46	0.0697	0.14
T=0.006 sec	105.96	100.62	12.26	21.83	99.92	99.46	0.0186	0.0075
T=0.01 sec	105.19	100.41	13.36	19.21	99.92	99.54	0.017	0.009

Table 6.4 Summary of statistical results –
Sensitivity of Fuzzy Logic Control Methodology to
External Parameter of Output Scaling Gain

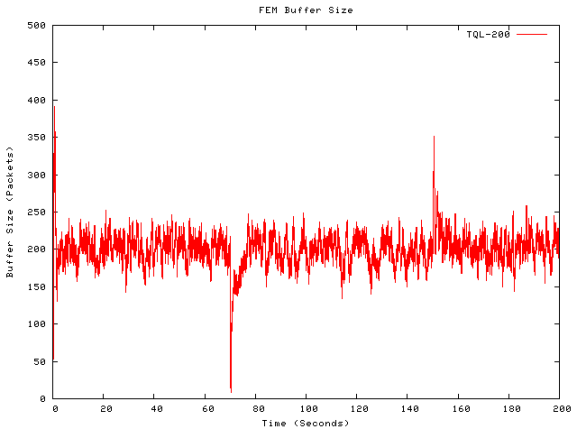
TQL=200: expected mean delay = 106.67 msec, T=0.006 sec,
Bottleneck propagation delay (Bpd)=30 or 120 msec

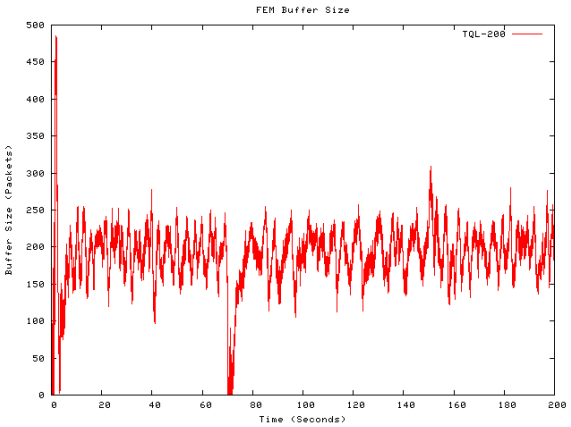
	Mean Delay (ms)		**Std-Deviation (ms)**		**Utilization (%)**		**Loss Rate (%)**	
	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms	Bpd= 30ms	Bpd= 120ms
SG_0=0.02	106.84	102.38	17.07	23.29	99.78	99.36	0.107	0.16
SG_0=0.1	105.96	100.62	12.26	21.83	99.92	99.46	0.0186	0.0075
SG_0=0.5	104.27	97.99	15.97	24.75	99.92	99.06	0.0245	0.02



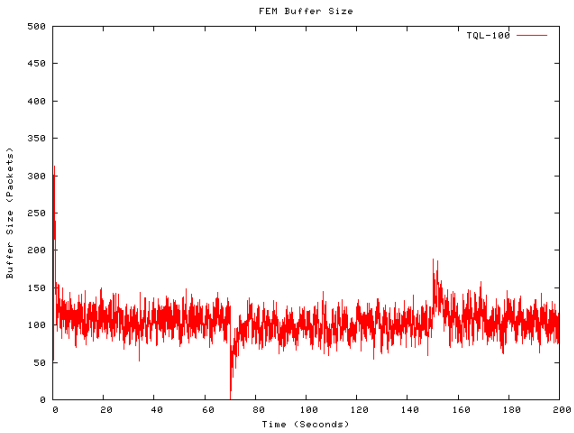


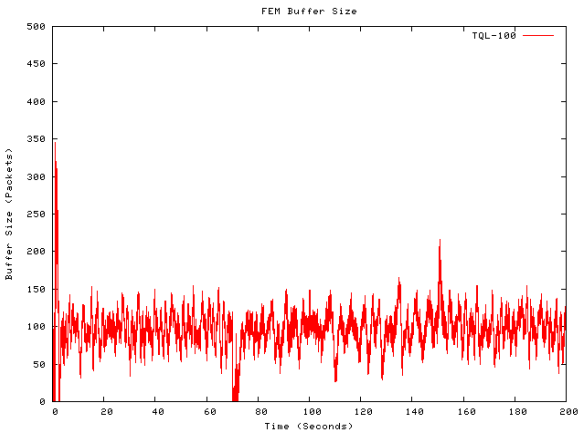
TQL = 300 (bottleneck propagation delay: 30 and 120 msec, respectively)



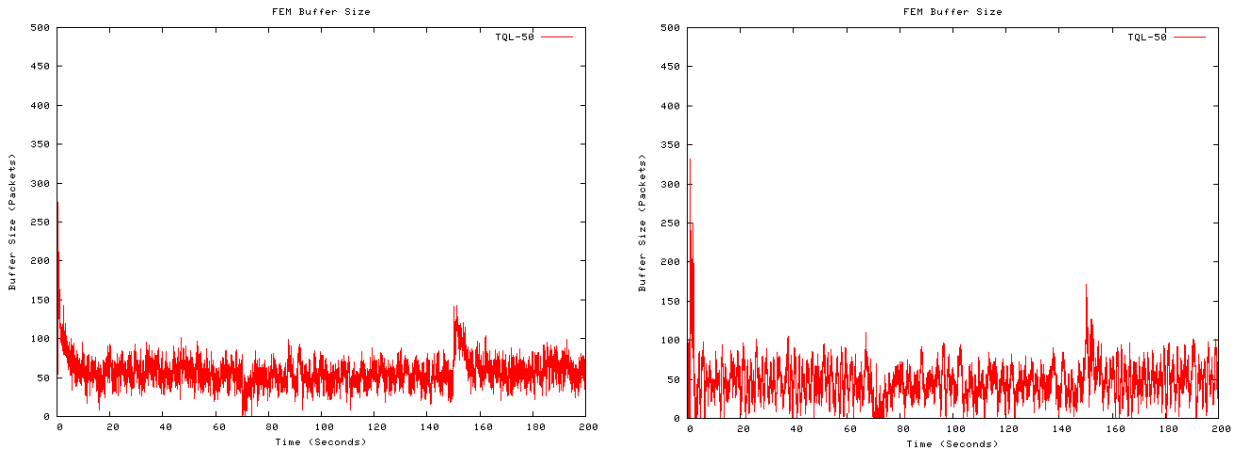


TQL = 200 (bottleneck propagation delay: 30 and 120 msec, respectively)



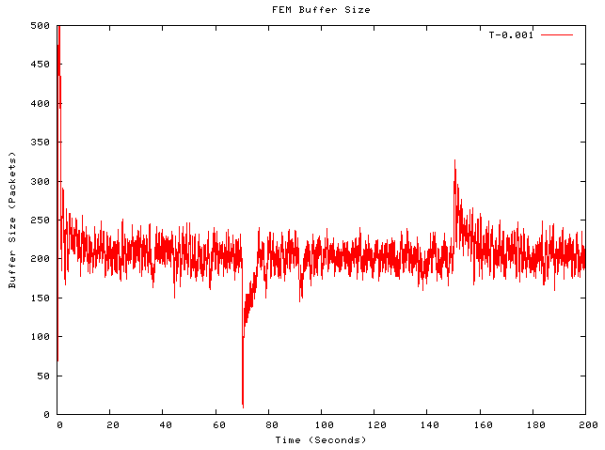


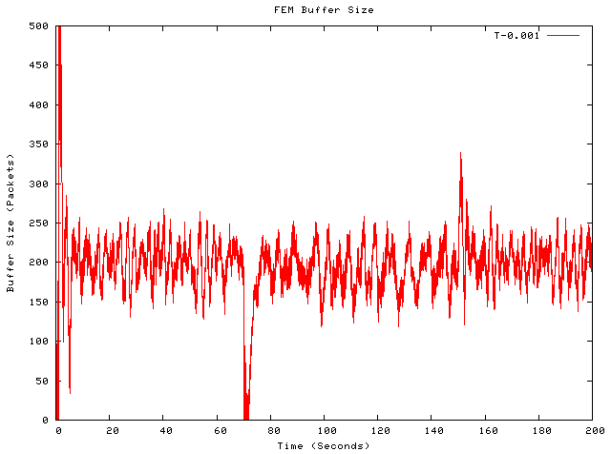
TQL = 100 (bottleneck propagation delay: 30 and 120 msec, respectively)



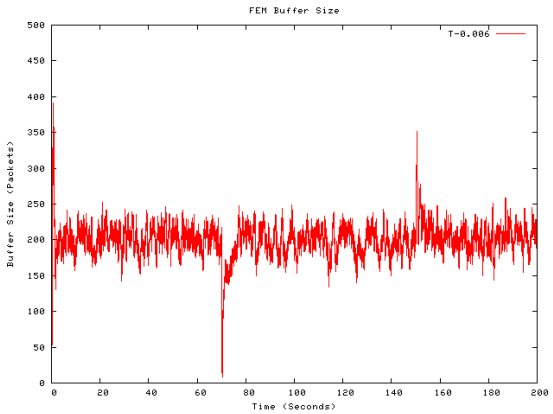
TQL = 50 (bottleneck propagation delay: 30 and 120 msec, respectively)

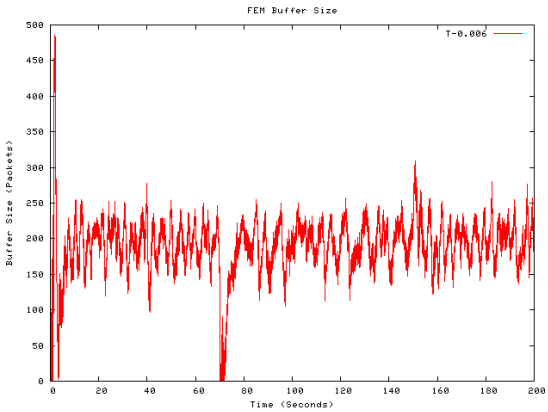
Figure 6.5 Sensitivity of Fuzzy Logic Control Methodology to External Parameter of TQL



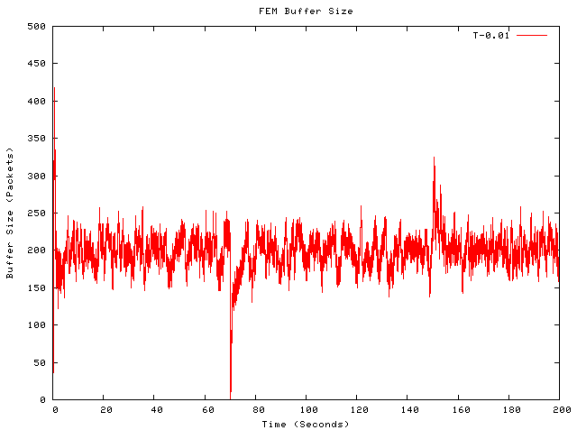


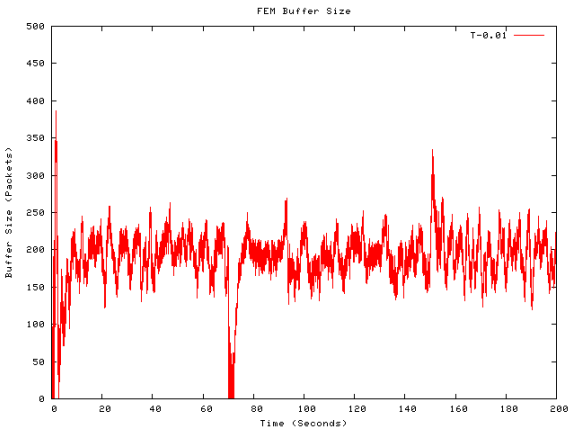
T = 0.001 sec (bottleneck propagation delay: 30 and 120 msec, respectively)





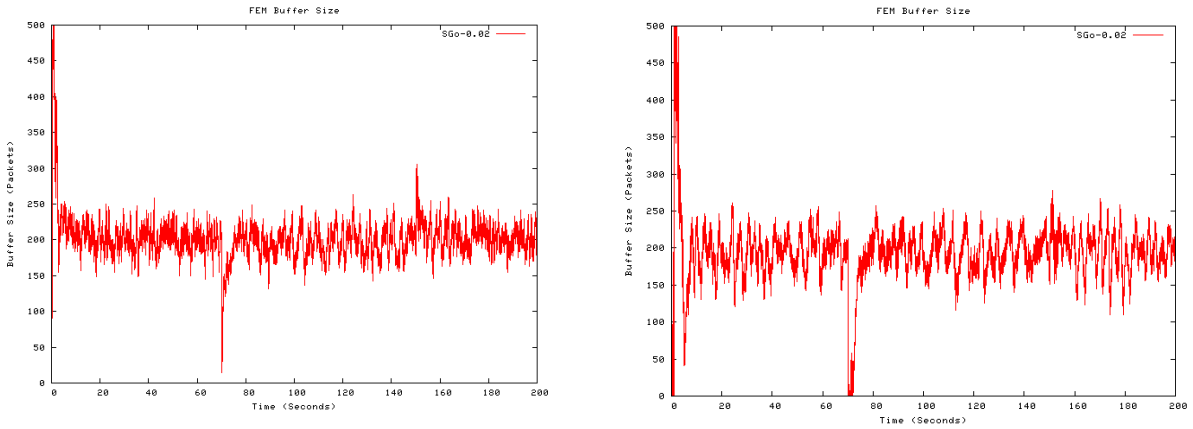
T = 0.006 sec (bottleneck propagation delay: 30 and 120 msec, respectively)



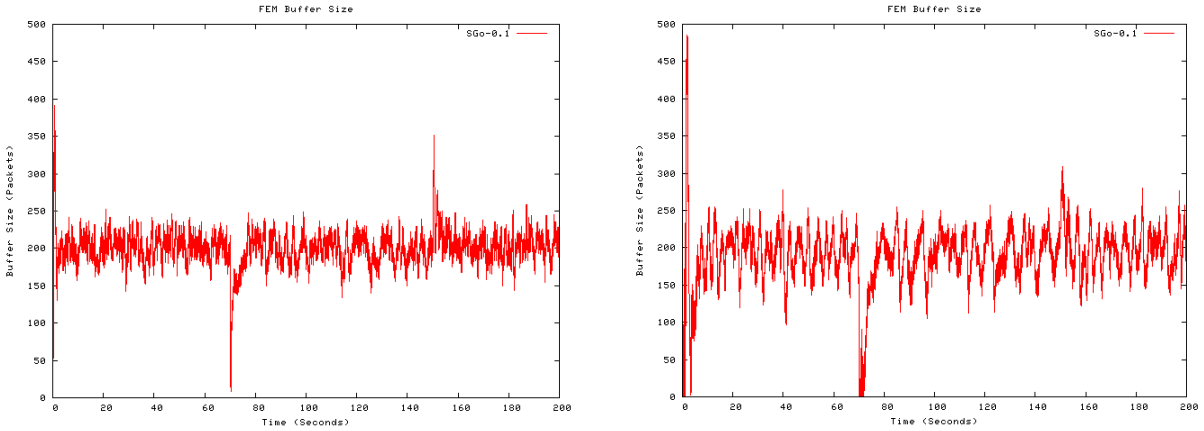


T = 0.01 sec (bottleneck propagation delay: 30 and 120 msec, respectively)

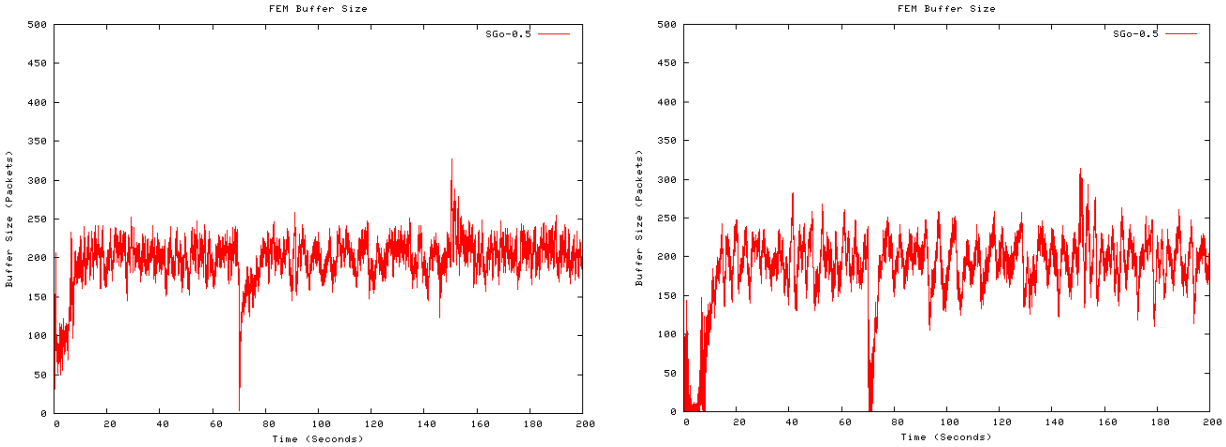
Figure 6.6 Sensitivity of Fuzzy Logic Control Methodology to External Parameter of Sampling Interval



SG_o = 0.02 (bottleneck propagation delay: 30 and 120 msec, respectively)



SG_o = 0.1 (bottleneck propagation delay: 30 and 120 msec, respectively)



SG_o = 0.5 (bottleneck propagation delay: 30 and 120 msec, respectively)

Figure 6.7 Sensitivity of Fuzzy Logic Control Methodology to External Parameter of Output Scaling Gain

6.7 Practicability of Fuzzy Logic Control Methodology

The fuzzy logic based control methodology is proposed having in mind the simplicity of the various parts that constitute the chosen fuzzy system structure. We can assert that this aim is achieved by the choice of the triangular-shaped membership functions, the careful design of overlapping-symmetric-and-equally shaped membership functions, the relatively small number of linguistic rules that our rule base consists of and the min-max inference process that is selected.

Of course, if severe implementation constraints are present in the implementation, there are a number of code optimisations with respect to memory and computation time one can adopt. This is beyond the scope of this thesis.

6.7.1 Computation Time

Our methodology defines a rule base of 49 rules. However, due to the specific design of our membership functions, at most two membership functions overlap at any one point, thus we have at most 4 rules that can be activated at any time. This significant property of the implementation methodology we selected can be used to smartly modify the implementation code so that it will compute only at most four values for the antecedent membership functions, only at most four values for areas of implied fuzzy sets, and hence have only at most four additions in the numerator and denominator of the centroid computation. This procedure reduces the number of required computations significantly.

6.7.2 Memory Requirements

In a real router there is no need for a fuzzy inference engine in the FLC controller. After the linguistic rules have been found and the linguistic values are tuned by a simulator, the control surface is known and can be stored as lookup table for selected sampling points requiring only a few kilobytes of read-only-memory (ROM) in a FEM-capable router. In combination with a simple interpolation algorithm FEM can be implemented in such a way with a very fast response time.

6.7.3 Ease of Implementation

The fuzzy logic based control methodology designed for best-effort networks can be easily adopted in different TCP/IP network environments as well due to the simplicity and generality of the control design of FEM controller. Thus the simplicity of the controller implementation is carried out to the differentiated services network, where only one controller is implemented and called by all differentiated services classes (see Chapter 8 later).

6.8 Conclusions

A nonlinear fuzzy controller is designed, as an alternative AQM mechanism, to supplement the standard TCP to obtain satisfactory performance, in terms of link utilization, delays, and losses. The proposed methodology does not require knowledge of dynamic system/network mathematical model parameters. Rather it relies on an intuitive linguistic model, which results in a nonlinear control law that drives quickly the system to be controlled into the steady-state. By capturing – in an easy way to interpret – the dynamics of the system the fuzzy controller deals with the uncertainties and the high variability appearing in the network and exhibits fast system response and robust behavior in spite of varying network conditions. A comprehensive evaluation and comparison with popular schemes will be presented in Chapter 7.

The fuzzy logic control methodology is expressed in an intuitive form using linguistic variables and semantic rules. The selection of the number of membership functions and their values is based on process knowledge and intuitive understanding and considerations of the concept of congestion control, and tuned manually offline from system behavior observation. At the same time we kept it as simple and generic as possible, by choosing the triangular-shaped membership functions, the careful designing of overlapping-symmetric-and-equally shaped membership functions, selecting a relatively small number of linguistic rules that our rule base consists of, combined with the min-max inference process.

Further, the proposed methodology is shown over a number of tested scenarios not to be very sensitive to the external parameters of the target queue length, the sampling interval, and the output scaling gain. Based on these results, the proposed methodology is shown to be reasonably robust against different parameters, rendering it tolerant to a wide range in parameter selection.

The design of the proposed fuzzy logic based congestion control system allows the use of linguistic knowledge to capture the dynamics of nonlinear probability marking functions, and uses multiple inputs to capture the dynamic state of the network more accurately. It aims to generally provide effective congestion control, and thus better utilization of the network, with lower losses and bounded delays than other AQM schemes, as for example in Floyd and Jacobson (1993), Floyd, Gummadi, and Shenker (2001), Hollot, Misra, Towsley, and Gong (2002), Athuraliya, Li, Low, and Yin (2001), and Kunniyur and Srikant (2004). The design of such a generic control methodology can be easily adopted in different TCP/IP network environments.

We have not attempted to optimally tune our fuzzy controller because, on one hand, this appears to be very demanding due to the many degrees of freedom associated with the membership functions, the rule base, and the parameters thereof, and on the other hand, any further tuning beyond the basic intuitive ideas is not necessary and the fuzzy controller performs adequately, as demonstrated in Chapter 7. Thus, keeping the fuzzy inference process as is, is quite acceptable; however, adaptively tuning of the output scaling gain, using formal adaptive control theory, to investigate whether the tradeoff between increased complexity and improved performance is worthwhile, can be a subject of future research.

Chapter 7

Performance Evaluation of Fuzzy Explicit Marking in TCP/IP Best-effort Networks

7.1 Introduction

In this chapter we use simulative evaluation to demonstrate the effectiveness and robustness of the AQM-based nonlinear fuzzy logic control methodology implemented in TCP/IP best-effort networks, namely *Fuzzy Explicit Marking* (FEM). Experiments with a wide range of network/traffic environments are conducted to compare FEM with other published results by taking some representative, well-known AQM schemes, namely A-RED (Floyd, Gummadi, & Shenker, 2001), PI (Hollot, Misra, Towsley, & Gong, 2002), REM (Athuraliya, Li, Low, & Yin, 2001), and AVQ (Kunniyur & Srikant, 2004).

7.2 Selection of Simulation Parameters

In order to evaluate the performance of the AQM schemes under comparison, we need to define an appropriate simulation environment and parameters that are of major significance in the investigation of the effectiveness and robustness.

We need to carefully choose the right network topologies and parameters to be examined, so as to be as close to the dynamics of the TCP/IP system as possible.

Furthermore, for comparative evaluations and commonly accepted stress results it is necessary to have a common simulative framework. However, this is not an easy task. There are as yet *no standard* topologies and performance indices. One may refer to the current ongoing discussion between the active members of the networking community, which aims to give recommendations/guidelines for a right evaluation of congestion control algorithms in terms of test topologies, and measurements (TMRG, 2006).

7.2.1 Simulation Environment

The Network Simulator NS-2 is the most widely used simulator for simulating advanced TCP/IP algorithms and protocols. It can expand other network modules flexibly and has trusted simulation results. Thus, in our study we adopted NS-2 for the simulative evaluation of the fuzzy logic control methodology.

7.2.2 Simulation Topologies – Network/Traffic Parameters

To stress our algorithm in as realistic scenarios as practical, we use both single- and multiple-congested (tandem) links (bottlenecks) network environments, and through a wide range of experiments we examine the effects, on the AQM schemes, of the following:

- dynamic traffic changes – time-varying dynamics
- traffic load factor
- heterogeneous propagation delays
- different propagation delays at bottleneck links
- different link capacities
- introduction of noise-disturbance (background traffic) to the network (e.g. short-lived TCP connections)
- introduction of reverse-path traffic
- different types of data streams, like TCP/FTP and TCP/Web-like traffic, as well as unresponsive traffic (UDP-like).

By examining the above effects on the selected AQM schemes we intend to get a clear view on the effectiveness and robustness of the algorithms under comparison. The choice of these network/traffic parameters, in our opinion, can lead to an acceptably close representation of TCP/IP's system dynamics.

7.2.3 Simulation Performance Indices

The performance metrics that we use for evaluating the performance of the fuzzy control methodology and the other selected AQM schemes are:

- Bottleneck link utilization (based on the useful throughput that is commonly called *goodput*)
- Loss rate
- Mean queuing delay and its standard deviation.

The selection of the above metrics as performance indices is common. These metrics indicate both network utilization and QoS. As such they are prominent design goals of an AQM scheme. By observing the queuing delay and the amount of delay variation, we can also evaluate indirectly the stabilization performance of the queue. The link utilization is defined as the useful throughput normalized by the link capacity, and the loss rate is defined as the ratio of the number of lost packets to the number of packets transmitted by the source.

7.2.4 AQM Control Parameters

The control parameter values of the AQM schemes under evaluation are set based on the recommendations each of the AQM scheme's authors give, unless otherwise specified (see Table B.1). The objective is to allow a fair comparison using the settings recommended by the respective authors and also allow for a comparative evaluation using the same controller objectives, as for example the queue length reference value.

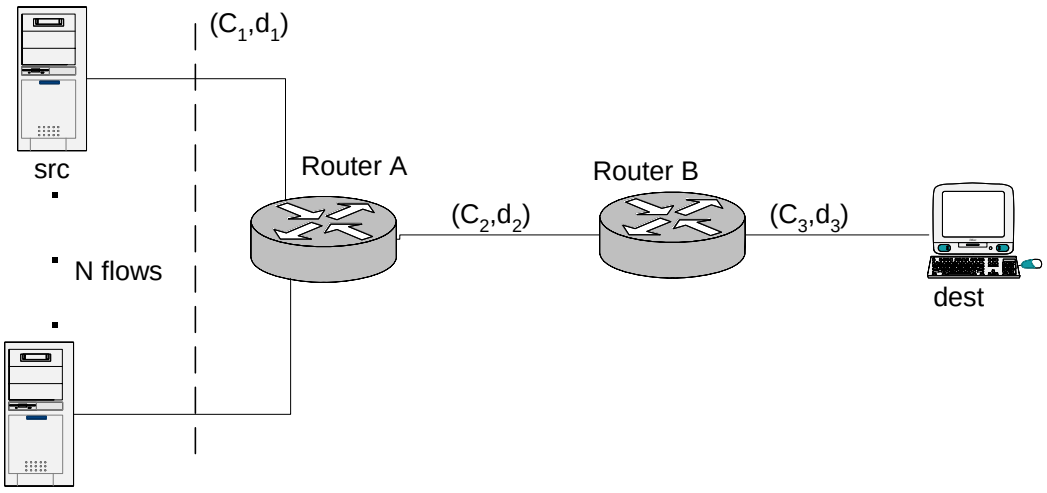


Figure 7.1 Single-bottleneck network topology I

7.3 Single-bottleneck Link

The network topology of a single-bottleneck link is shown in Figure 7.1. We use TCP/Newreno. The buffer size of all queues is set to 500 packets (1000 bytes each). The sampling interval for FEM controller is set to *0,006 sec* (this allows direct comparison with the scheme proposed by Hollot, Misra, Towsley, and Gong, (2002)), and the TQL for FEM, PI, and REM controllers is set to 40% of the buffer size, that is, 200 packets (which is also adopted by Hollot, Misra, Towsley, and Gong, (2002)). For A-RED, we set the minimum threshold to 20% of buffer size (i.e., 100 packets) and the maximum to 60% of buffer size (i.e., 300 packets), giving an average TQL of 40% of buffer size, that is 200 packets, which is in line with the TQL setting of the other algorithms). All nodes and algorithms are ECN-enabled. The simulation time is 100 sec.

To allow fairness in the comparative evaluation, for the various network parameters (e.g., range of RTTs, link capacities, etc) we again select values as used by other researchers in their study of AQM-based congestion control (i.e. Hollot, Misra, Towsley, & Gong, 2002; Floyd, Gummadi, & Shenker, 2001).

7.3.1 Scenarios I

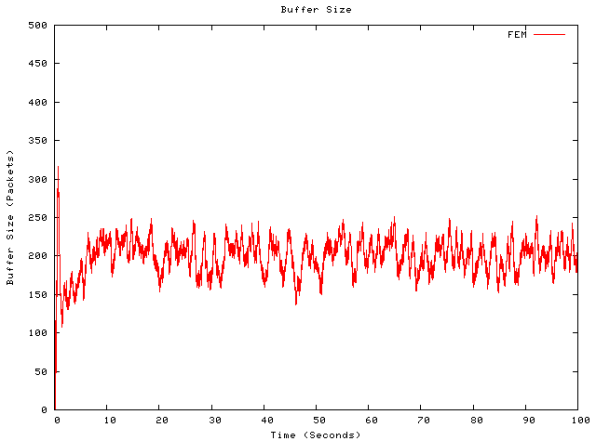
Table B.3 (see Appendix B) contains the statistical results (mean queuing delay and its standard deviation, loss rate, and bottleneck link utilization) obtained for each of the AQM schemes under comparison for the conducted experiments.

7.3.1.1 Scenario I-1: Simple case

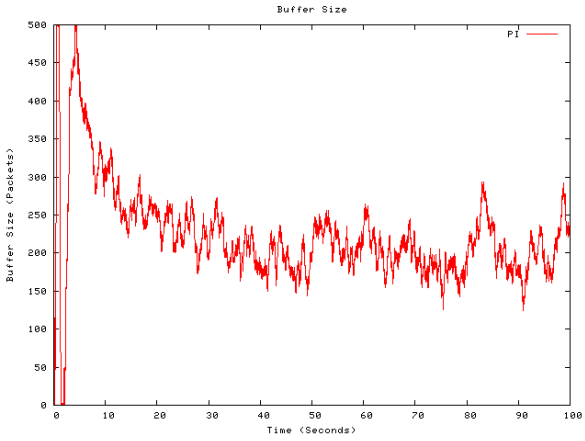
We conduct a first comparison of the selected AQM mechanisms using the following parameters:

- number of TCP flows (greedy sustained FTP traffic) is 60,
- link capacities and propagation delays are set as
    - (C_1, d_1) = (15Mbps, 40ms),
    - (C_2, d_2) = (15Mbps, 5ms), which is the bottleneck (note that 15Mbps is adopted in simulations used in Hollot, Misra, Towsley, and Gong, (2002) and Floyd, Gummadi, and Shenker (2001), and in other published papers regarding AQM-based congestion control), and
    - (C_3, d_3) = (30Mbps, 5ms).

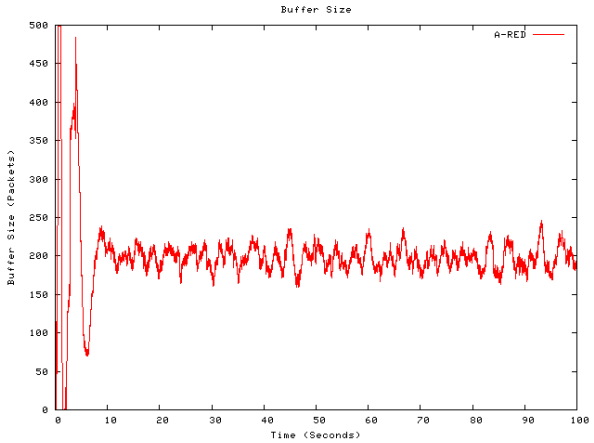
Figure 7.2 shows the queue length with respect to time of all AQM schemes under comparison. FEM quickly regulates the queue to the reference value, while the PI controller after a significant overshoot spends a considerably long time to reach steady state. A-RED and REM show good control performance, however, after a significant transient period with large overshoots. AVQ, on the other hand, does not have any explicit control for the queue length. It always tries to keep the queue length as small as possible. However, the queue length cannot be controlled, as it oscillates from empty to 200 packets approximately. This has as a negative impact the AVQ scheme to have the longest variation in delay (see Table B.3), and also the worst utilization, as compared to the other AQM schemes. FEM achieves the highest utilization, with no losses and the lowest delay variation (see Table B.3).



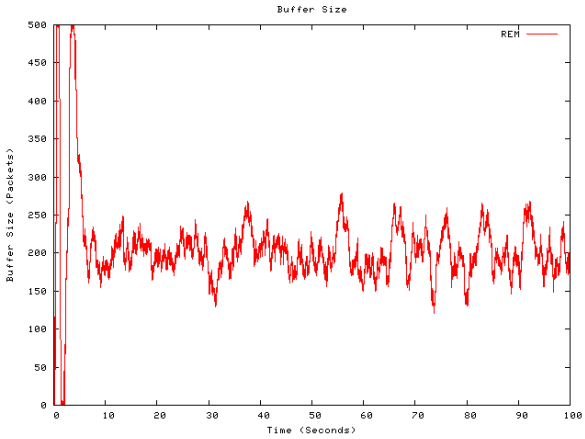
(a) FEM



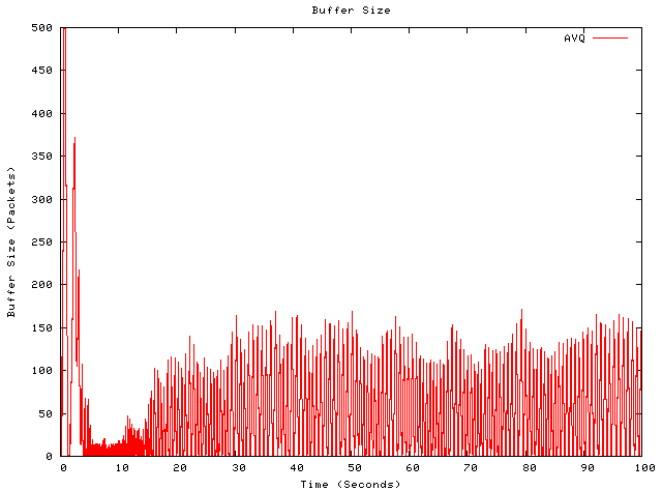
(b) PI



(c) A-RED



(d) REM



(e) AVQ

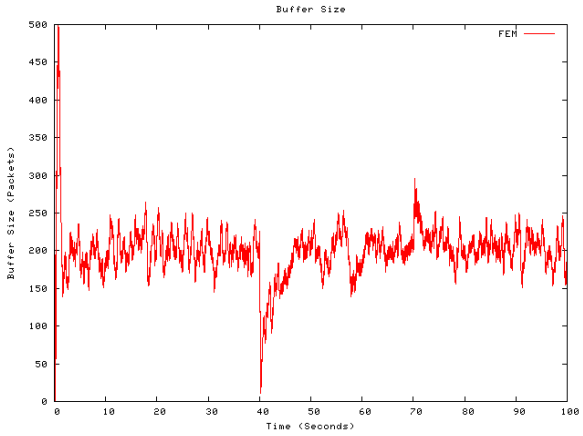
Figure 7.2 Scenario I-1: Queue lengths

7.3.1.2 Scenario I-2: Transient Performance – Speed of Response

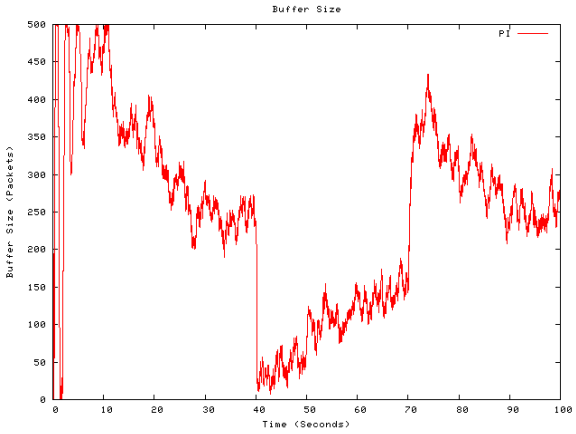
In *Scenario I-2* we increase the number of flows from 60 to 100, in order to explore the transient performance of the AQM schemes. The performance of the AQM schemes under dynamic traffic changes is also examined. We provide some time-varying dynamics by stopping half of the flows at time *t = 40 sec* and resuming transmission at time *t = 70 sec*. The results (see Figure 7.3 and Table B.3) show that FEM is very robust against the dynamic traffic changes and keeps very good response by successfully maintaining the queue length at the target value. Further, FEM has the highest utilization and lowest delay variation with minimal losses, as compared to the others (see Table B.3). PI and REM are not as robust (especially in the case of PI), as they are slower to settle down to the reference value, resulting in large queue fluctuation. A-RED responds well, except for some larger overshoots at the time of the traffic changes. AVQ shows weaknesses in responding to dynamic changes, as it exhibits sluggishness in adapting to the changing network conditions. This has as a consequence, to have large variations in delay, and the lowest link utilization.

7.3.1.3 Scenario I-3: Effect of Heterogeneous Propagation Delays

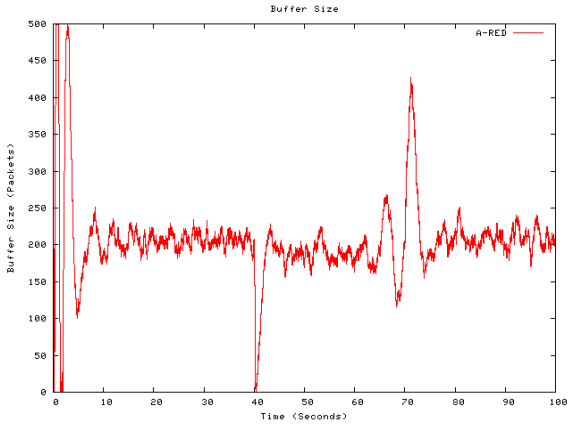
In *Scenario I-3*, we investigate the effect of having heterogeneous propagation delays at the access links. In particular, we use the previous *Scenario I-2* and we split the *100* flows into 5 groups with propagation delays ranging from *5 – 25 msec*, with a step increase of *5 msec* for each group. The queue length evolutions of the AQM schemes are shown in Figure 7.4, and the statistical results in Table B.3. We can observe that FEM is not influenced by the heterogeneity of the propagation delays, and in fact it increases the link utilization with a small decrease in the delay variation, thus achieving the highest utilization and the lowest delay variation with no losses. On the other hand, A-RED has increased its losses, and the utilization has decreased. PI has increased a bit its losses, and REM has shown a more sluggish response than in the previous scenario. AVQ has increased its utilization, retaining however the large sluggishness in adapting to the changing network conditions.



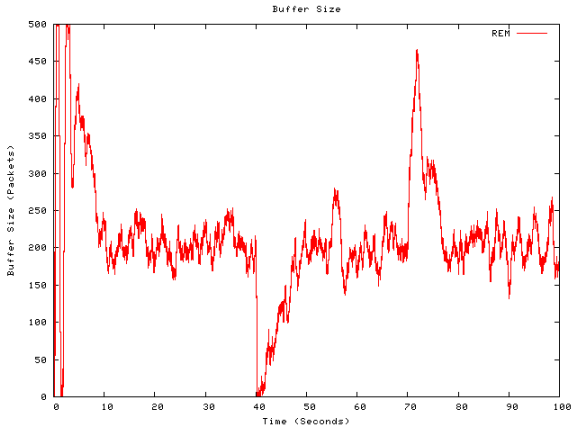
(a) FEM



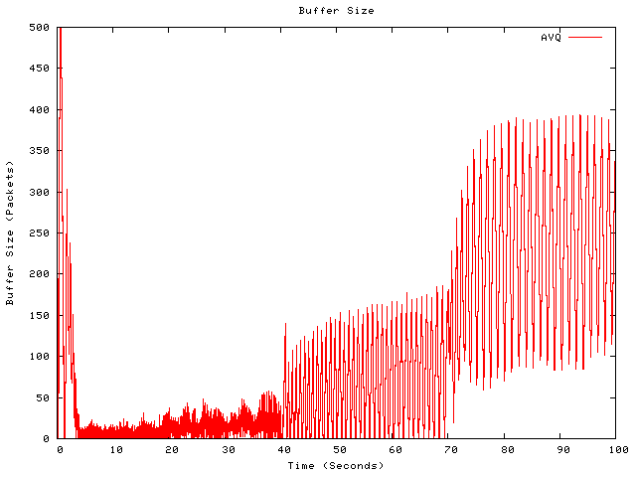
(b) PI



(c) A-RED

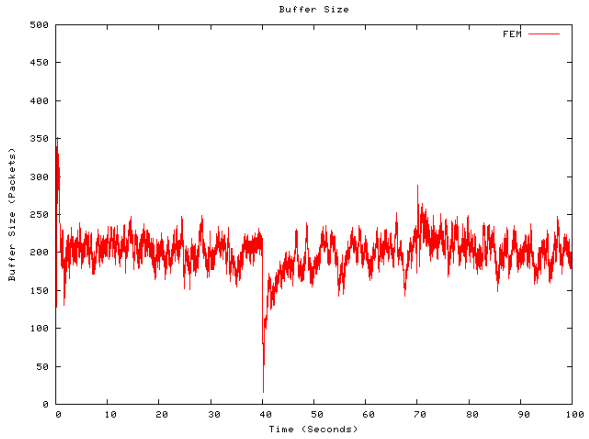


(d) REM

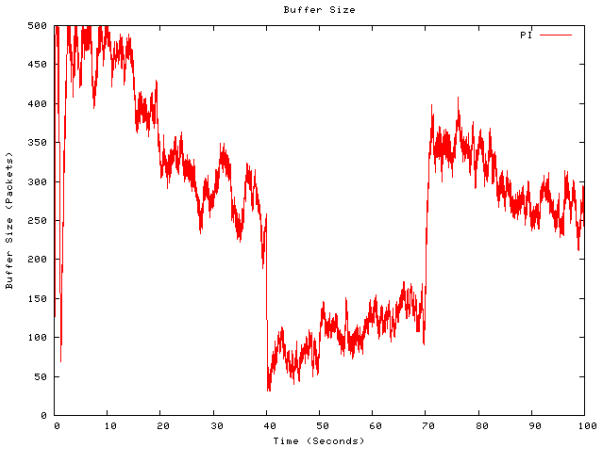


(e) AVQ

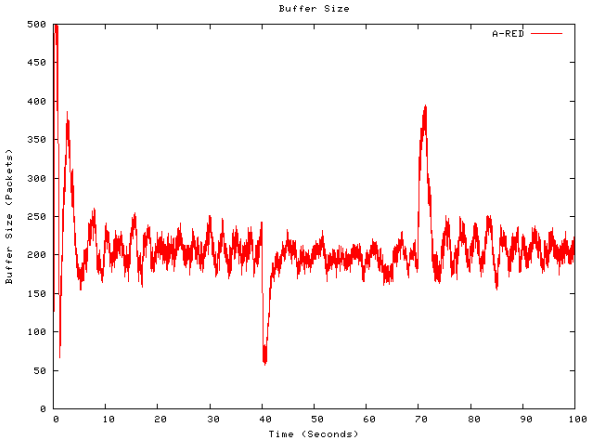
Figure 7.3 Scenario I-2: Queue lengths



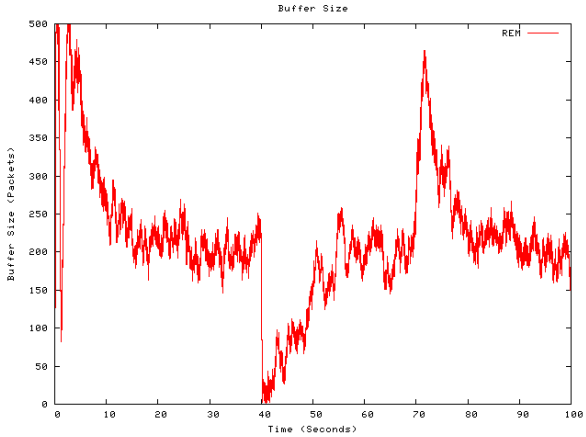
(a) FEM



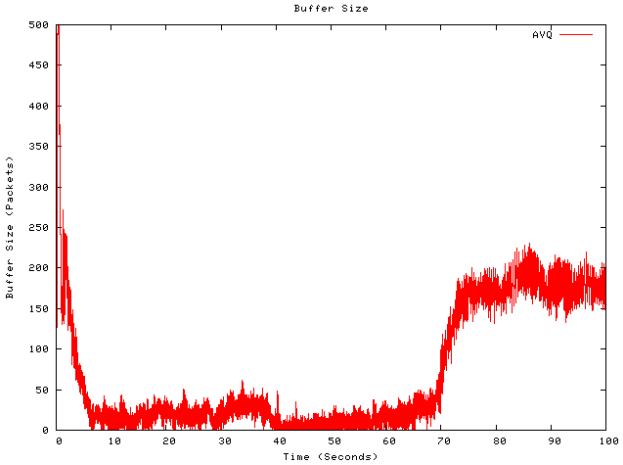
(b) PI



(c) A-RED



(d) REM



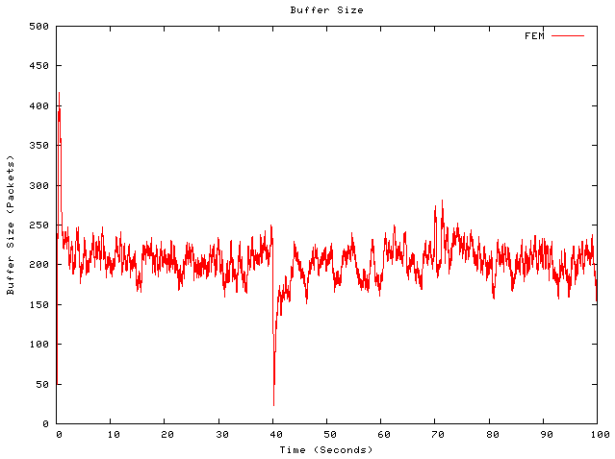
(e) AVQ

Figure 7.4 Scenario I-3: Queue lengths

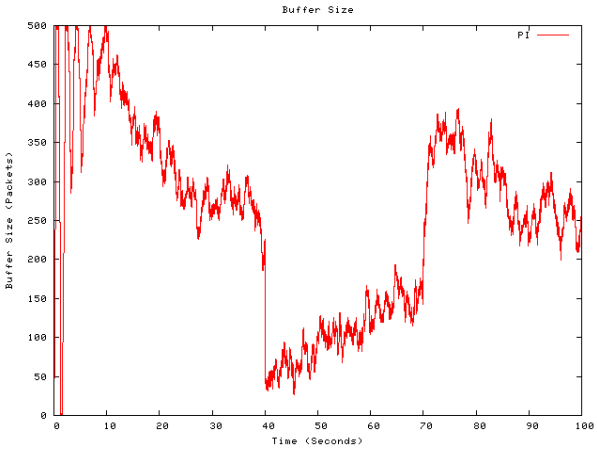
7.3.1.4 Scenario I-4: Effect of Delays

In *Scenario I-4*, we investigate the performance of AQM schemes under variation of bottleneck link propagation delays. We specifically examine the effect of the round-trip time by increasing the propagation delay to *30, 60, and 120 msec*. We have also increase the access link capacities, that is, we set (C_1, d_1) = (100Mbps, 5ms) and (C_3, d_3) = (200Mbps, 5ms), while we keep the number of flows, $N = 100$. We also keep the time-varying dynamics on the network, as used in *Scenario I-2*. The results are shown in Figure 7.5 up to Figure 7.7 (for *30msec*, 60msec, *and* 120msec, respectively), and Table B.3. From the results, we can observe the superior steady performance of FEM with stable queue dynamics, irrespective of the increase of RTT. FEM has the highest utilization, and the lowest losses and the shortest delay variation. PI, REM, A-RED and AVQ exhibit large queue fluctuations that result in degraded utilization and high variance of queuing delay. Thus, these mechanisms are shown to be sensitive to variations of RTT.

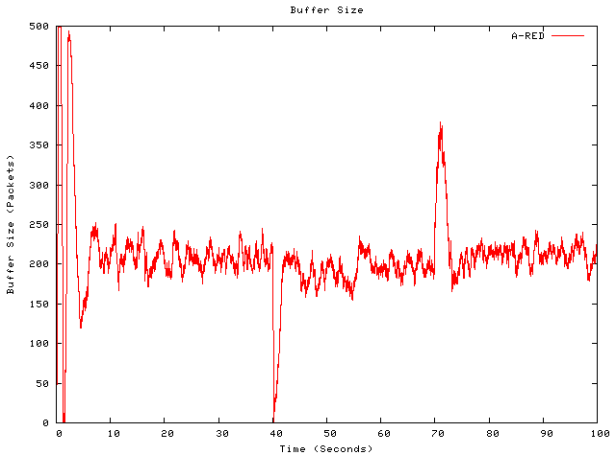
This is clearly illustrated in Figure 7.8, where we show the utilization of the bottleneck link with respect to the mean queuing delay, as well as in Figure 7.9, where we can see the utilization with respect to the queuing delay variation. FEM outperforms the other AQMs, in managing to achieve high utilization, and at the same time regulating the queue and thus providing bounded mean delay, and delay variation.



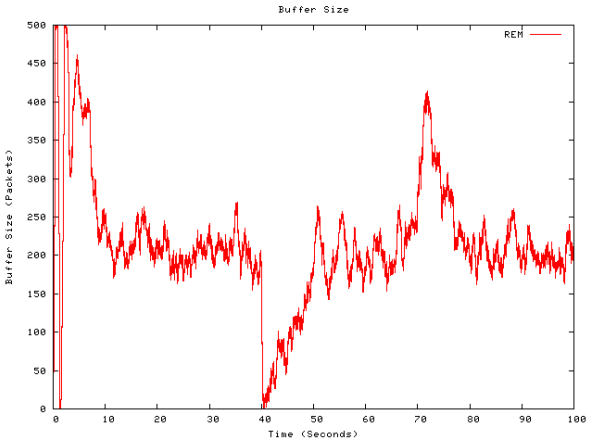
(a) FEM



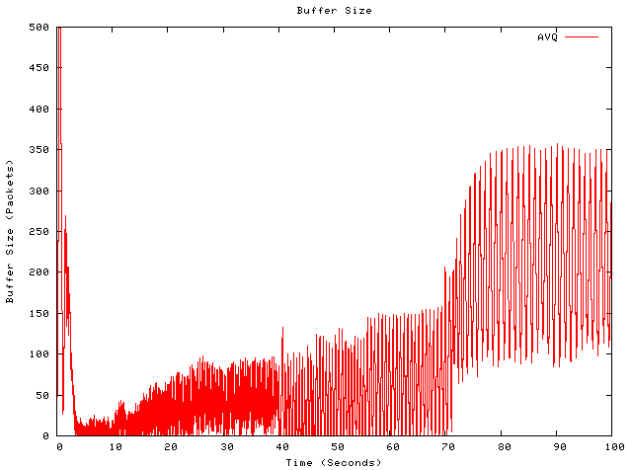
(b) PI



(c) A-RED

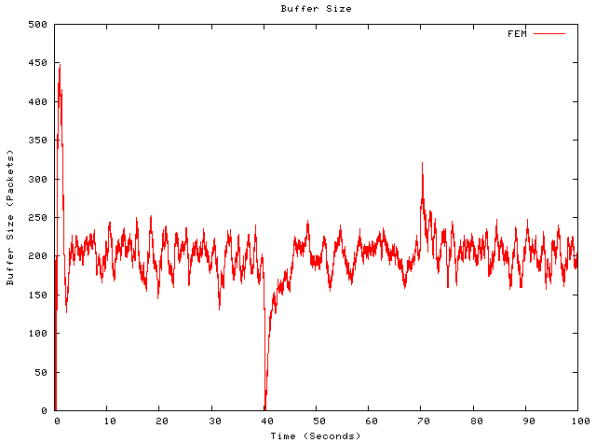


(d) REM

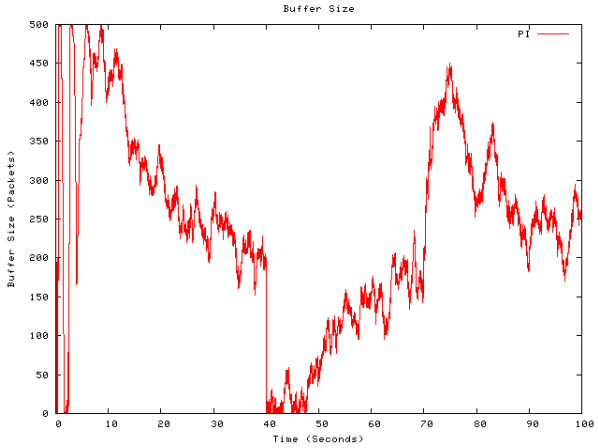


(e) AVQ

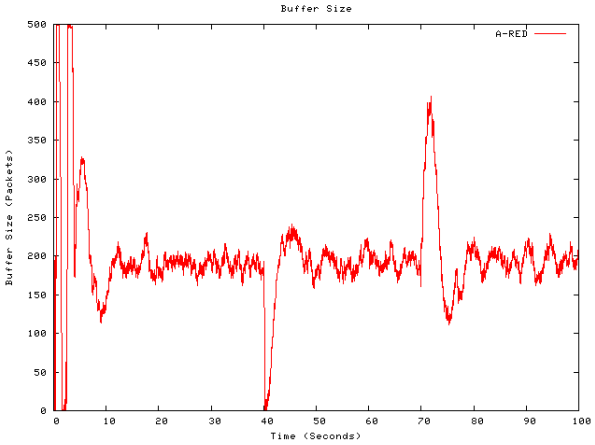
Figure 7.5 Scenario I-4: Queue lengths for bottleneck prop. delay = 30 msec



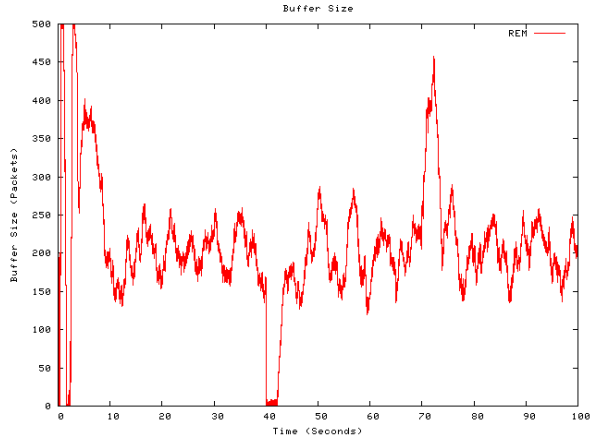
(a) FEM



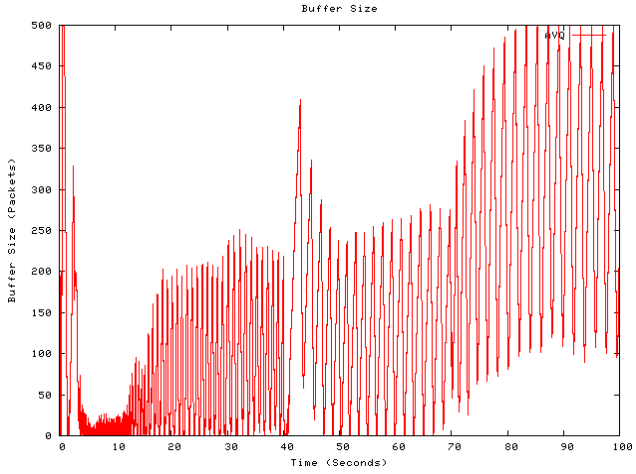
(b) PI



(c) A-RED

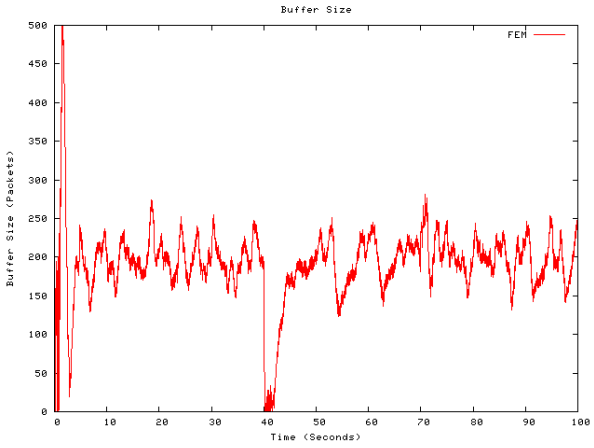


(d) REM

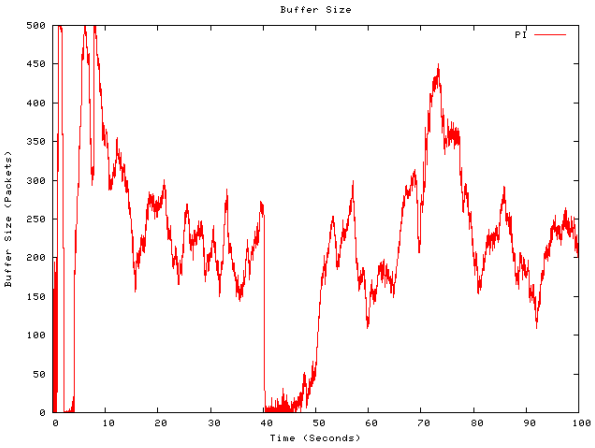


(e) AVQ

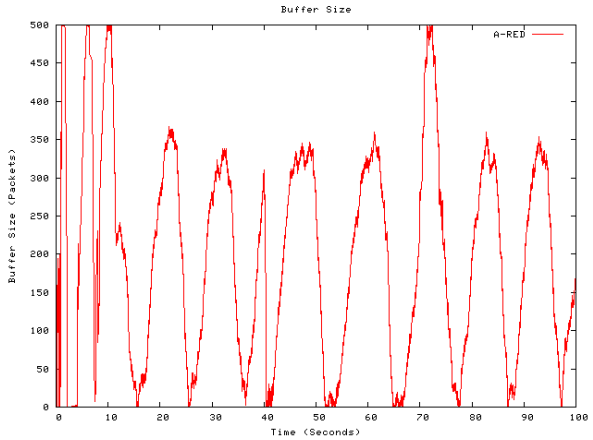
Figure 7.6 Scenario I-4: Queue lengths for bottleneck prop. delay = 60 msec



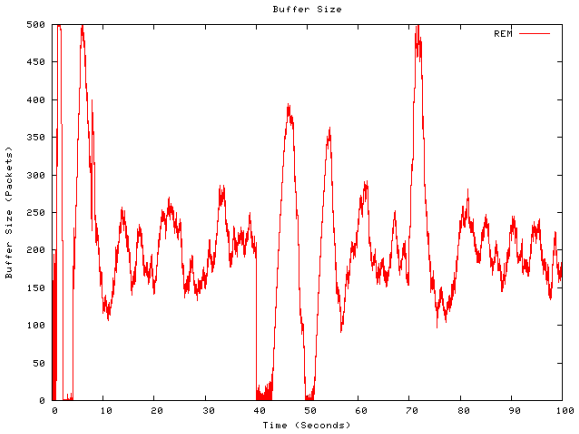
(a) FEM



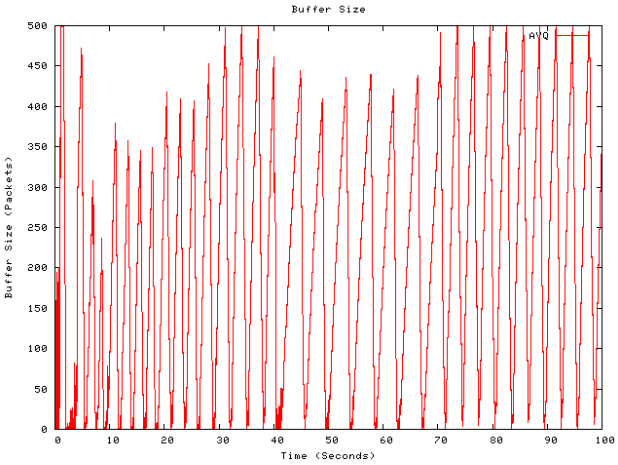
(b) PI



(c) A-RED



(d) REM



(e) AVQ

Figure 7.7 Scenario I-4: Queue lengths for bottleneck prop. delay = 120 msec

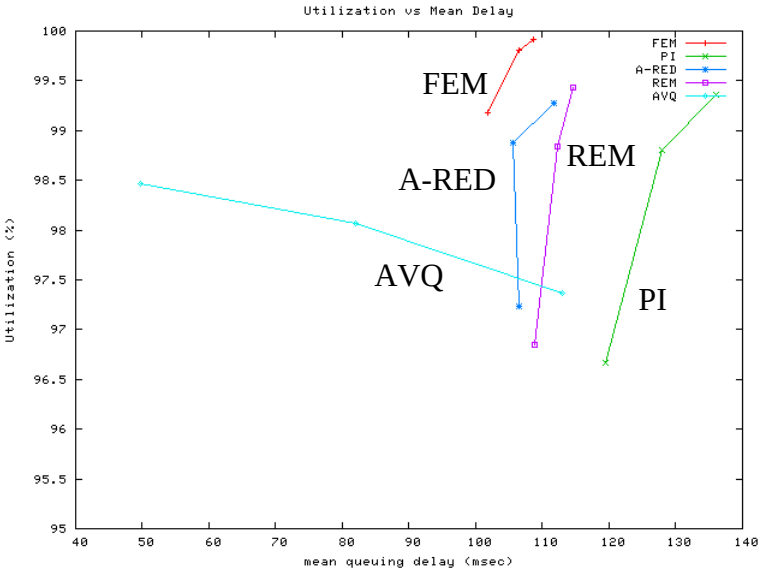


Figure 7.8 Scenario I-4: Utilization vs mean delay
(bottleneck propagation delay varies from 30, 60, 120 msec)

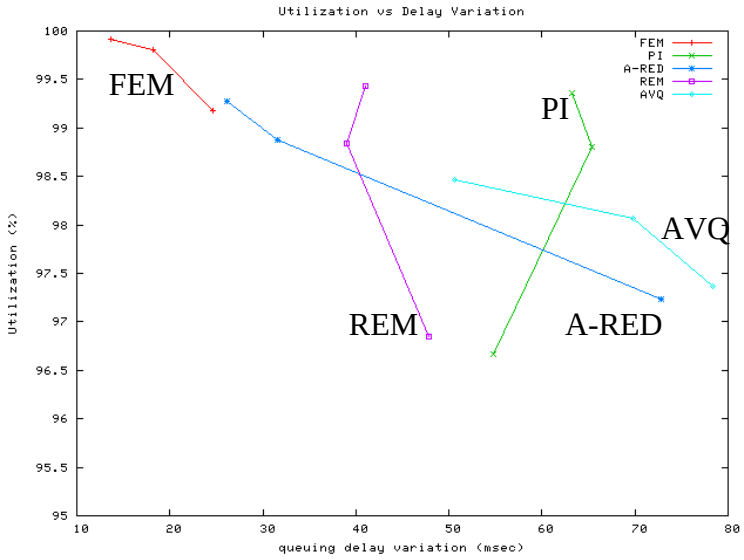


Figure 7.9 Scenario I-4: Utilization vs delay variation
(bottleneck propagation delay varies from 30, 60, 120 msec)

7.3.1.5 Scenario I-5: Effect of Traffic Load

Scenario I-5 investigates the effect of the traffic load factor (*N*) in the last scenario (i.e., bottleneck propagation delay = *120 msec*), by increasing *N* from 100 to 200, 300, 400, and 500 flows. Figure 7.10 shows the loss rate as traffic load increases, where it can be seen that FEM has the lowest drops. FEM shows stable and low packet loss over large traffic load. A-RED has the largest drops with a large increase of packet loss with respect to higher loads. Figure 7.11 and Figure 7.12 show the utilization of the bottleneck link with respect to the mean queuing delay, and with respect to the delay variation, respectively. FEM outperforms other AQM schemes on both high utilization and low delay variation, thus it exhibits a more stable, and robust behavior with a bounded delay. The other AQM schemes show a poor performance as the number of traffic load increases, achieving much lower link utilization, and large queuing delays, far beyond the expected value. From Table B.3, and Figure 13-16 (for *N=200-500*, respectively – note that for *N=100* Figure 7.7 applies), it is clear that FEM has the lowest variance in queuing delay, resulting in a stable and robust behavior. On the other hand, the other AQM schemes exhibit very large queue fluctuations with large amplitude that inevitably deteriorates delay jitter.

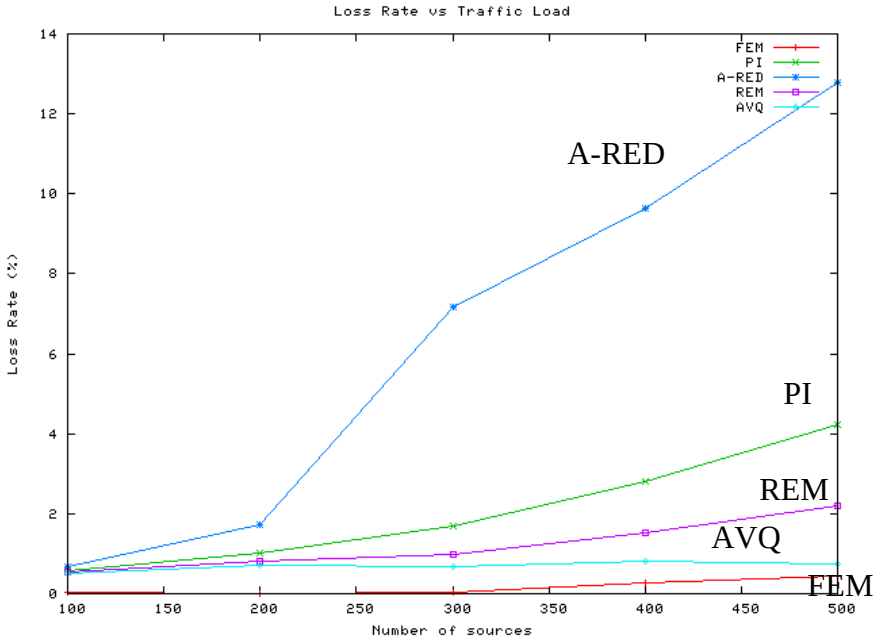


Figure 7.10 Scenario I-5: Loss Rate vs Traffic Load (for 100-500 flows)

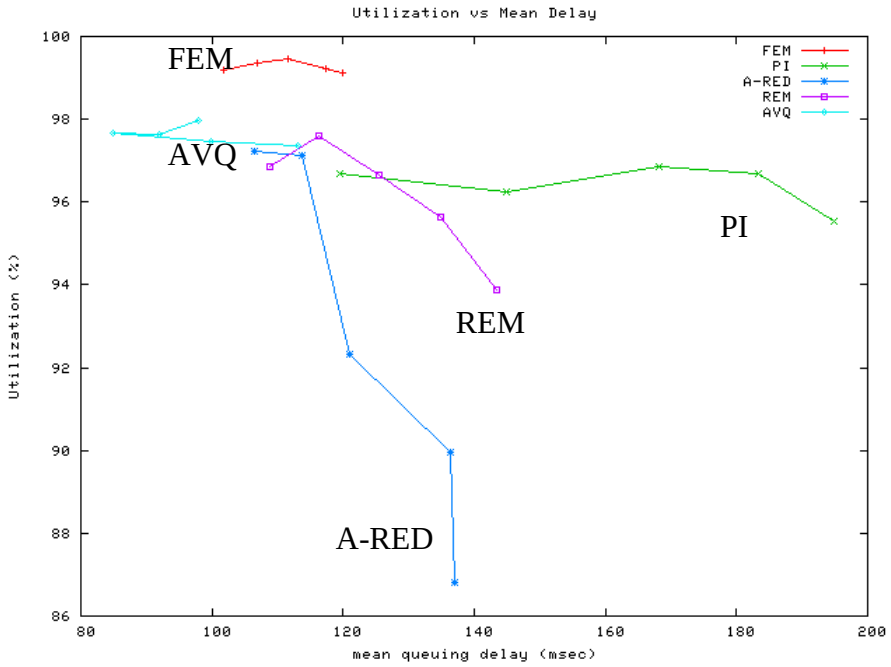


Figure 7.11 Scenario I-5: Utilization vs Mean Delay
(for 100-500 flows)

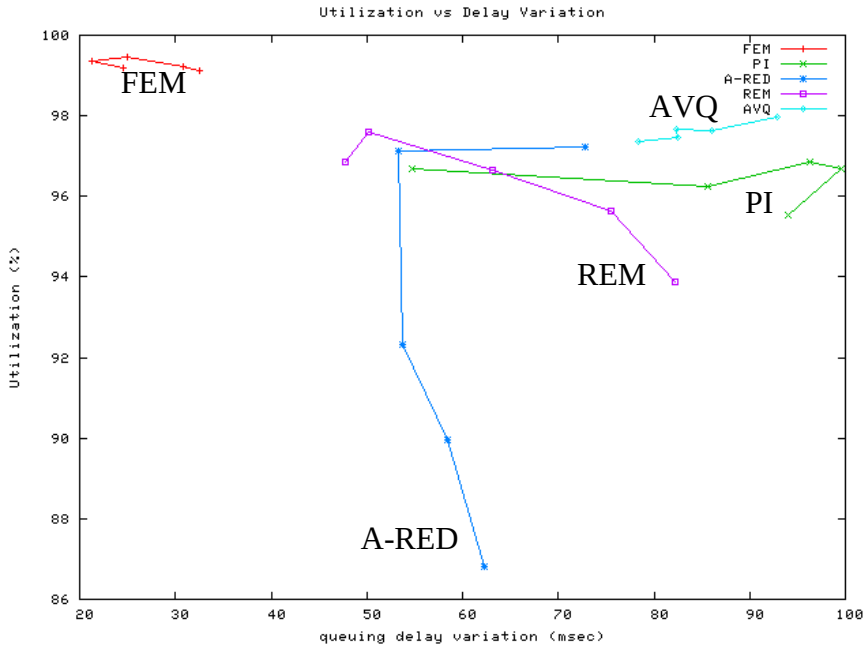
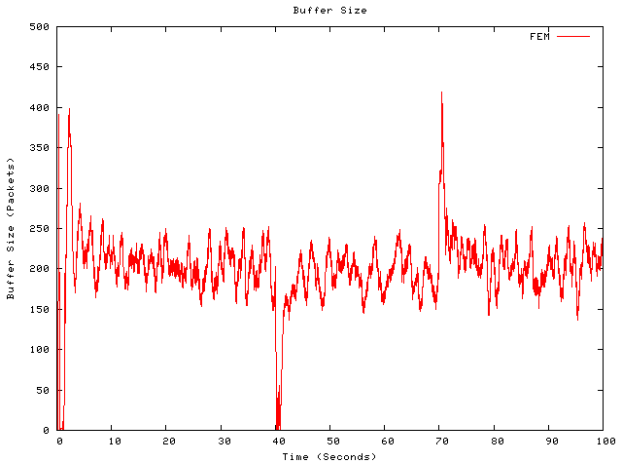
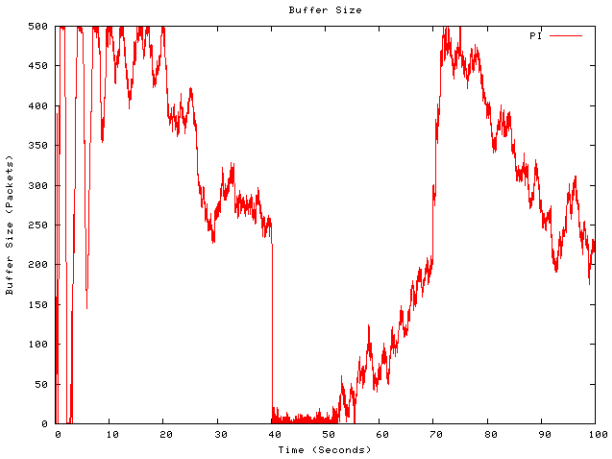


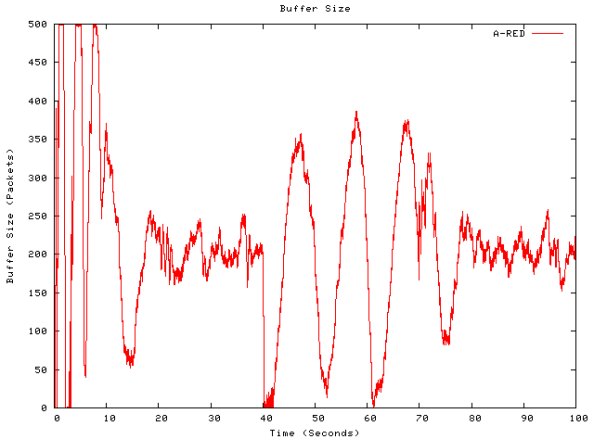
Figure 7.12 Scenario I-5: Utilization vs Delay Variation
(for 100-500 flows)



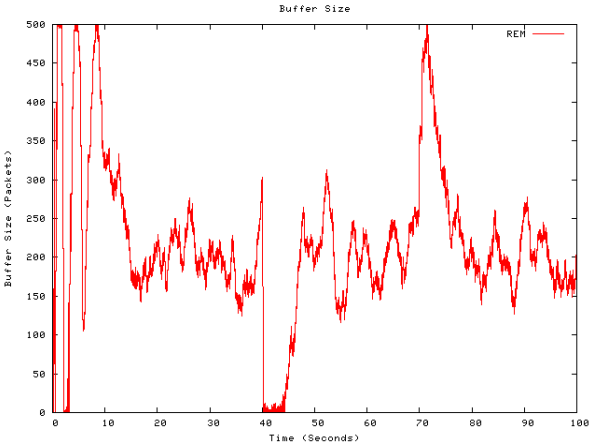
(a) FEM



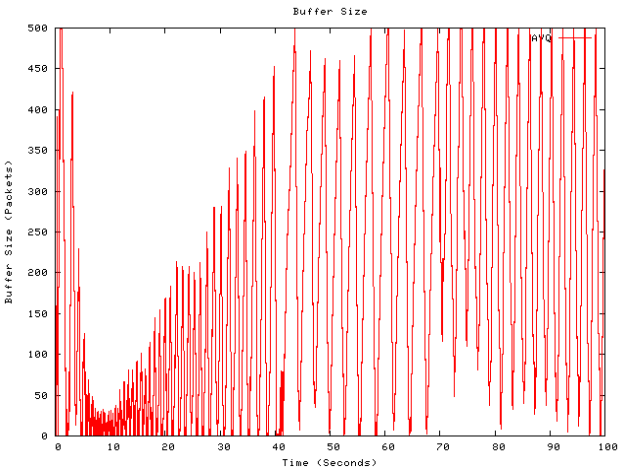
(b) PI



(c) A-RED

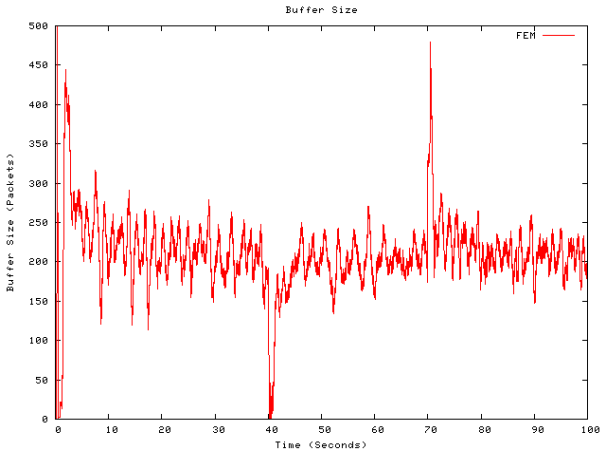


(d) REM

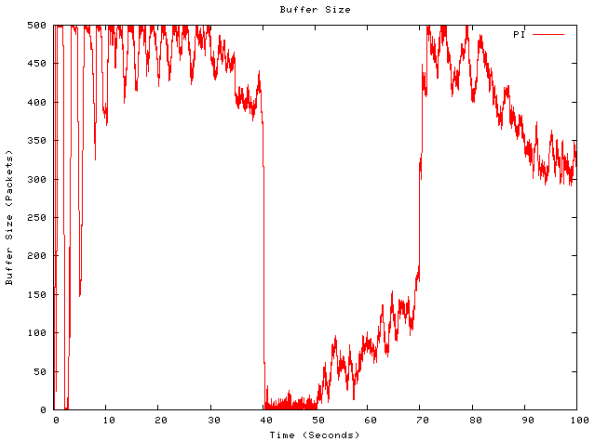


(e) AVQ

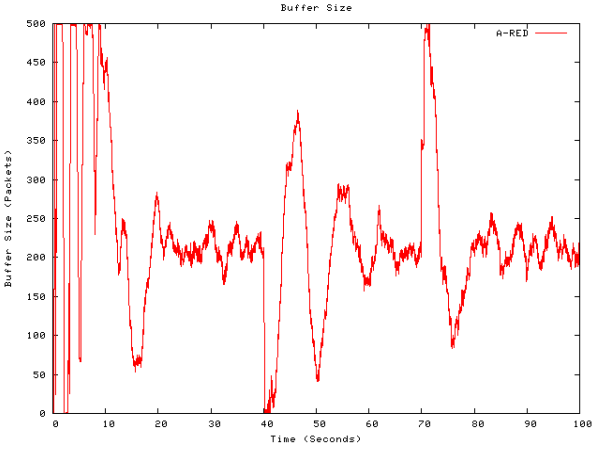
Figure 7.13 Scenario I-5: Queue lengths (for 200 flows)



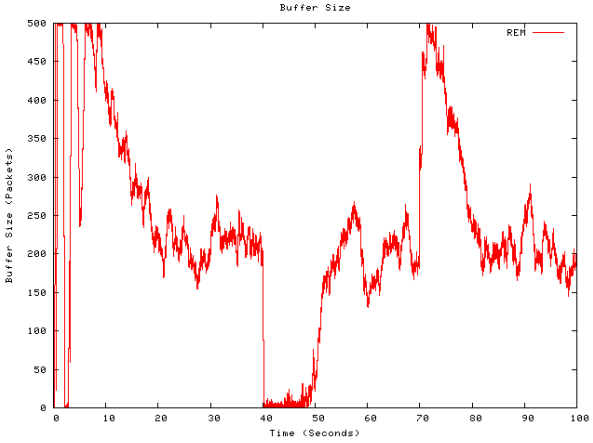
(a) FEM



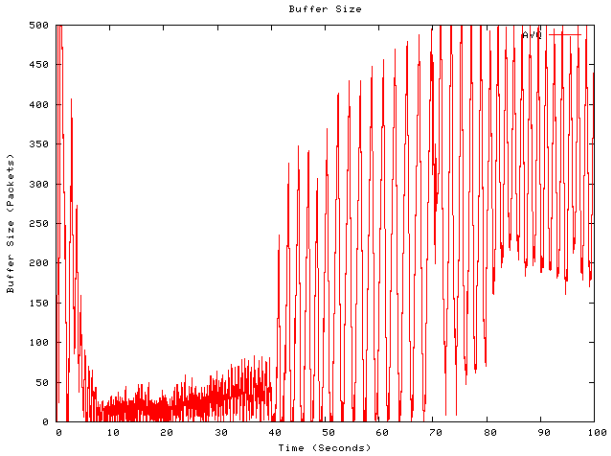
(b) PI



(c) A-RED

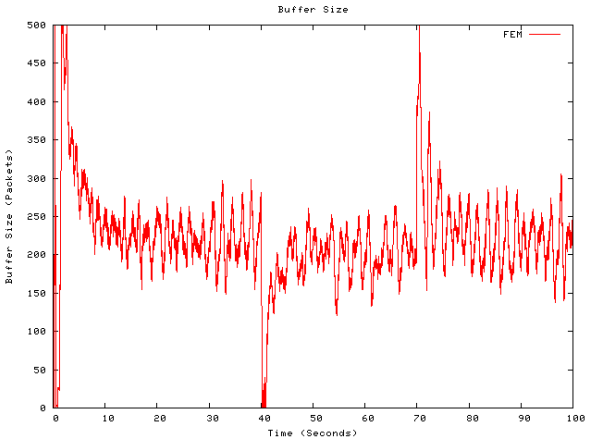


(d) REM

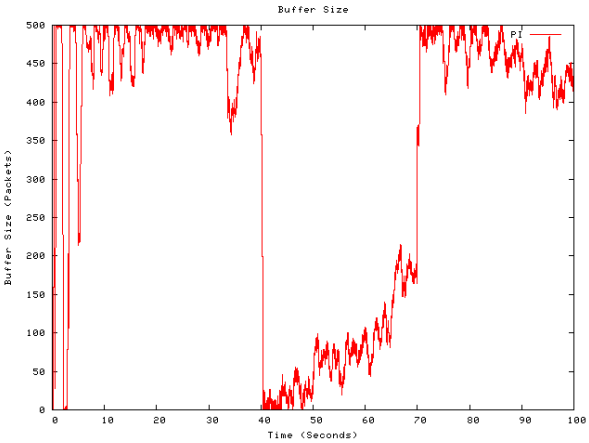


(e) AVQ

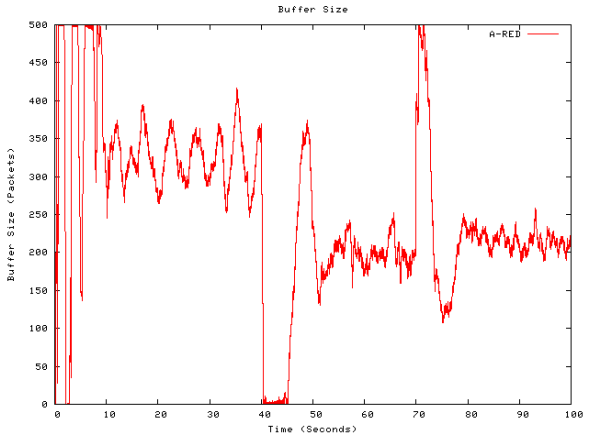
Figure 7.14 Scenario I-5: Queue lengths (for 300 flows)



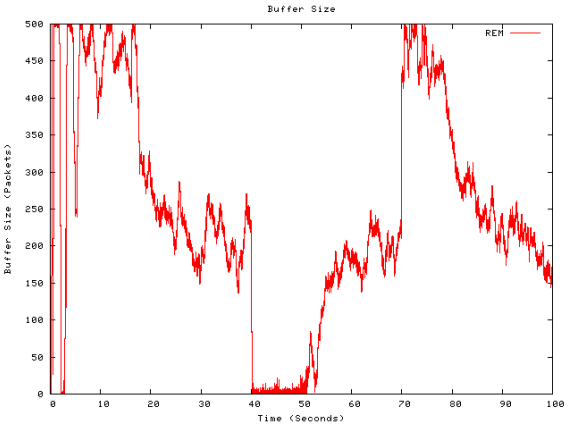
(a) FEM



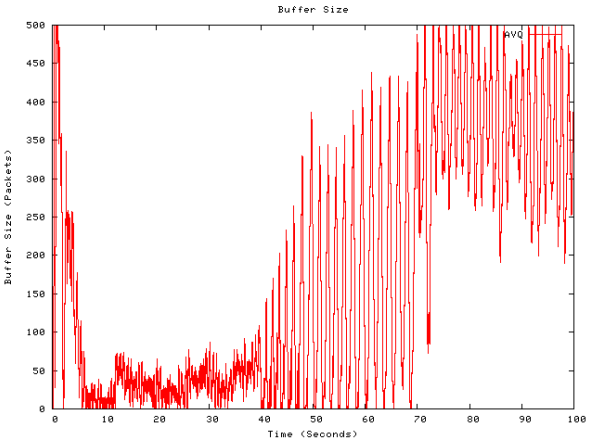
(b) PI



(c) A-RED

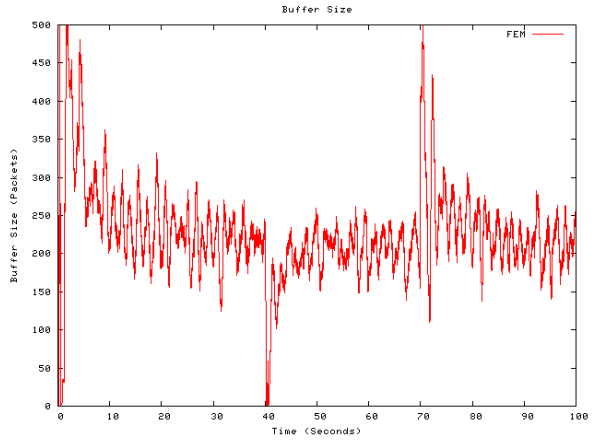


(d) REM

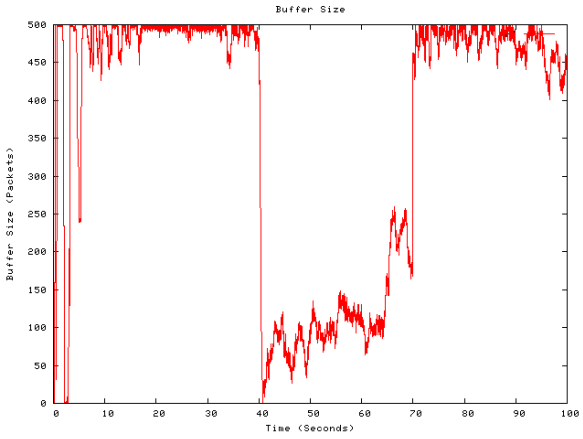


(e) AVQ

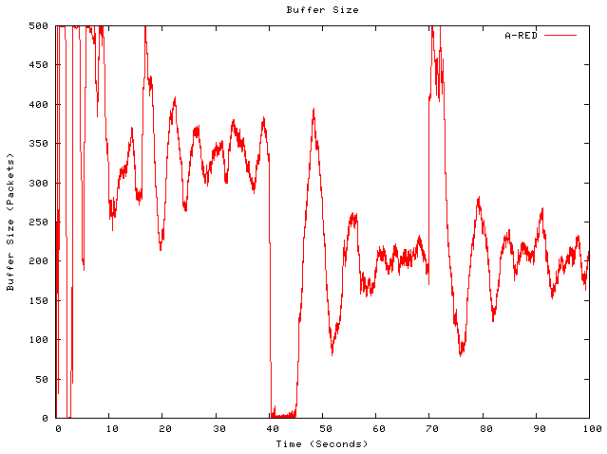
Figure 7.15 Scenario I-5: Queue lengths (for 400 flows)



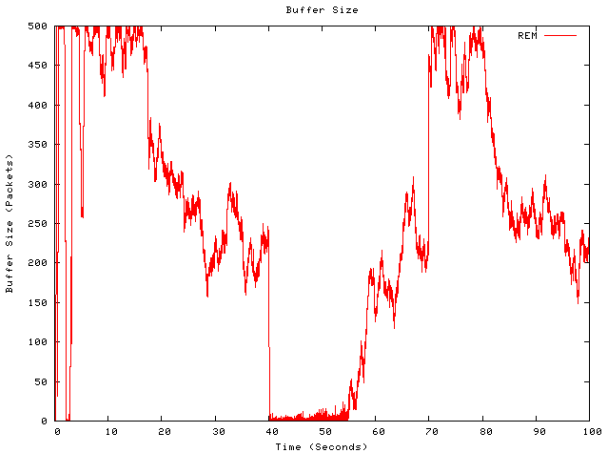
(a) FEM



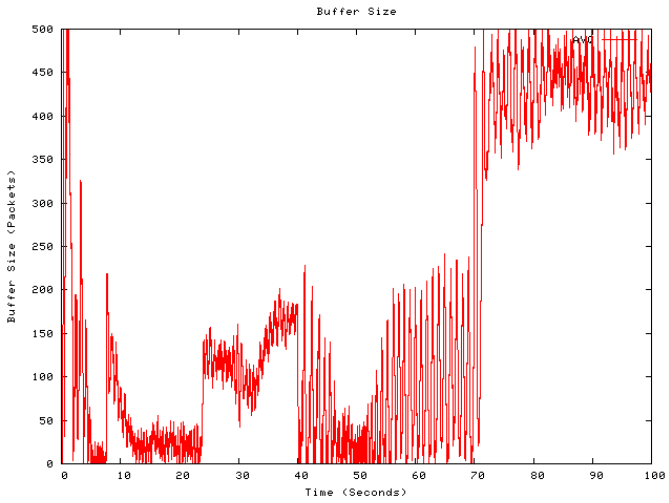
(b) PI



(c) A-RED



(d) REM

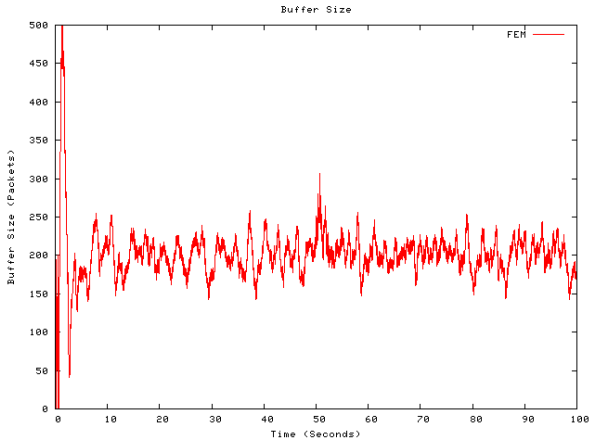


(e) AVQ

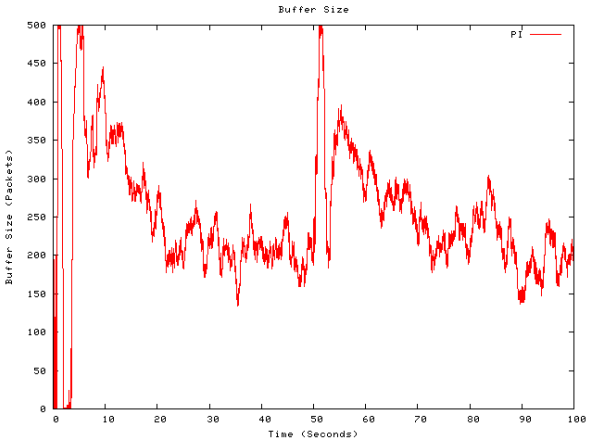
Figure 7.16 Scenario I-5: Queue lengths (for 500 flows)

7.3.1.6 Scenario I-6: Performance in the Presence of Short-lived Flows

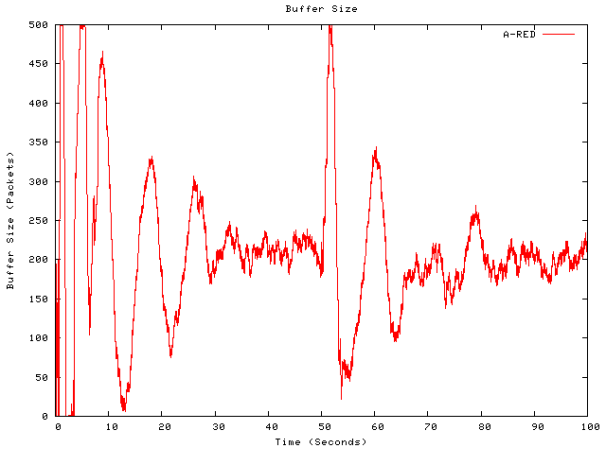
Scenario I-6 investigates the performance of AQM schemes by introducing additional web-like traffic that can be seen as noise-disturbance to the network. As a large part of the connections in the Internet comprise of short-lived flows, it is important to study the response of the AQM schemes under comparison in the presence of short flows. In particular, for a bottleneck link propagation delay of *100 msec* we start with 100 long-lived TCP flows, and at the middle of the simulation (i.e., *t=50 sec*) we introduce short-lived flows, which arrive at the link at the rate of 30 flows per second (of 20 packets each), as suggested by Kunniyur and Srikant (2004). Figure 7.17 shows the queue length evolution of the AQM schemes. We can observe the robustness of the FEM controller that adequately controls the queue at the specified TQL, without being affected by the sudden introduction of short flows. It exhibits the highest utilization, with the lowest losses, and the shortest delay variation (see Table B.3). This is in contrast with the other AQM schemes that it is evident at *t=50sec* that they are greatly influenced by the introduction of short flows. REM, PI, and A-RED after a significant transient response with large overshoots, they are slow to settle down to the reference value, whereas AVQ has further increased the delay variation at the queue, resulting in degraded utilization.



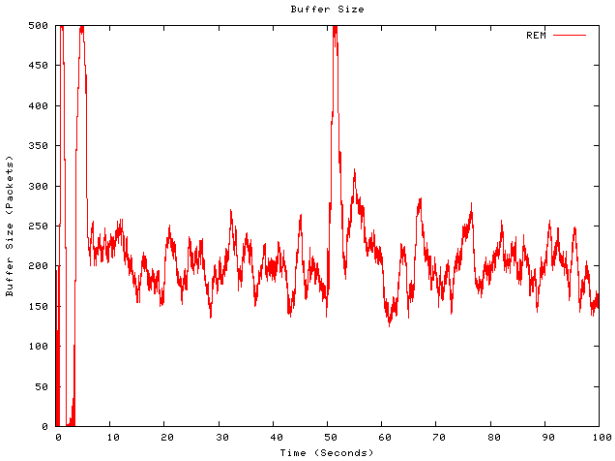
(a) FEM



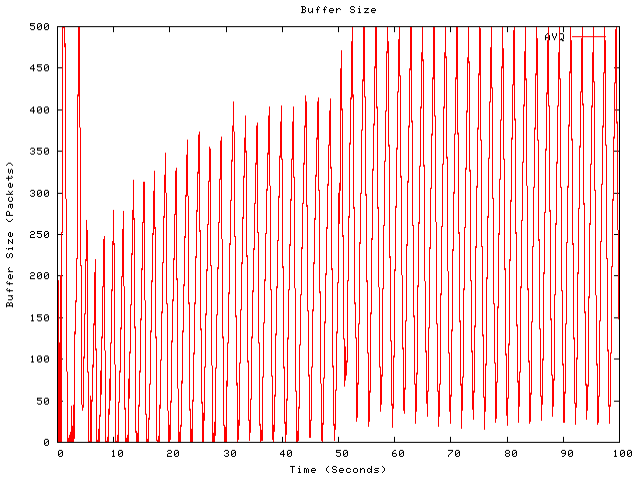
(b) PI



(c) A-RED



(d) REM



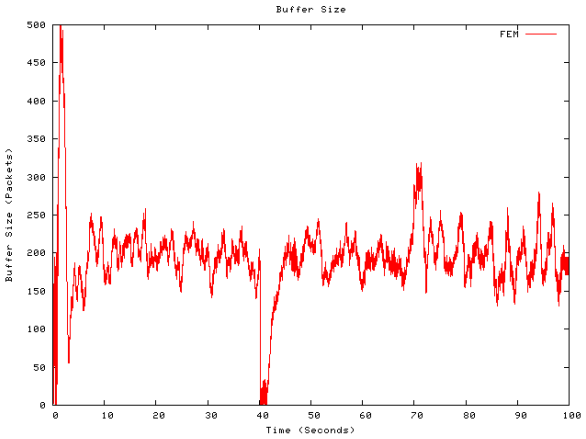
(e) AVQ

Figure 7.17 Scenario I-6: Queue lengths

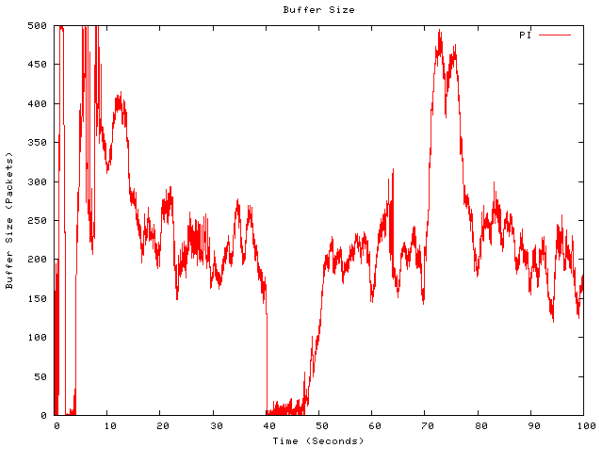
7.3.1.7 Scenario I-7: Effect of Reverse-path Traffic

Scenario I-7 investigates the effect of reverse traffic on the behavior of the AQM schemes. We particularly introduce Web traffic in the reverse path of the simulation topology, where 200 web-like sources are sending data based on distributions and parameters (see Table B.2) as introduced by Iannaccon et al. (2001). In the forward path we have 100 long-lived FTP sources sharing a bottleneck link of *120 msec* propagation delay, and we keep the time-varying dynamics as explained in *Scenario I-2*. The results are shown in Figure 7.18 and Table B.3. We further introduce 2 additional FTP flows sending data in the reverse path (see Figure 7.19 and Table B.3).

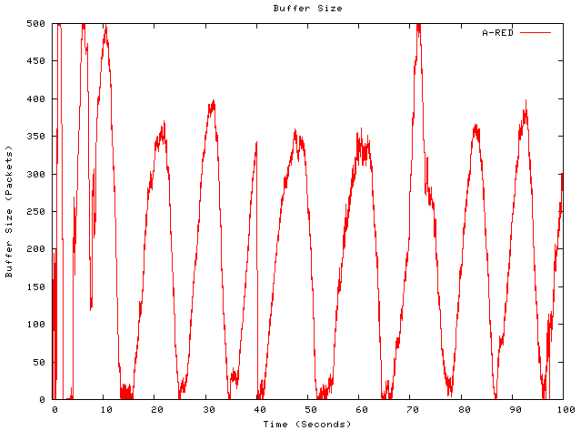
We can observe that as the reverse-path traffic increases, FEM responds adequately, and manages to keep the queue around the desired value, while it exhibits the highest utilization and the lowest – minimal – losses in comparison to the other schemes under study. On the other hand, the other AQM schemes responded badly in the presence of increased reverse-path traffic. Specifically, A-RED, PI, and REM cannot regulate the queue, and show large oscillations, with no sign of “driving” the queue at the reference value. This has as a result to have larger loss rate and much lower link utilization than FEM achieves. AVQ tries to keep the queue length as small as possible. However, this has a negative impact on the utilization of the link, as it leads to an unwanted underutilization (AVQ achieves only 81.58% of the link utilization, as opposed to FEM that achieves almost 99%), thus, it fails to accomplish its general goal of gaining high utilization.



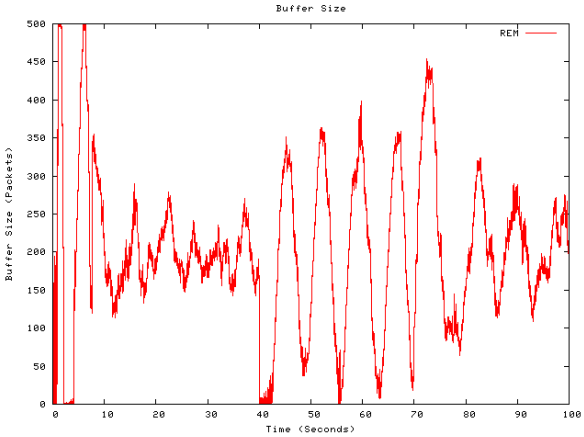
(a) FEM



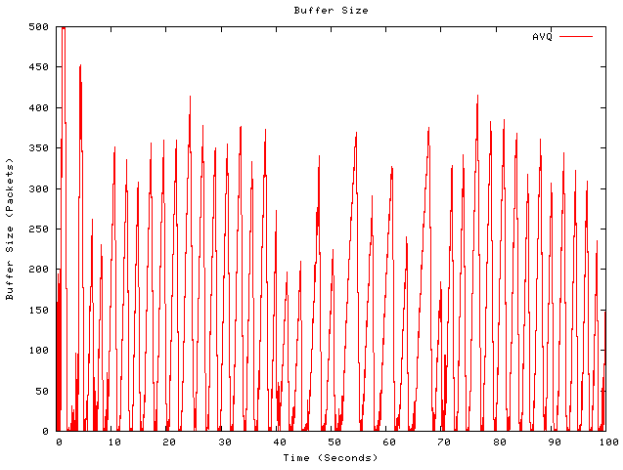
(b) PI



(c) A-RED

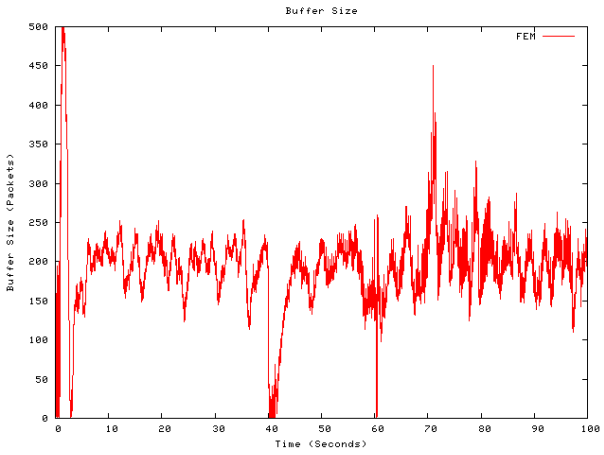


(d) REM

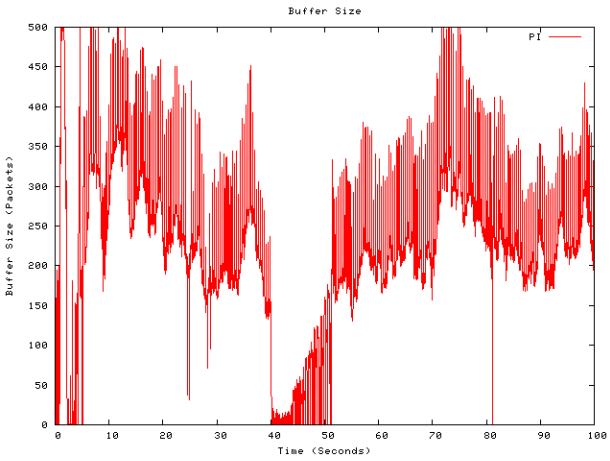


(e) AVQ

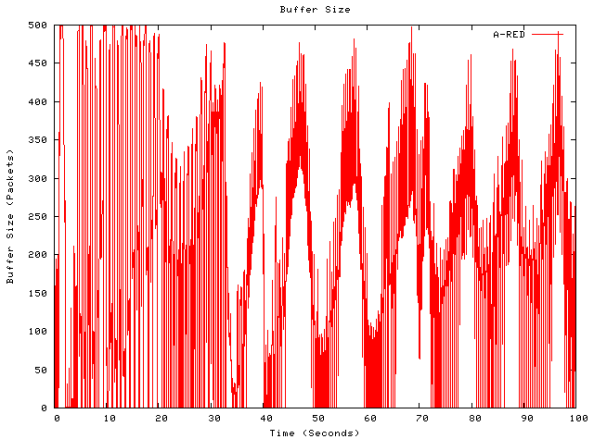
Figure 7.18 Scenario I-7: Queue lengths (with reverse-path web-traffic)



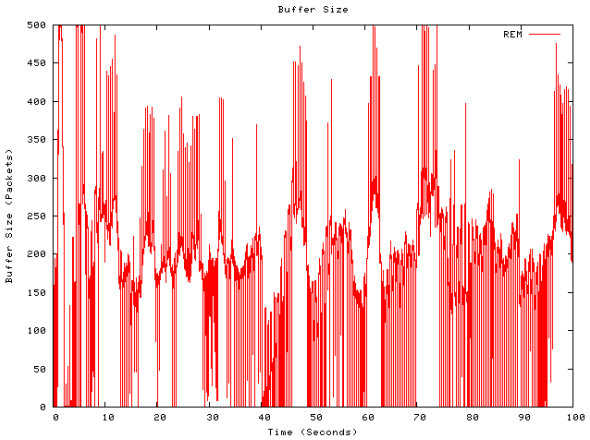
(a) FEM



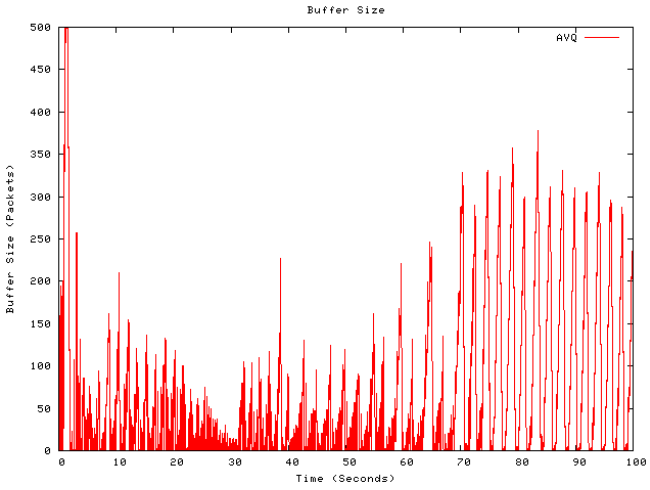
(b) PI



(c) A-RED



(d) REM

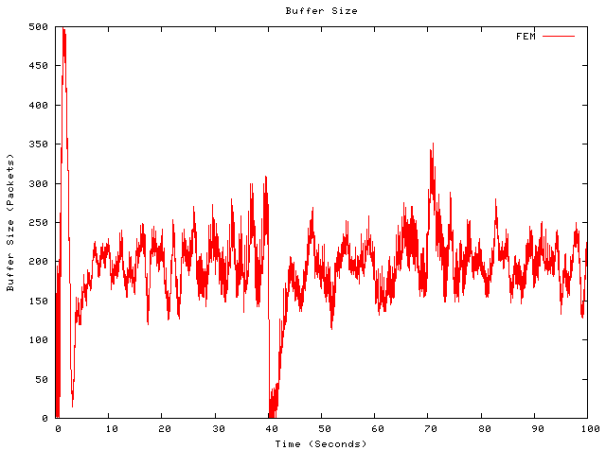


(e) AVQ (note that AVQ suffers from underutilization; it only achieves 81% of link utilization, in contrast with FEM that achieves the highest utilization among others of 99%)

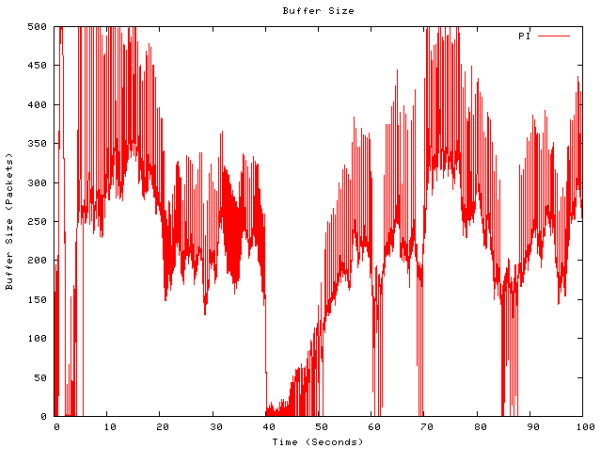
Figure 7.19 Scenario I-7: Queue lenghs (with reverse-path web-traffic and FTP)

7.3.1.8 Scenario I-8: Performance in the Presence of Unresponsive Traffic

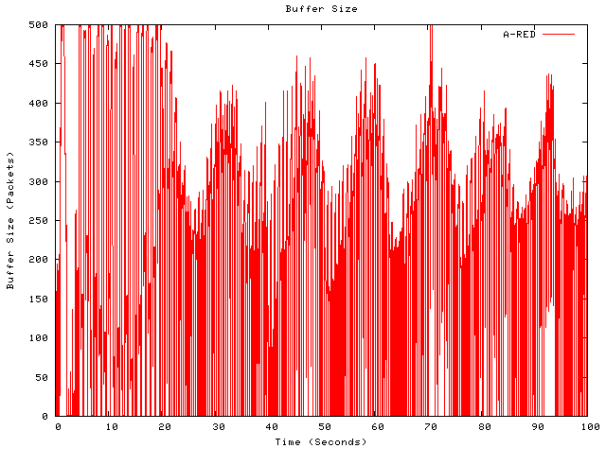
Scenario I-8 investigates the performance of AQM schemes under the presence of unresponsive traffic. We use the previous *Scenario I-7* with the reverse-path traffic, and we add two UDP unresponsive forward flows that simulate the voice-over-IP (VoIP) application with 64kbps. From Figure 7.20 and Table B.3, we can observe that the FEM controller can still manage to regulate the queue, exhibiting robust behavior, and it achieves the highest utilization of 99.12%, much larger than the other AQM schemes; thus showing its superiority against the others. On the other hand, A-RED, PI, and REM schemes exhibit large queue fluctuations, and show their weaknesses to control the queue under such situations; this has a negative impact to the link utilization. The AVQ scheme exhibits a very poor behavior; while it keeps the queue length at very low levels, at the same time the link is badly underutilized (it achieves only *44*% utilization, in contrast with FEM that achieves 99% utilization). This behavior apparently shows weakness of AVQ to find an adequate tradeoff between high utilization and low delay.



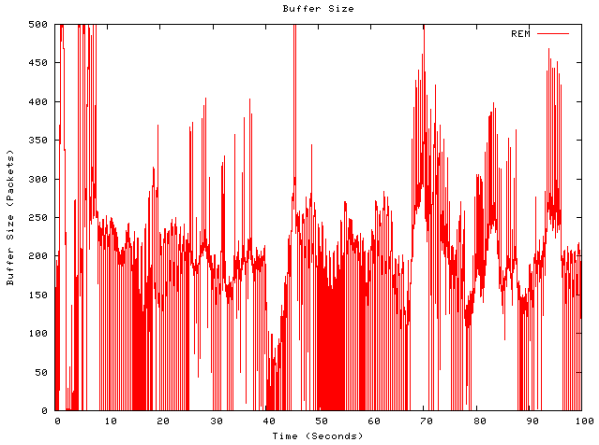
(a) FEM



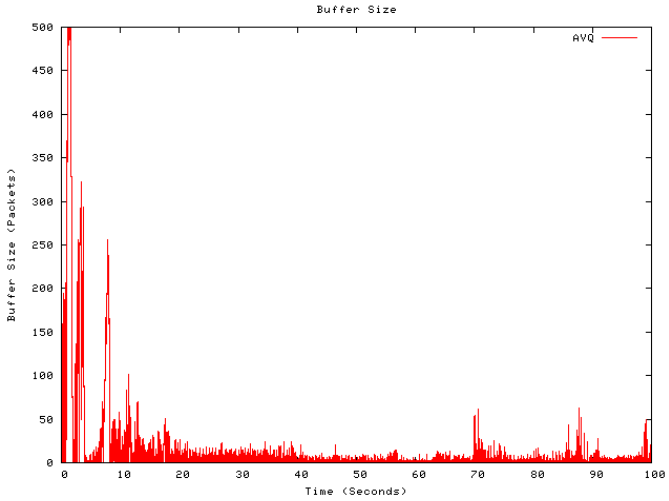
(b) PI



(c) A-RED



(d) REM



(e) AVQ (note that AVQ suffers from underutilization; it only achieves 44% of link utilization, in contrast with FEM that achieves the highest utilization among others of 99%)

Figure 7.20 Scenario I-8: Queue lengths (with reverse-path web-traffic and FTP + unresponsive forward traffic)

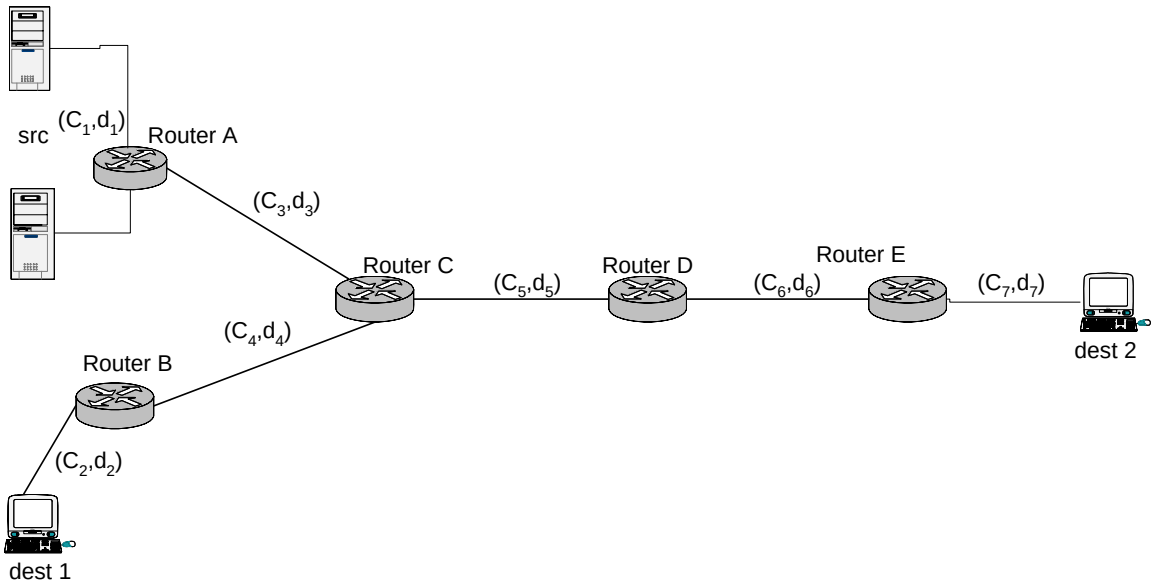


Figure 7.21 A network topology with congestion at peripheral links

7.4 Congestion at Peripheral Links

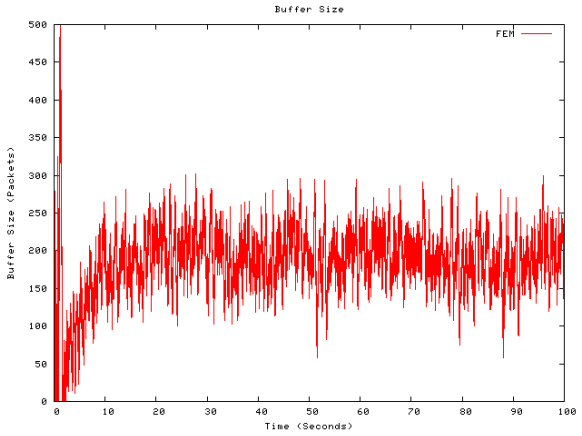
Figure 7.21 is used for the following simulations. Its topology consists of a high-bandwidth *1Gps* core link between router-C and router-D, with a set of lower-bandwidth *45 Mbps* attached to it, and sources and receivers connected by *100 Mbps*. Particularly, we set (C_1, d_1) = (C_2, d_2) = (C_7, d_7) = (100Mbps, 5ms), (C_3, d_3) = (C_4, d_4) =(C_6, d_6) = (45Mbps, 30msec), and (C_5, d_5) = (1Gps, 10msec). With these settings (taken by Bitorika et al. (2004)), the congestion occurs typically in the *45 Mps* links between the high-bandwidth core and the LAN-like traffic source links. This scenario provides a more realistic network topology, in which the core is over-provisioned and congestion occurs on more peripheral links.

7.4.1 Scenarios II

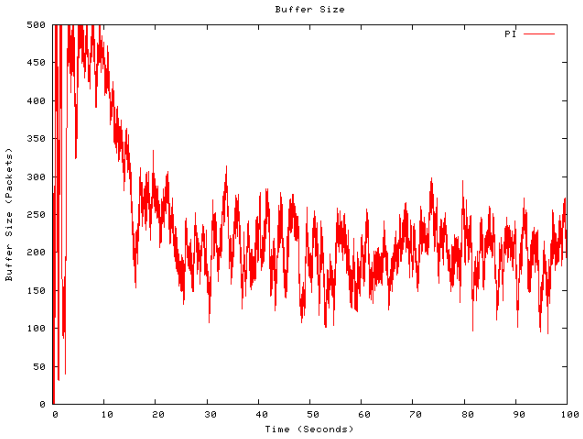
Table B.4 (see Appendix B) contains the statistical results of the conducted experiments. Note that we keep the control parameter values of the AQM schemes as used in Section 7.3.

7.4.1.1 Scenario II-1

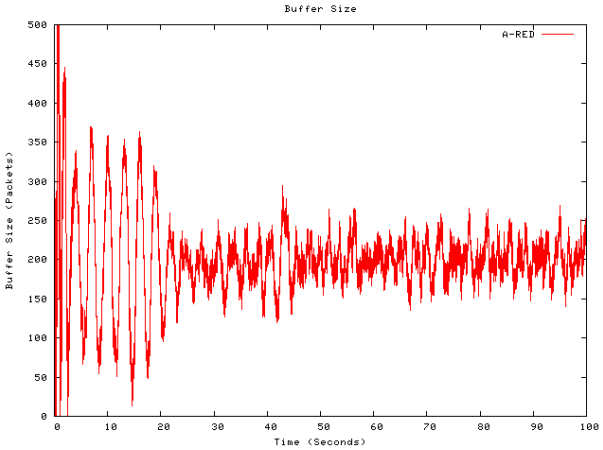
In *Scenario II-1* we use 100 FTP flows sending data from one edge to the other, passing through the core link, from router-A to router-E, and finally to destination node attached to router-E. Another 100 FTP flows are active sending data from the LAN-like *100Mbps* link to router-A, with destination node the one attached to router-B, through the *45Mps* link between router-A and router-C. Thus, the peripheral link of *45Mps* at router-A is the bottleneck link. Table B.4 shows the statistical results obtained and Figure 7.22 shows the queue length evolution with respect to time of each AQM scheme. FEM controller manages to control the queue around the reference point, while it exhibits the highest utilization of the *45Mps* link, and no losses. On the other hand, the other AQM schemes have large oscillations, and show slow response to regulate the queue. This has a negative impact with the occurrences of losses, and lower link utilization than FEM has. The AVQ exhibits large variation in delay that degrades the link utilization; it exhibits the lowest link utilization as compared with the others.



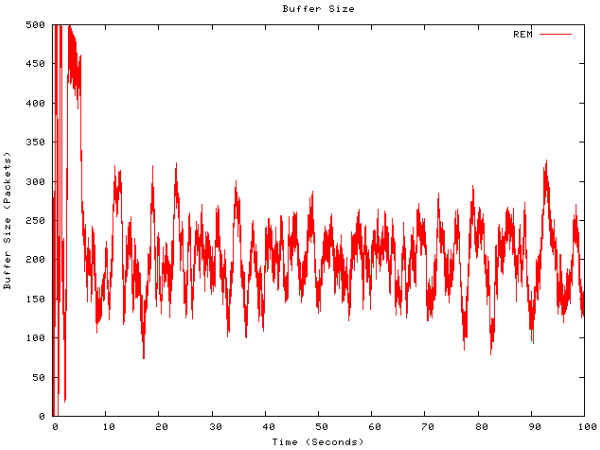
(a) FEM



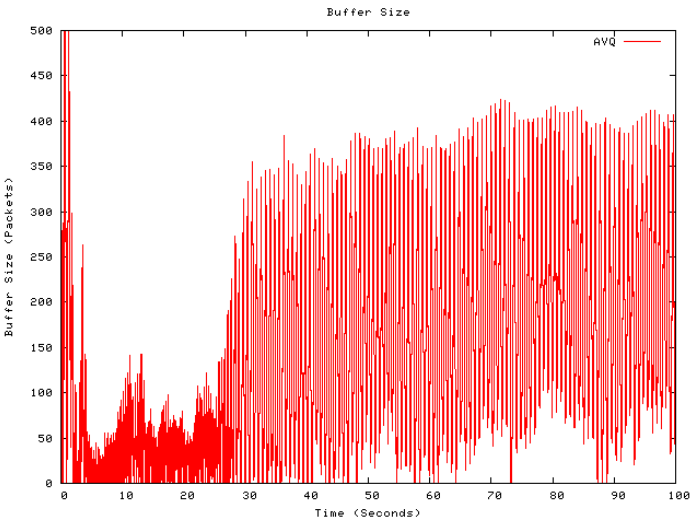
(b) PI



(c) A-RED



(d) REM

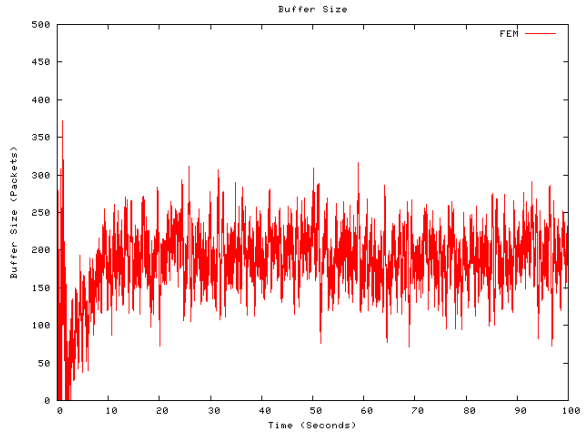


(e) AVQ

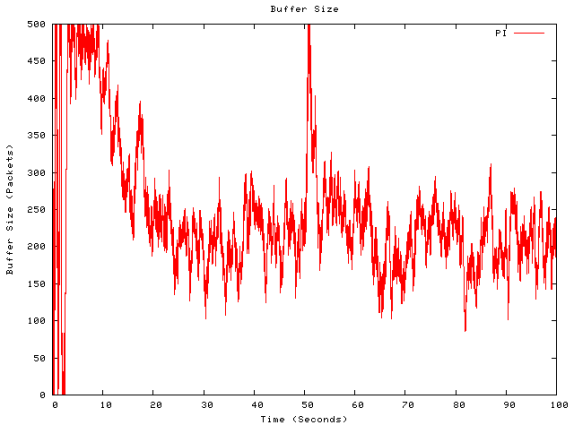
Figure 7.22 Scenario II-1: Queue lengths

7.4.1.2 Scenario II-2: Performance in the Presence of Short-lived Flows

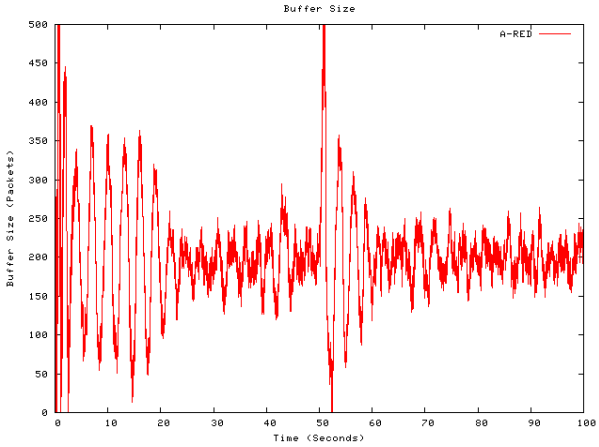
Scenario II-2 investigates the performance of AQM schemes by introducing additional web-like traffic that can be seen as noise-disturbance to the network. We keep the traffic conditions as used in the previous *Scenario II-1*, and at the middle of the simulation (i.e., *t=50 sec*) we introduce the short-lived flows, which arrive at the link at the rate of 30 flows per second (of 20 packets each), as suggested by Kunniyur and Srikant (2004). Figure 7.23 shows the queue length evolution of the AQM schemes. We can observe the robustness of the FEM controller that adequately controls the queue at the specified TQL, without noticeably being affected by the sudden introduction of short flows. It exhibits the highest utilization, with no losses, and the shortest delay variation (see Table B.4). This is in contrast with the other AQM schemes, which at *t=50sec* are influenced by the introduction of short flows. PI, and A-RED after a significant transient response with large overshoots, are slow to settle down to the reference value, REM exhibits a large overshoot at *t=50sec* that results in losses, whereas AVQ has further increased the delay variation at the queue, resulting in degraded utilization. All, except for FEM controller, have decreased the link utilization – with AVQ having the lowest value – and have further increased the loss rate.



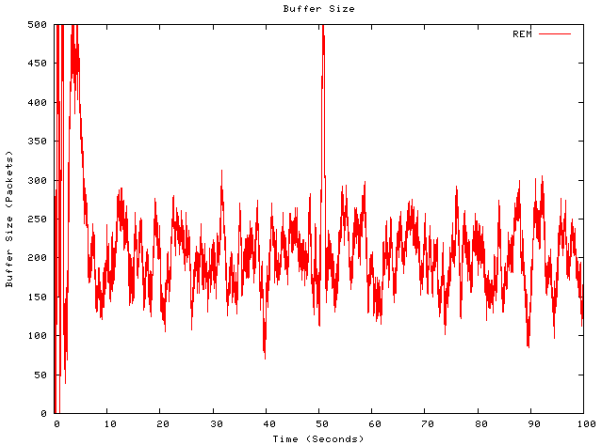
(a) FEM



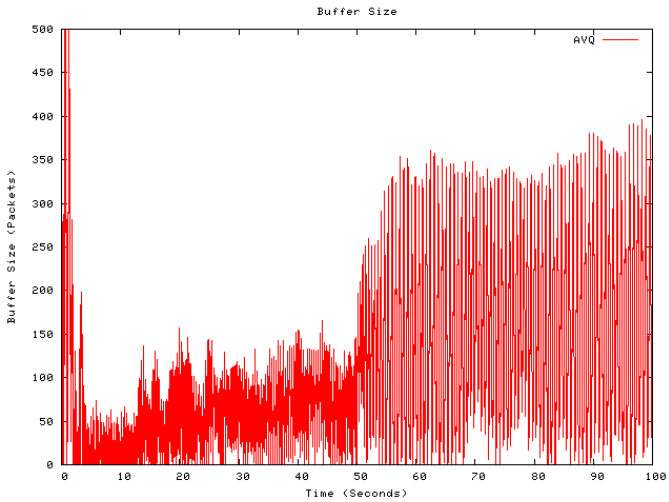
(b) PI



(c) A-RED



(d) REM



(e) AVQ

Figure 7.23 Scenario II-2: Queue lengths

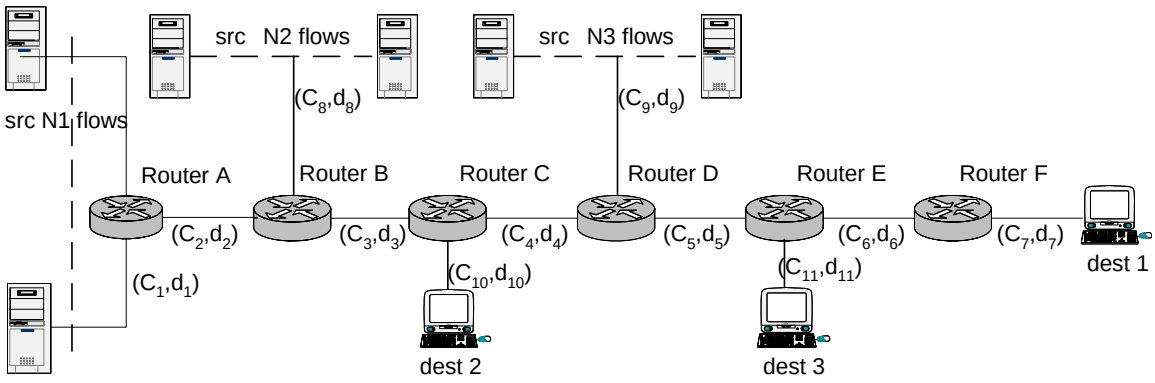


Figure 7.24 Multiple-bottleneck network topology

7.5 Multiple-bottleneck Links

The network topology of a multiple-bottleneck link is shown in Figure 7.24. We have considered network topologies with multiple bottleneck links in order to examine the performance of the AQM schemes in more realistic scenarios.

We use AQM in the queues of all core links from router-A to router-F. All other links (access links) have a simple Drop Tail queue. The link capacities and propagation delays are set as follows: (C_1, d_1) = (C_8, d_8) = (C_9, d_9) = (100Mbps, 5ms), (C_2, d_2) = (C_4, d_4) = (C_6, d_6) = (15Mbps, 10ms), (C_3, d_3) = (15Mbps, 60ms), (C_5, d_5) = (15Mbps, 30ms), and (C_7, d_7) = (C_{10}, d_{10}) = (C_{11}, d_{11}) = (200Mbps, 5ms). N_1 flows end up at destination 1, whereas N_2 flows end up at destination 2, and N_3 flows end up at destination 3 creating cross traffic. The results show that both bottleneck links, where the cross traffic exists, (i.e., between router-B and router-C, and between router-D and router-E) exhibit similar behaviour, as far as the performance comparison is concerned. Therefore, we have chosen the bottleneck link between router-D and router-E to show the results obtained.

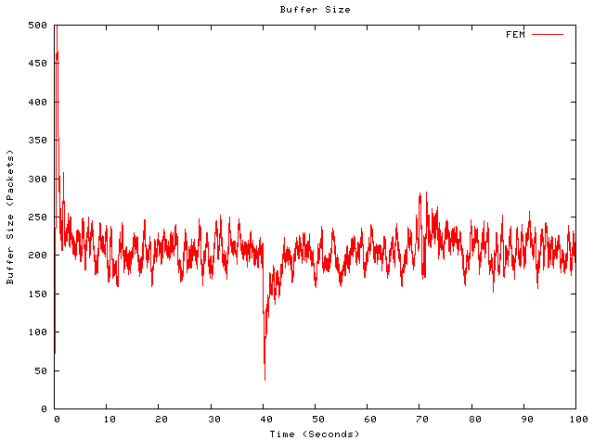
7.5.1 Scenarios III

All results (the bottleneck link utilization, the loss rate, and the mean queuing delay with its standard deviation) are summarized in Table B.5 (see Appendix B).

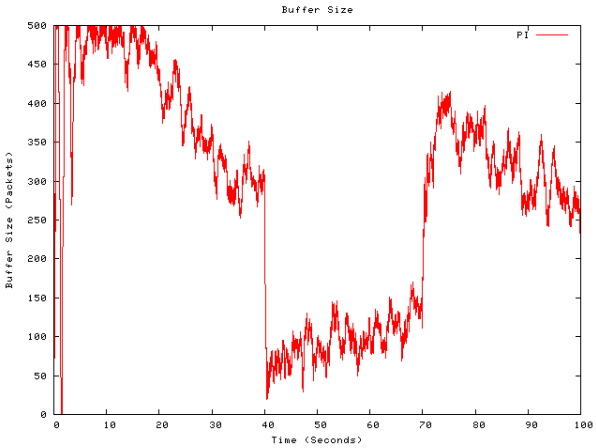
Note that we keep the control parameter values of the AQM schemes as used in Section 7.3.

7.5.1.1 Scenarios III-1-3: Effect of Traffic Load and Speed of Response

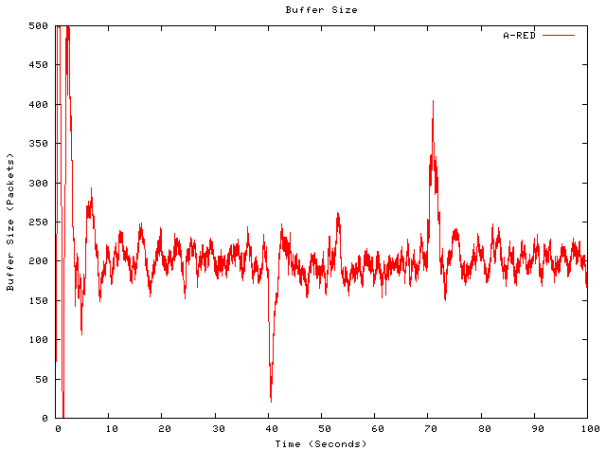
We investigate the performance of AQM schemes under comparison for dynamic traffic changes and different traffic loads. In *Scenario III-1* the number of FTP flows is $N_1 = 100$, $N_2 = 50$, and $N_3 = 100$. We also introduce time-varying dynamics on the network, by stopping half of all the flows at time $t = 40$ *sec*, and resuming transmission at $t = 70$ *sec*. In *Scenario III-2*, we have increased the N_1 flows to 500 flows, in order to examine the performance of the AQM schemes in high traffic load with time-varying dynamics and kept the other traffic conditions as in *Scenario III-1*. Finally, in *Scenario III-3*, we kept the traffic conditions as in *Scenario III-2* with an increase of the number of N_2 and N_3 flows to 100 and 200 flows, respectively. Table B.5 (see Appendix B) gives the obtained statistical results. Also, Figures 7.25-7.27 show the queue length evolutions of the AQM schemes for the three scenarios. It is clear from these figures that the FEM controller exhibits a robust behaviour in terms of both increased traffic load, as well as quick response to the time-varying dynamics that we have introduced. Under such conditions, FEM is able to maintain the queue around the specified target value. Thus FEM achieves high utilization (the highest among the others), minimal losses (the lowest among the others), and bounded delay variation (the shortest among the others). On the other hand, the other AQM schemes exhibit slow response to regulate the queue that inevitably deteriorates delay variation. AVQ, despite the fact that it always tries to keep the queue length as small as possible, shows weaknesses in responding to dynamic changes, as is exhibits sluggishness in adapting to the changing network conditions. This has as a consequence to have large variations in delay.



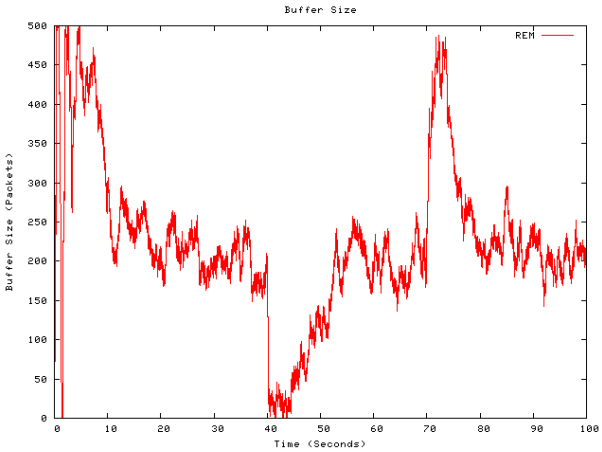
(a) FEM



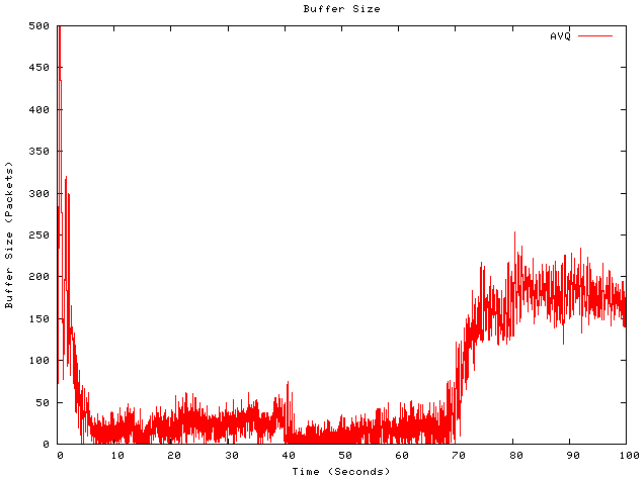
(b) PI



(c) A-RED

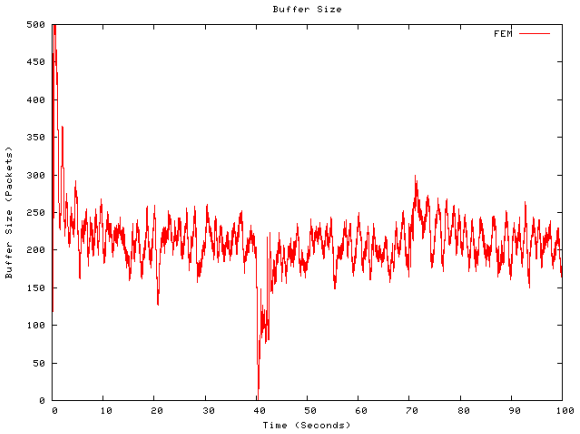


(d) REM

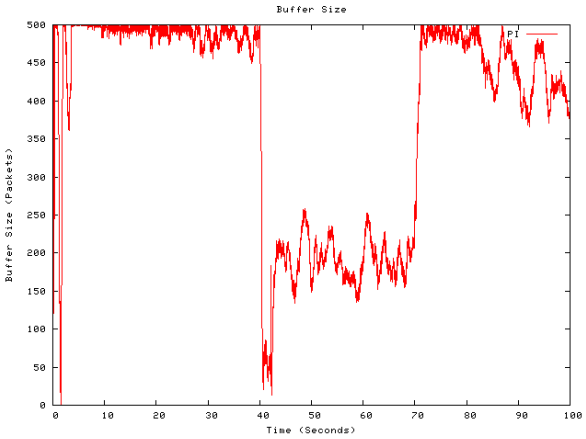


(e) AVQ

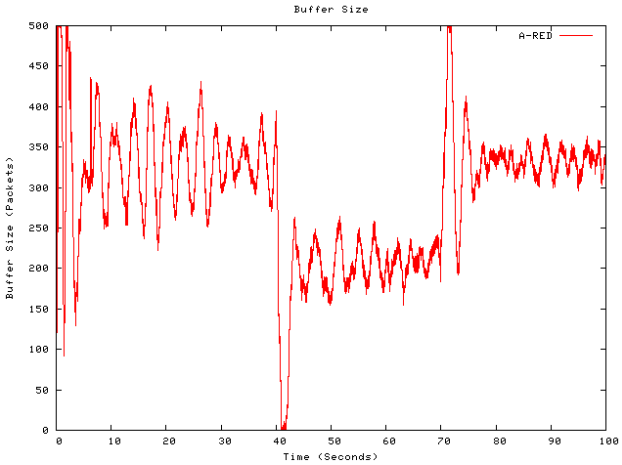
Figure 7.25 Scenario III-1: Queue lengths (for 200 flows)



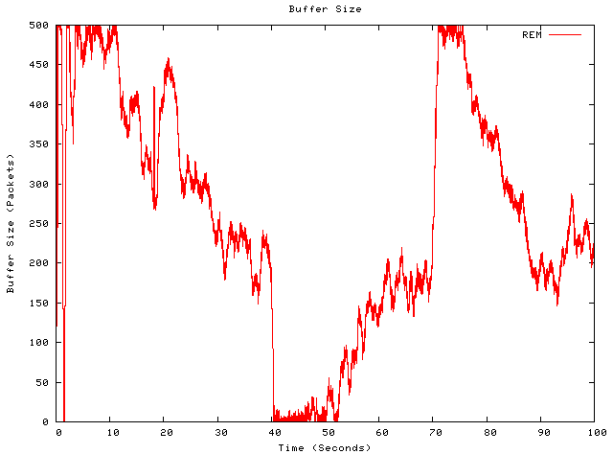
(a) FEM



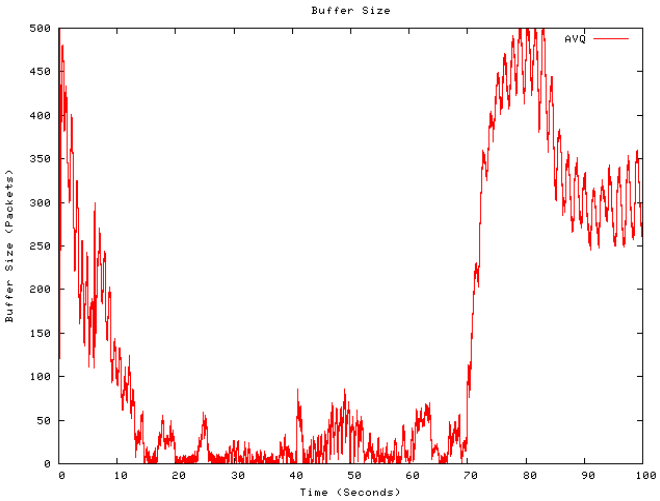
(b) PI



(c) A-RED

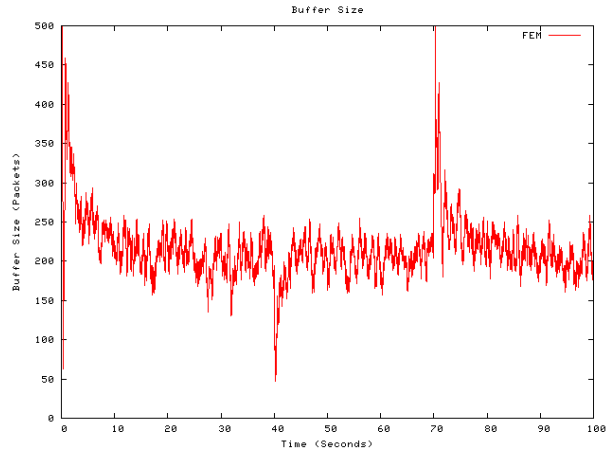


(d) REM

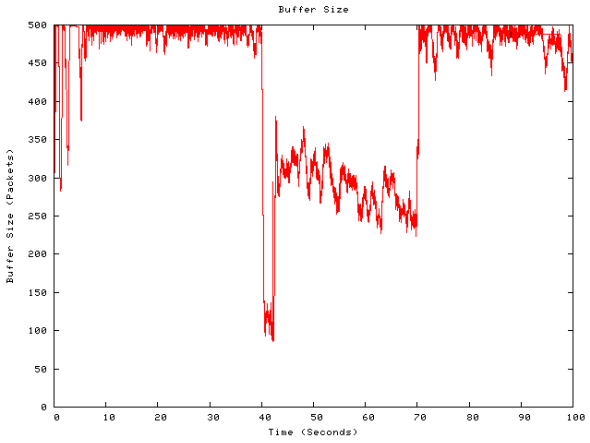


(e) AVQ

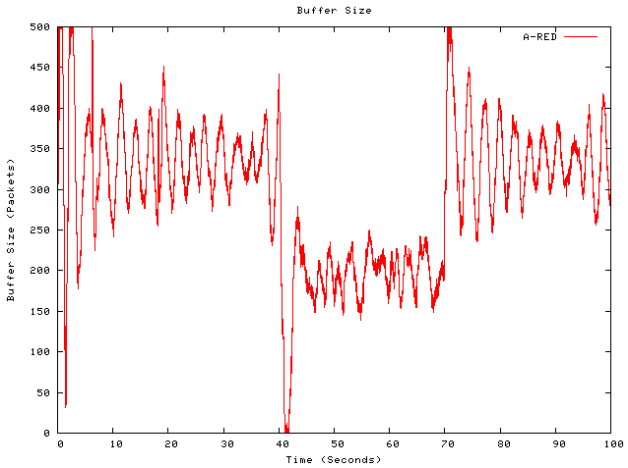
Figure 7.26 Scenario III-2: Queue lengths (for 600 flows)



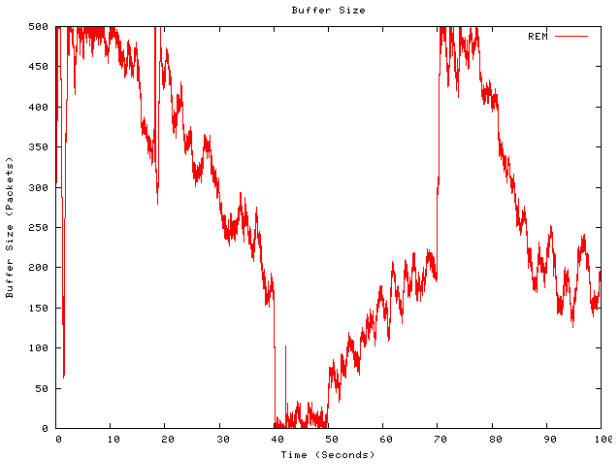
(a) FEM



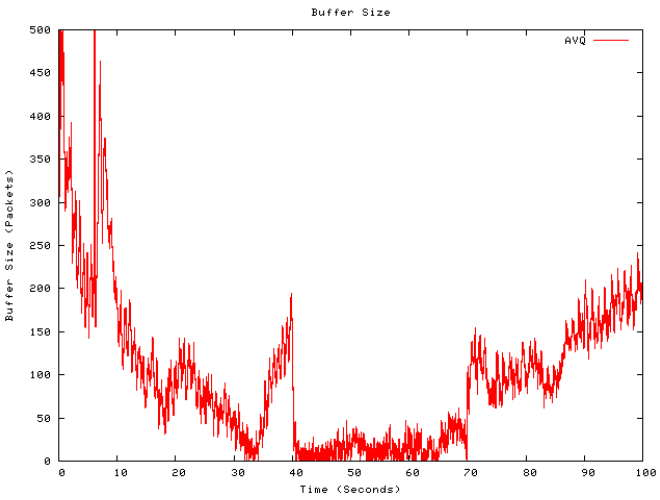
(b) PI



(c) A-RED



(d) REM



(e) AVQ

Figure 7.27 Scenario III-3: Queue lengths (for 700 flows)

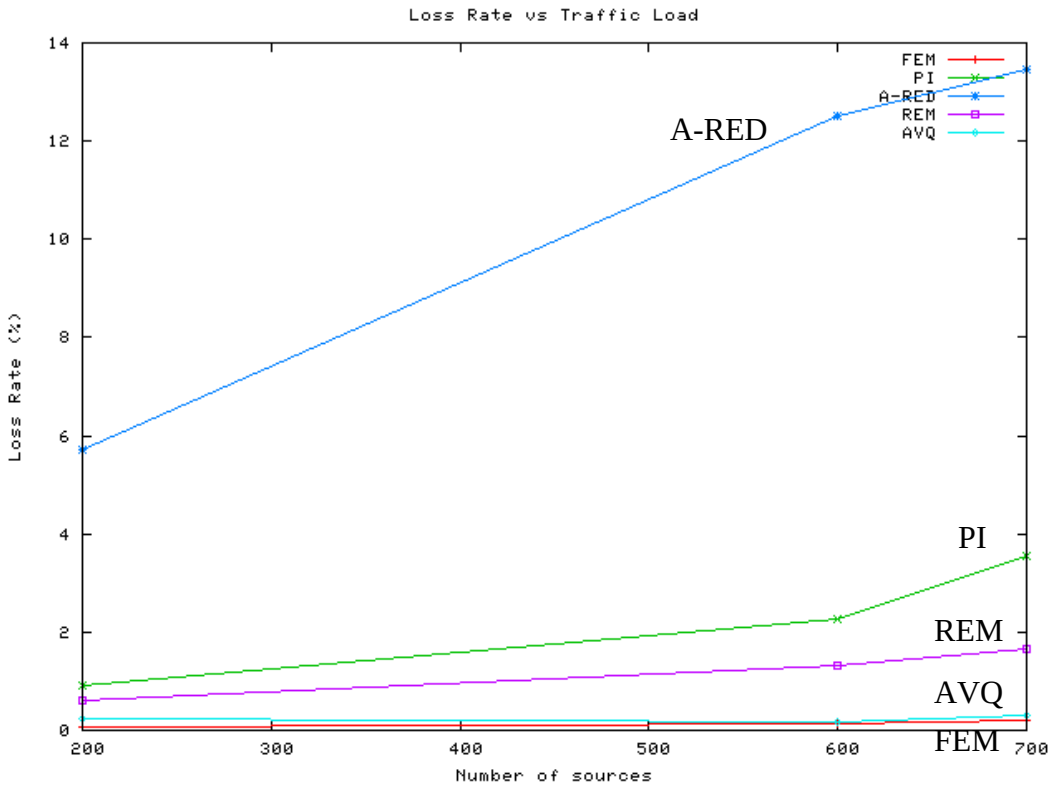


Figure 7.28 Scenario III-1-3: Loss Rate vs Traffic Load
(for 200, 600, 700 flows)

Figure 7.28 shows the loss rate as traffic increases at the bottleneck link of concern. FEM has the lowest losses, following by the AVQ scheme that is very close to FEM. A-RED has the largest loss rate with a large increase of packet loss with respect to higher loads. Figure 7.29 and Figure 7.30 show the utilization of the bottleneck link with respect to the mean queuing delay, and with respect to the delay variation, respectively. FEM outperforms the other AQM schemes on both high utilization and low delay variation, thus it exhibits a more stable, and robust behavior with a bounded delay. Note that in accordance with its design objectives, AVQ has lower delay performance, but it achieves this at the expense of a lower throughput and with high delay variation. On the other hand, FEM achieves the designed delay, as set by the reference value of 200 (TQL=200: expected mean delay = 106.67 msec). The other AQM schemes show weaknesses to maintain a robust (adequate) behavior irrespective of traffic changes, and time-varying dynamics. They achieve lower link utilization, and large queuing delays and delays variation.

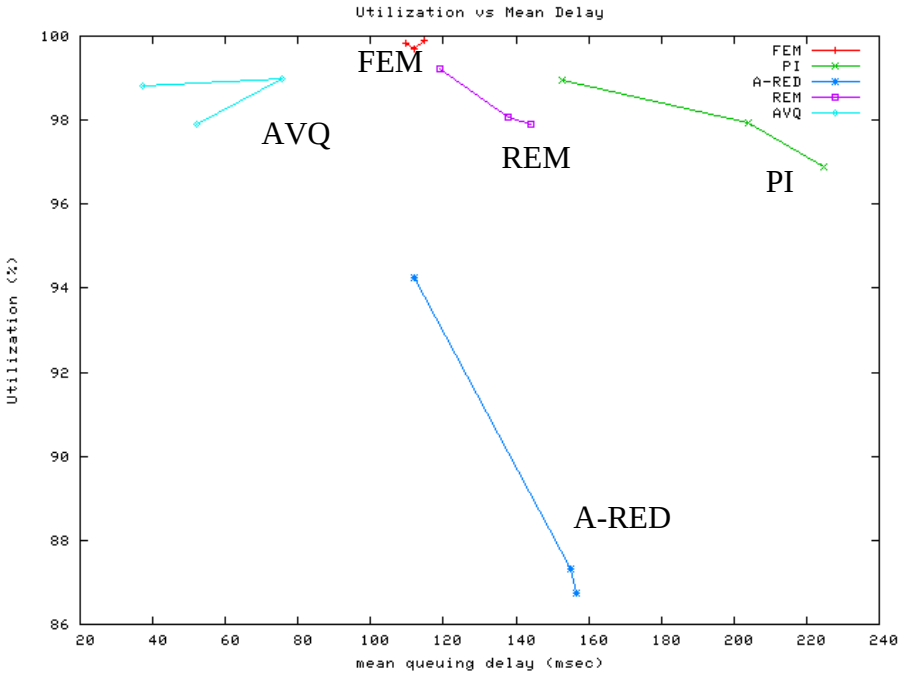


Figure 7.29 Scenario III-1-3: Utilization vs Mean Delay
(for 200, 600, 700 flows)

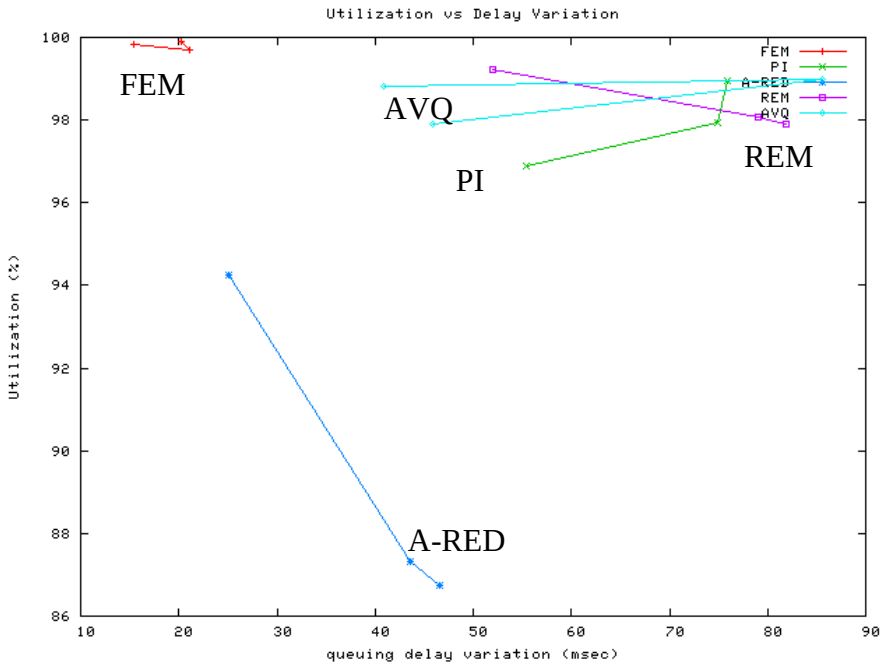
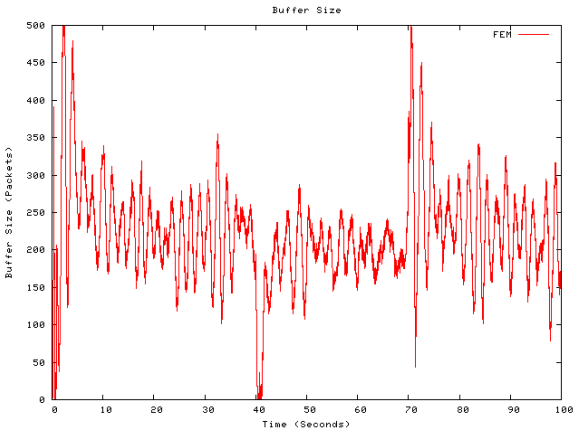


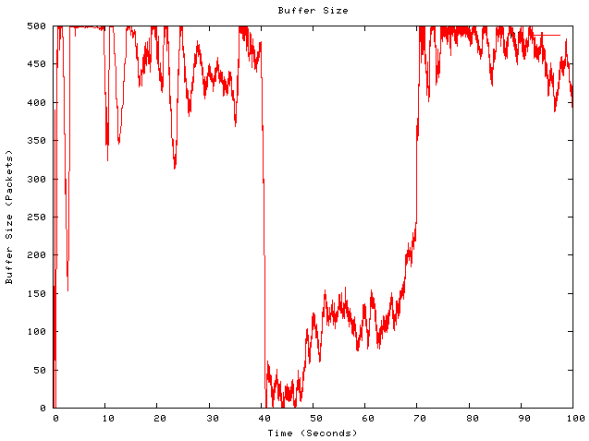
Figure 7.30 Scenario III-1-3: Utilization vs Delay Variation
(for 200, 600, 700 flows)

7.5.1.2 Scenarios III-4: Effect of Roud-Trip-Delays

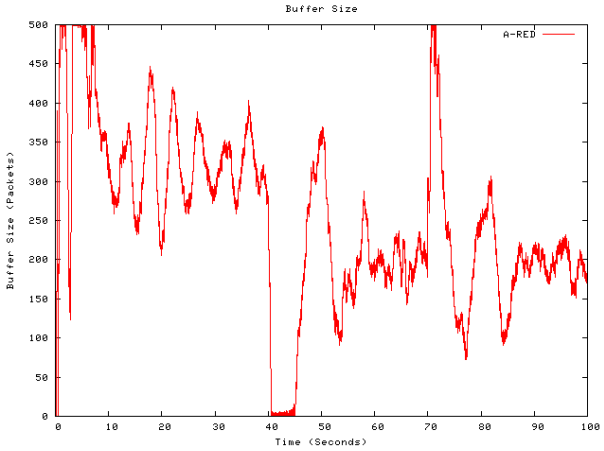
In *Scenario III-4*, we investigate the performance of AQM schemes under variation of the bottleneck link propagation delays. We specifically, take as basis the last Scenario III-3, and examine the effect of the round-trip time by increasing the propagation delay from *30 msec to 120 and 200 msec*. The results are shown in Figure 7.31 and Figure 7.32 (for 120msec *and* 200msec, respectively – note that for 30msec Figure 7.27 applies), and Table B.5. From the results, we can observe the superior steady performance of FEM with stable queue dynamics, with graceful performance degradation as the propagation delay increases up to a value of 200 msec. FEM has the highest utilization, the lowest losses and the shortest delay variation (even though for *200 msec*, FEM exhibits larger variation around the TQL than in previous situations, it still behaves much better that the other schemes as seen by Table B.5). PI, REM, A-RED and AVQ exhibit large queue fluctuations, and show weakness to react quickly to dynamic changes resulting in degraded utilization and high variance of queuing delay.



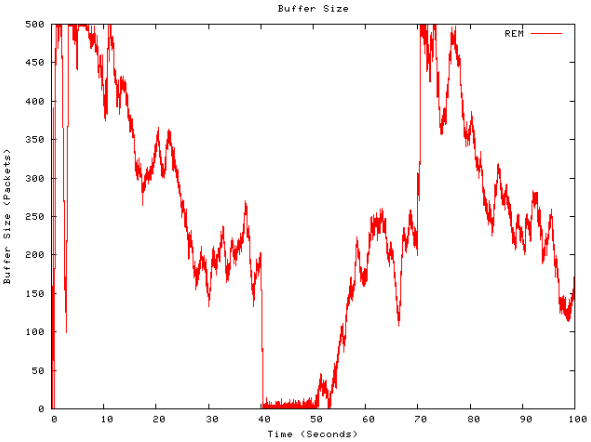
(a) FEM



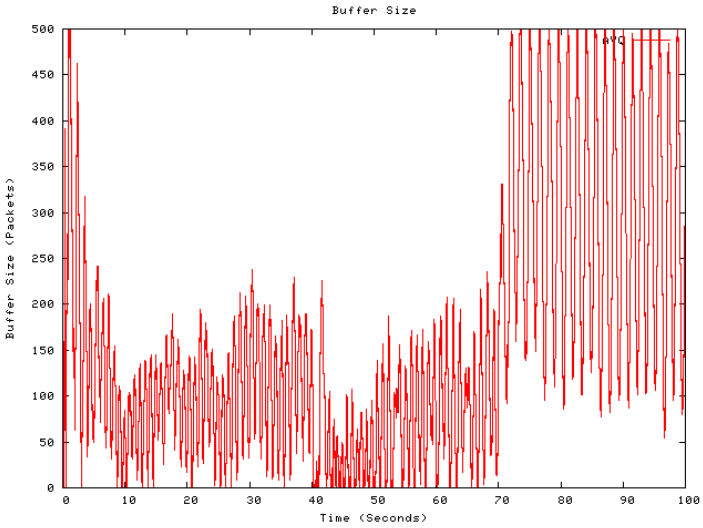
(b) PI



(c) A-RED

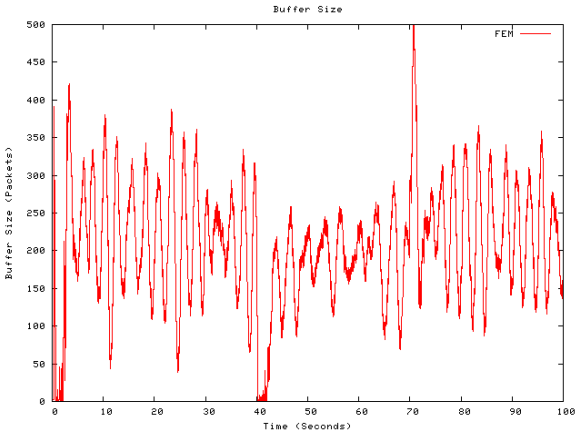


(d) REM

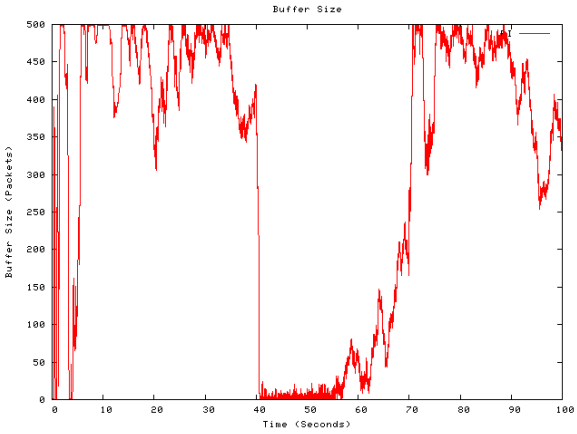


(e) AVQ

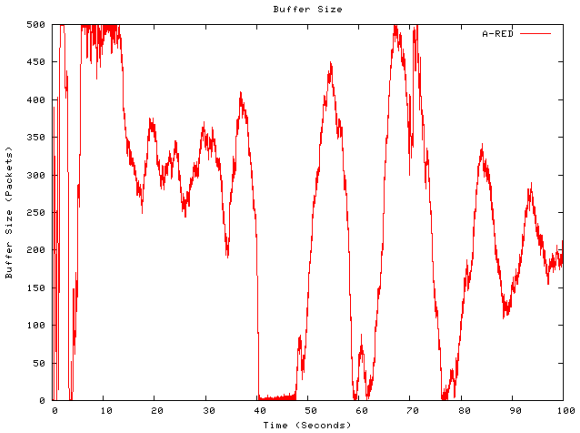
Figure 7.31 Scenario III-4: Queue lengths (for bottleneck prop. delay = 120 msec)



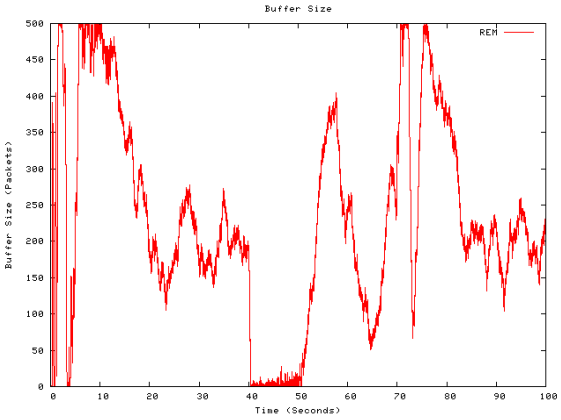
(a) FEM



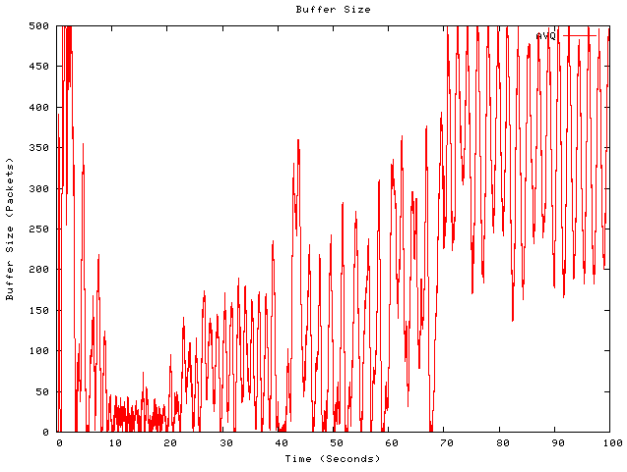
(b) PI



(c) A-RED



(d) REM



(e) AVQ

Figure 7.32 Scenario III-4: Queue lengths (for bottleneck prop. delay = 200 msec)

Thus, these mechanisms are shown to be sensitive to variations of RTT within the range of interest. This is clearly illustrated in Figure 7.33, where we show the loss rate as the propagation delay increases. FEM shows robustness by having minimal, and the lowest among the others, losses. Figure 7.34 shows the utilization of the bottleneck link with respect to the mean queuing delay, and Figure 7.35, shows the utilization with respect to the queuing delay variation. FEM outperforms the other AQMs, in managing to achieve high utilization, and at the same time regulating the queue and thus providing bounded mean delay, and delay variation.

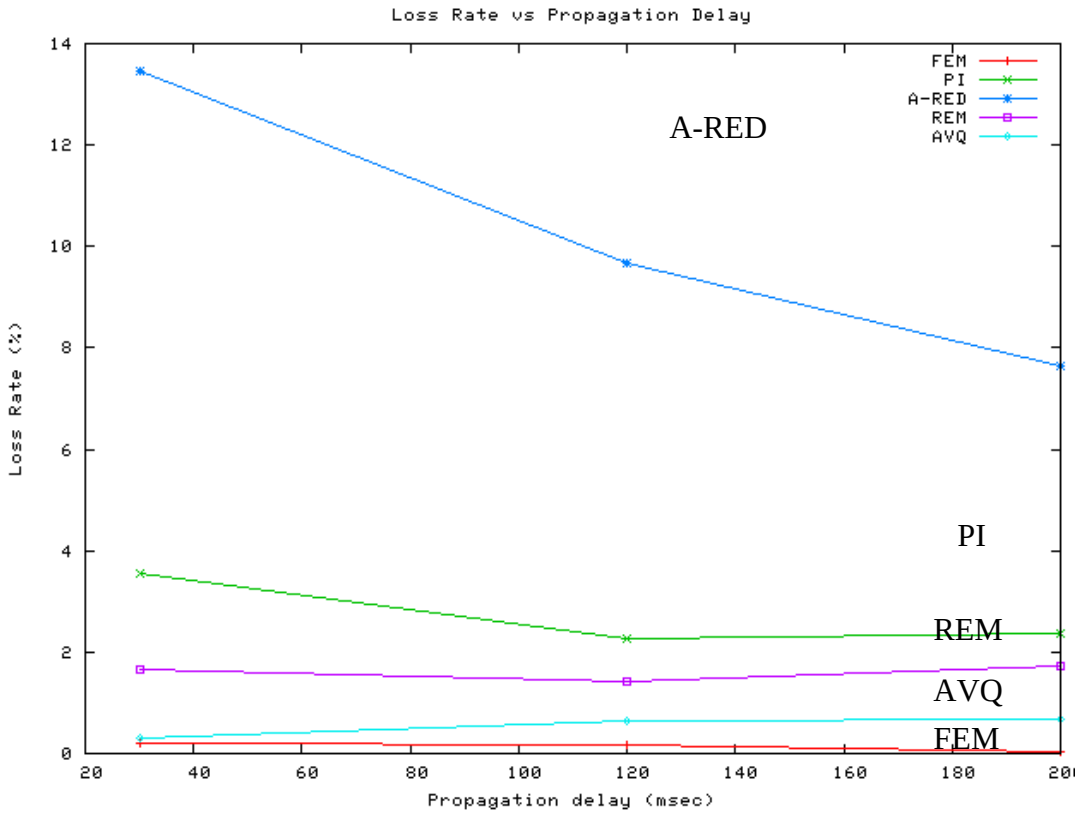


Figure 7.33 Scenario III-4: Loss Rate vs Propagation Delay
(bottleneck propagation delay varies from 30, 120, 200 msec)

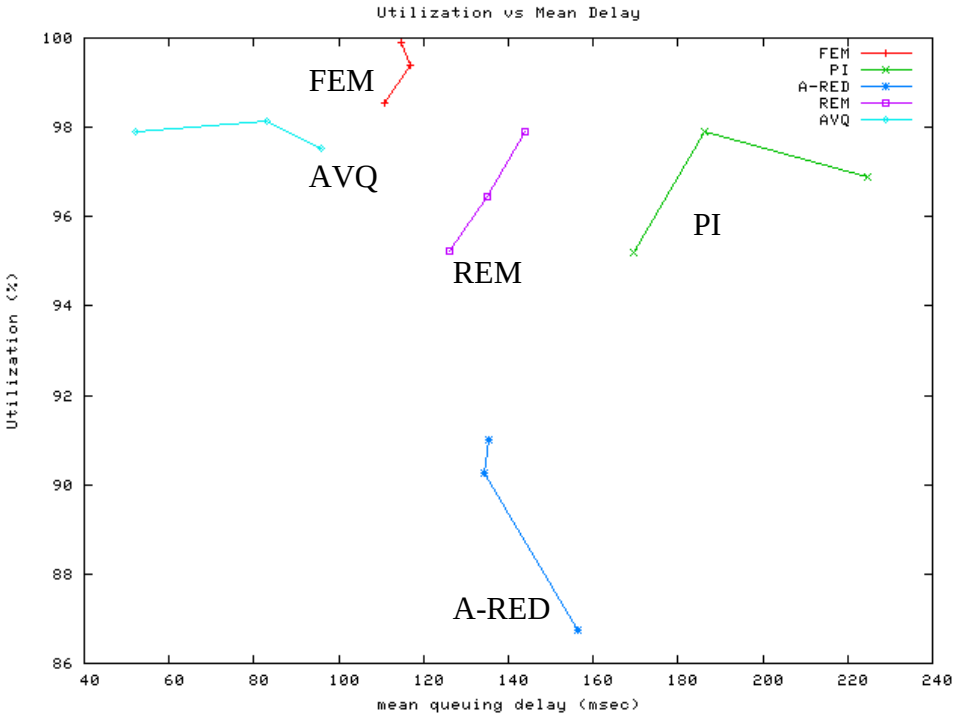


Figure 7.34 Scenario IIII-4: Utilization vs Mean Delay
(bottleneck propagation delay varies from 30, 120, 200 msec)

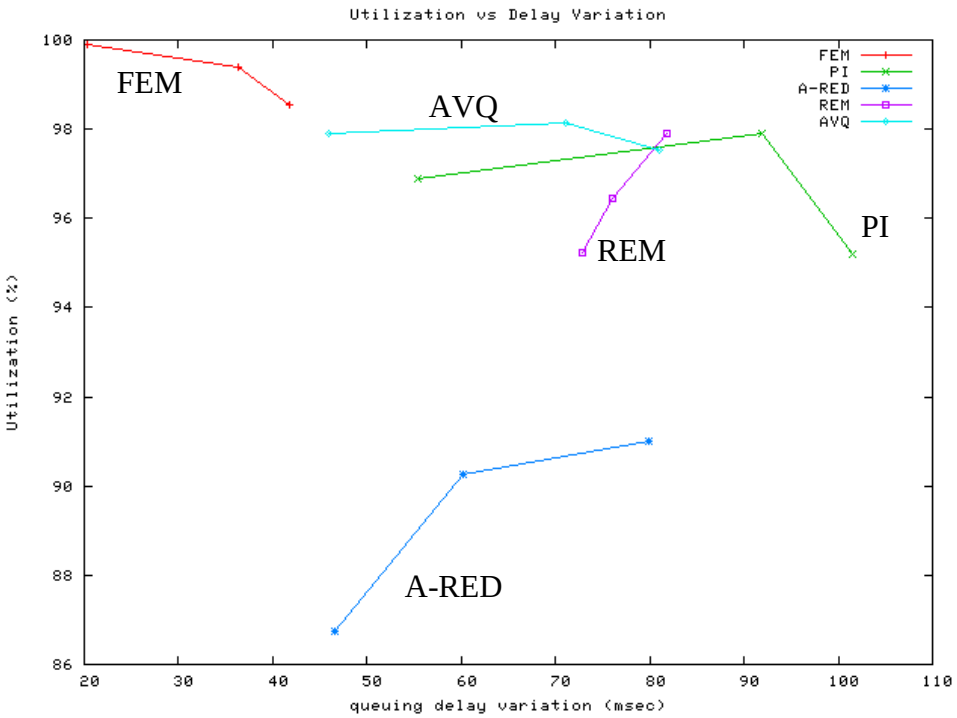
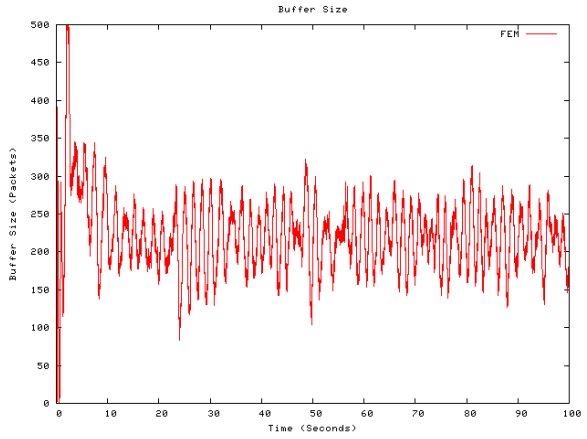


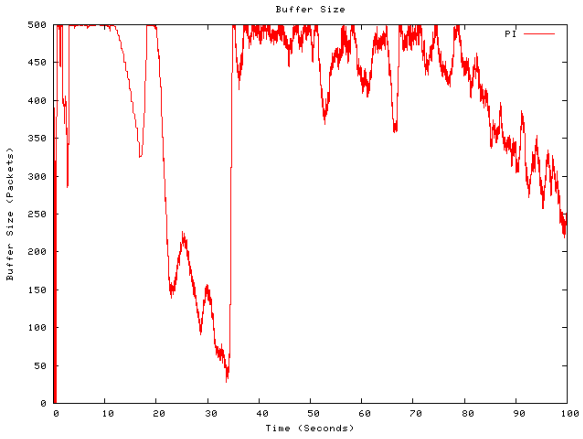
Figure 7.35 Scenario III-4: Utilization vs Delay Variation
(bottleneck propagation delay varies from 30, 120, 200 msec)

7.5.1.3 Scenario III-5: Performance in the Presence of Short-lived Flows

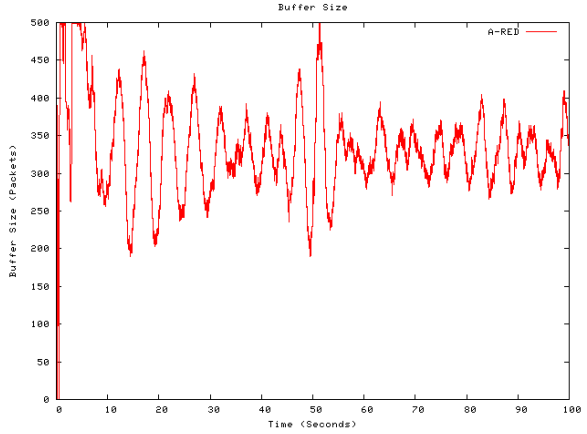
Scenario III-5 investigates the performance of AQM schemes by introducing additional web-like traffic that can be seen as noise-disturbance to the network. In particular, we use Scenario III-3, without the dynamic changes, and for a bottleneck link with propagation delay of *100 msec*. We introduce short-lived flows, as cross traffic passing through the bottleneck link, at the middle of the simulation (i.e., *t=50 sec*), which arrive at the link at the rate of 30 flows per second (of 20 packets each), as suggested by Kunniyur and Srikant (2004). Figure 7.36 shows the queue length evolution with respect to time of the AQM schemes. We can observe the robustness of the FEM controller that adequately controls the queue at close to the specified TQL, without being noticeably affected by the sudden introduction of short flows. At the same time, it exhibits the highest utilization, with the lowest losses, and the shortest delay variation (see Table B.5). This is in contrast with the other AQM schemes, which at *t=50sec* are influenced by the introduction of short flows. REM after a significant transient response with large overshoots, is slow to settle down to the reference value, whereas PI and A-RED are far away of the reference value and cannot regulate the queue. AVQ has further increased the delay variation at the queue. This poor behavior of these AQM schemes has as a result to exhibit large variation in delay, and degraded utilization, as compared with the FEM controller.



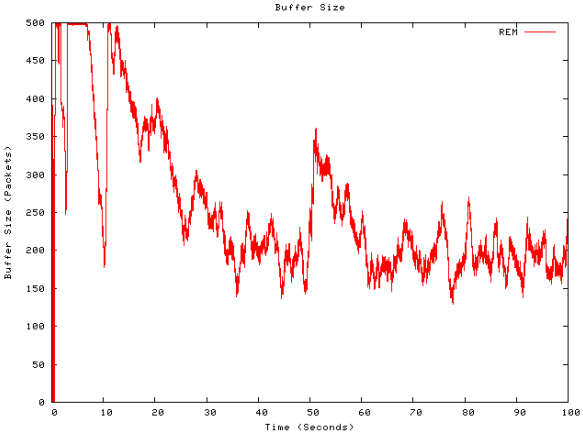
(a) FEM



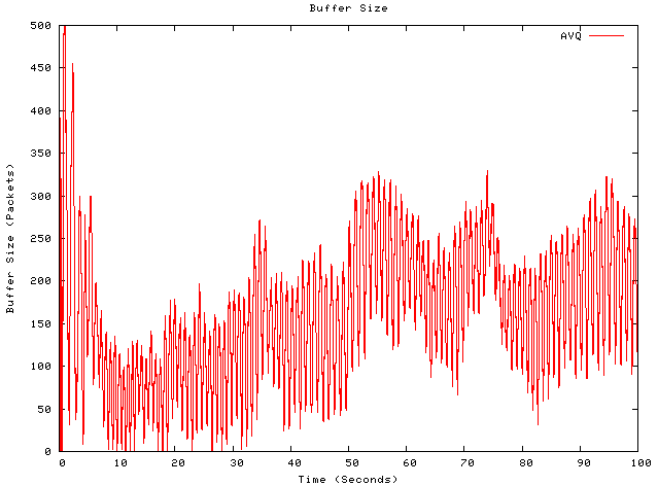
(b) PI



(c) A-RED



(d) REM



(e) AVQ

Figure 7.36 Scenario III-5: Queue lengths

7.6 Conclusions

A detailed evaluation and comparison of the proposed AQM-based nonlinear fuzzy logic controller (FEM) with well-known AQM schemes (A-RED, PI, REM, and AVQ) is conducted using the NS-2 simulator. Simulation scenarios are carefully designed in order to investigate the effectiveness and robustness of the selected schemes. Results have demonstrated that the FEM controller exhibits many desirable properties, like robustness and fast system response, and behaves better than the other AQM schemes in terms of queue fluctuations, bounded mean delay and delay variations, packet losses, and link utilization, with capabilities of adapting to highly variability and uncertainty in network.

FEM achieves better robustness and queue stabilization than other AQM schemes under dynamic environments, where the number of flows, RTT and propagation delays varies significantly, or where there are short-lived TCP flows or unresponsive UDP flows. FEM can effectively control the queue length to a given reference value with short variation in delay. It also achieves a better tradeoff between utilization and queuing delay than the AVQ scheme, since by regulating the queue around a specified value, it achieves bounded mean delay and short delay variation, and at the same time it achieves the highest link utilization among the other AQM schemes, including the AVQ mechanism.

The AVQ scheme tries to keep the queue length as small as possible while trying to achieve a target (in our study, full) utilization. However, it is shown that FEM outperforms AVQ in terms of utilization. Moreover, the queue length of AVQ cannot be controlled to a target value, as it does not have any explicit control for queue length, and consequently to control jitter; thus it shows much variation in the queue when time-varying dynamics exist, and it exhibits sluggishness in adapting to the changing network conditions. Further, it has queue properties dependant on traffic loads. Hence it is shown that AVQ is not as robust as FEM is demonstrated to be. With the introduction of short-lived flows and unresponsive flows, AVQ is shown to perform poorly in terms of link utilization (recall that the main goal of AVQ is to achieve a target – high – utilization).

A-RED is sensitive to the level of network load, and dynamic changes; the queue length usually oscillates significantly; Since A-RED controls the macroscopic behavior of the queue length (average), it often causes sluggish response and fluctuation in the instantaneous queue length. As a result, an important variation in delay is observed, and very high loss rates are exhibited.

PI and REM suffer from a long response time under heavy congestion. When network conditions are changed dynamically, and/or short-lived flows are introduced, in PI and REM, there is a much longer response time before arriving at a new steady state, which causes large queue deviation and lower utilization, with significant amount of loss rate.

Overall, we have demonstrated the effectiveness and robustness of the proposed methodology with simulative evaluation, and at the same time we demonstrated limitations the other AQM schemes have. Thus, from the results presented, the fuzzy logic AQM-based control methodology offers significant improvements on controlling congestion in TCP/IP networks under widely differing operating conditions. It is worth pointing out that the settings of FEM, as well as of the other controllers, remain unchanged (i.e. they are not retuned) throughout all simulation scenarios.

Chapter 8

Fuzzy Explicit Marking In/Out (FIO): An Intelligent Nonlinear Fuzzy Logic-based Control Methodology in TCP/IP Diff-Serv Networks

8.1 Introduction

Congestion control at the Diff-Serv core requires AQM schemes to preferentially drop/mark packets based on the level of precedence they belong to, by giving priority to low drop precedence against high drop precedence traffic. At the same time, it is significant to provide adequate QoS-centric performance objectives, in terms of bounded delays, with high link utilization and minimal losses in overall. The existing AQM schemes for Diff-serv congestion control are identified to show weaknesses to respond to such objectives. Thereafter, we build on the fuzzy controller we designed for best-effort service and investigate its extension and suitability to provide effective congestion for Diff-Serv environments.

8.2 The Need for the Alternative

The Diff-Serv architecture (Braden et al., 1998) proposes a scalable means to deliver IP QoS based on handling of traffic aggregates. The focus is on the

forwarding path behaviour required in routers[*]. AQM mechanisms are needed at the core of the Diff-Serv domain to provide differentiated services aware congestion control algorithms, and more precisely to provide differential dropping algorithms for network core routers by preferentially drop/mark non-conforming (high drop precedence) against conforming (low drop precedence) packets during periods of congestion. By doing so, a differential treatment, that is, a differentiation between high- and low-priority traffic aggregates, is provided.

An advantage of the aggregated behavior of Diff-Serv is its simplicity and scalability. For example, the preferential dropping scheme adopted in routers in the network core is not likely to change over time due to new applications with new service requirements in the future. Since the characteristics of a service are defined and captured by its corresponding traffic profile, it is only necessary to create this profile at the edge of the Diff-Serv network, while the mechanisms at the core remain unaffected. A disadvantage due to the aggregated treatment is that Quality of Service cannot be guaranteed for individual users, but it is differentially provided. To provide aggregated QoS effective mechanism are required.

Recently, a number of AQM mechanisms have been proposed (e.g., Clark & Fang, 1998; Chait et al., 2002) within the framework of Diff-Serv architecture to provide differential *marking* probabilities at the core, aiming at throughput differentiation to different traffic aggregates. However, these mechanisms show weaknesses to detect and control congestion under dynamic traffic changes, and a slow response to regulate queues (Chrysostomou et al.). The inability of providing, in most cases, a bounded delay on the queue as a whole, has a negative impact on the overall performance regarding delays, losses and link utilization.

Therefore, there is a need of an effective AQM scheme, implemented at the core routers of the Diff-Serv domain, addressing utilization and QoS issues, like high throughput, low losses, and bounded delays, combined with an adequate differentiation among different traffic levels of priority.

[*] In this thesis we focus on the core aspects of Diff-Serv, and assume that appropriate edge policies for Diff-Serv are in place.

Hence, the main objective is to investigate the *alternative* of using fuzzy control approach for Diff-Serv AQM, based on the ability of fuzzy logic control to provide simple and effective solutions on controlling nonlinear, time-varying systems. We also investigate whether we can adopt the same controller construction as for the best-effort service (see Chapter 5 and 6 for related discussion). Thus, the problem we solve is the following: *formulate and evaluate an intelligent TCP/AQM control methodology implemented at the Diff-Serv core, which performs adequately over a wide variety of network/traffic conditions, capturing the nonlinearities and dynamics of the process itself.* By *performing adequately*, we mean to provide an adequate differentiation among different priorities of traffic aggregates, and at the same time to provide bounded delays, with high link utilization and low losses, thus addressing QoS issues in TCP/IP Diff-Serv environments.

Summarising our earlier assertions for the use of Fuzzy Logic in best-effort, the use of fuzzy logic can provide us with a nonlinear mapping of the inputs and output of the controller, which does not require knowledge of dynamic network parameters. Thus, an effective and robust AQM system at Diff-Serv core can be designed to drive quickly the controlled system into the steady-state. This is contrasted with the linear counterpart that itself is not robust enough to capture the dynamics and nonlinearities of TCP/IP networks. Existing Diff-Serv AQM mechanisms (e.g., Clark & Fang, 1998; Chait et al., 2002) fail to achieve such an important requirement, due to the linearity of the control law they obey and the strong dependency of their control parameters on network/traffic parameters (like the number of flows and RTTs).

8.3 Diff-Serv Fuzzy Logic Control Methodology Design Goals

Our focus is on the queue management disciplines required to implement the differential treatment an individual packet receives at the core of a Diff-Serv-compliant domain. In particular, we are interested for the employment of such a methodology of queuing disciplines for the Assured Forwarding PHB, since this kind

of PHB allows differential treatment of packets by the use of buffer management (Heinamen, Baker, Weiss, & Wroclawski, 1999). Thus, we focus on services built on top of the AF PHB (see related discussion in Chapter 4).

The main design goal is, in case of congestion, to protect packets with a lower drop precedence value from being lost by preferentially discarding packets with a higher drop precedence value; thus differentiating traffic aggregates with different drop precedence levels. At the same time, we aim to achieve bounded delays by stabilizing the queue variations around pre-defined targets (which are different for each priority class) that indirectly provide delay assurances, while also ensuring high link utilization and low losses; thus, offering (differentiated) QoS to traffic aggregates.

Summarizing from Chapter 6, the design goals of the nonlinear fuzzy logic control methodology proposed for effective AQM are the following:

- Dynamic and effective fast system response with robustness to the time-varying, dynamic nature of the controlled system
- High link utilization (based on the useful throughput)
- Minimal packet losses
- Bounded-regulated queue fluctuations and delays (mean and variation).

These goals have been utilized for providing congestion control in TCP/IP best-effort networks (see Chapter 6). Since these goals are QoS-based, they are also utilized for providing QoS assurances in the Internet, albeit aggregated for Diff-Serv.

Thus, the *nonlinear fuzzy logic-based control methodology* proposed in Chapter 6 is adopted for Diff-Serv AQM to operate on the core routers' buffer queue. The goals are to achieve *differentiated treatment of traffic aggregates*, ensuring at the same time *bounded queuing delays*, *low losses*, and *high link utilization* in overall; hence offering (differentiated) QoS in traffic aggregates. In accomplishing these goals, *fast system response* with *robustness* to the time-varying dynamic nature of the controlled

system play a significant role and hence are important design requirements. Furthermore, low complexity is also sought.

8.4 Fuzzy Explicit Marking In/Out System Model

The proposed fuzzy logic control methodology in TCP/IP Diff-Serv networks provides a nonlinear probability function for a differentiated *mark* treatment during congestion to support different levels of precedence.

Particularly, a *two-level*[†] *of precedence* FEM controller (see Figure 8.1) is designed to operate on the core routers' buffer queue, called *Fuzzy Explicit Marking In/Out* (FIO), where "In" and "Out" terms are used to distinguish packets that are classified into different precedence traffic aggregates, distinguished by the *drop/mark precedence level* they belong to. "In" packets belong to the low drop/mark precedence (i.e., high-priority traffic), while the "Out" packets belong to the high drop/mark precedence (i.e., low-priority traffic).

Both high- and low-priority traffic aggregates share a FIO queue. FIO comprises of two identical FEM controllers (as described in Chapter 6), one for each traffic aggregate, and we introduce two different TQLs, on the *total queue length*, one for each FEM controller. The TQL for low-priority traffic is lower than the one for high-priority traffic. Therefore, low-priority packets are more likely to be *marked* than the high-priority ones.

The idea behind this is to regulate the queue at the lower TQL. In this case, the *mark* probability of the high-priority traffic is closer to zero, as the TQL is set higher and thus it is less likely that high priority packets will be marked. In the presence of a small amount of high-priority packets, the queue would be mostly regulated at the lower TQL and thus marking of high-priority packets would be less likely. If however, the high-priority traffic is high in comparison to the low-priority traffic,

[†] Extension to multiple-levels of precedence are trivial, since the same controller is implemented for each level, but with a different target queue length reference setting, as described later. By selecting appropriate target queue references adequate differentiation between the multiple levels can be provided.

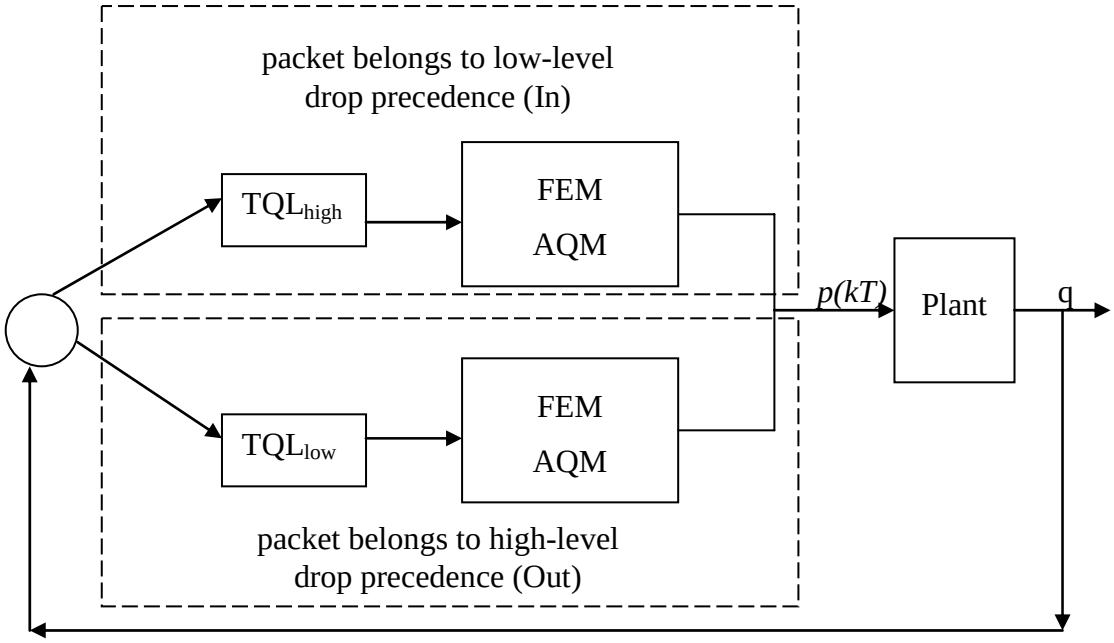


Figure 8.1 FIO system model

then the queue is regulated at the higher TQL (as there is not enough low priority traffic to ensure the lower TQL is maintained). In this case the *mark* probability for low-priority traffic is closer to one. In either case, the lower-priority traffic is marked at a higher rate. Therefore, we can accomplish both differentiation as well as a *bounded delay*, by *regulating the queue between the two TQLs*, depending on the dynamic network traffic conditions.

It is therefore expected that FIO[‡] can achieve an adequate differentiation between the two precedence traffic aggregates in the presence of congestion, by preferentially *marking* the lowest-priority packets, and giving priority/preference to high-priority-tagged traffic, while controlling the queue at the predefined levels, and thus providing QoS assurances for delay, loss, and link utilization.

[‡] This is not a unique feature of FIO. Any scheme with two regulators operating on two different Target Queue Lengths (reference setpoints) can potentially achieve differentiation in the sense we described above.

8.5 Advantages of FIO over Existing Schemes

The FIO controller can maintain short queue length (around the lowest TQL) or bounded queue, between low and high TQL, and high throughput with low losses, depending on the dynamic network traffic conditions. The FIO controller, at the same time, can offer adequate differentiation among different drop precedence traffic. It can achieve a bounded delay at the buffer based on the fact that the overall-total buffer occupancy is considered as the basis for each decision to enqueue, drop or mark a packet, and not the virtual queue of a particular precedence level, as it is the case of RIO (Clark & Fang, 1998) for "In" packets. We consider this as a drawback for RIO, if we want to have a bounded delay for the queue as a whole and under any congestion level.

In particular, RIO cannot adequately control the queue size, since its control laws for "In" and "Out" are not related to each other. The drop probability of "In" packets is not correlated with the total buffer size, in contrast to "Out" packets, thus, it fails to regulate the queue length, and consequently the queuing delay of the router's output buffer, to any targeted predefined values. This is because when excess packets arrive at the router, and congestion begins, the "In" TQL[§] is unlikely to be achieved, since the "In" drop probability only considers the "In" average queue length[**], whereas the "In" TQL is set by the network administrator over the total buffer occupancy[††]. Thus it would be more appropriate to correlate the "In" drop probability with the total average queue size, in order to achieve the specified TQL during periods of congestion. This has a negative impact to link utilization, as well as to packet loss, while introducing unwanted large queue oscillations with large delay variation.

Further, the nonlinear probability function that the FIO controller follows has as a consequence to deal the dynamics and the nonlinearities of the network, providing a

[§] The TQL for RIO is indirectly set by the network operator by the min and max thresholds for "In" packets.

[**] That is, the possibility of dropping/marking "In" packets depends on the buffer occupancy of "In" packets.

[††] In other words, the "In" average queue size, used in the calculation of the drop/mark probability, is compared with the "In" queue thresholds on the total average queue length.

dynamic and effective fast system response. On the other hand, existing AQM schemes for Diff-Serv control (like RED implementation - RIO, and the two-level PI controller by Chait et al. (2002)), retain the weaknesses and limitations identified in Chapter 3. RIO retains RED's basic linear structure, and the two-level PI controller also retains the undesired PI behaviour (e.g., the dependency of PI control parameters on dynamic network parameters).

Hence, the implementation of the FIO controller at the Diff-Serv core is expected to offer significant improvement over the existing AQM schemes, providing adequate (differentiated) QoS.

8.6 Illustrative Examples of FIO Operation

The properties of the FIO controller are illustrated using some indicative examples. We use the case of a *single-bottleneck* network topology – shown in Figure 7.1 (see Chapter 7) – with the following setup: The sampling interval of FIO is fixed at 0.*006 sec*, the TQL for high-priority (low drop precedence) traffic is set to 200 packets, whereas the TQL for low-priority (high drop precedence) traffic is set to 100 packets, for a buffer size of 500 packets. The maximum drop/mark probability for high-priority traffic is set to 0.02, whereas for the low-priority traffic is set to 0.1. We use TCP/FTP traffic for 100 flows, and we assume that the classification is already done at the Diff-Serv boundary nodes, and that the traffic profiles are based on source-destination IP pairs. The link capacities and propagation delays are set as follows: (C_1, d_1) = (100Mbps, 5ms), (C_2, d_2) = (15Mbps, 120ms), which is the bottleneck link, and (C_3, d_3) = (200Mbps, 5ms). The simulation time is set to *100 sec*.

Table 8.1 Summary of statistical results – Illustrative examples of FIO operation

TQL=100: expected mean delay = 53.33 msec

TQL=200: expected mean delay = 106.67 msec

Scenarios	**Utilization (%)**			**Loss Rate (%)**			**Delay (ms)**	
	Low-Priority	High-Priority	Total	Low-Priority	High-Priority	Total	Mean-Delay	Std-Deviation
(a)	49.69	49.33	99.02	0.47	0.019	0.25	62.61	23.05
(b)	14.42	84.6	99.02	1.285	0.084	0.26	84.27	28.59
(c)	48.78	46.58	95.36	0.39	0.15	0.27	84.55	34.93
(d)	0.28	99.07	99.35	1.8	0.41	0.41	113.74	25.12

We have conducted four experiments. The main statistical results of mean queuing delay and its standard deviation, the utilization of the bottleneck link (regarding the useful throughput), and the loss rate are shown in Table 8.1. Figure 8.2 shows the queue length evolution for all experiments.

Scenario (a) considers a limited number of flows tagged as high-priority traffic; 2 out of 100 flows are considered belonging to high-priority, whereas the rest, 98 flows, are tagged as low-priority. From Figure 8.2(a) we can observe that FIO regulates its queue to the lower TQL (100 packets). Furthermore, FIO achieves an adequate differentiation between the two traffic classes (see Table 8.1), even though the number of high-priority flows is very small compared to the low-priority.

Scenario (b) increases the number of flows tagged as high-priority traffic class to 10. FIO accomplishes a bounded queuing delay, between the two TQLs, that results in high link utilization and minimal losses (see Figure 8.2(b) and Table 8.1). Furthermore, FIO achieves a high differentiation between the two traffic aggregates, thus can provide adequate QoS.

Scenario (c) examines the behavior of the FIO AQM scheme under dynamic traffic changes. We use the previous experiment, and provide some time-varying

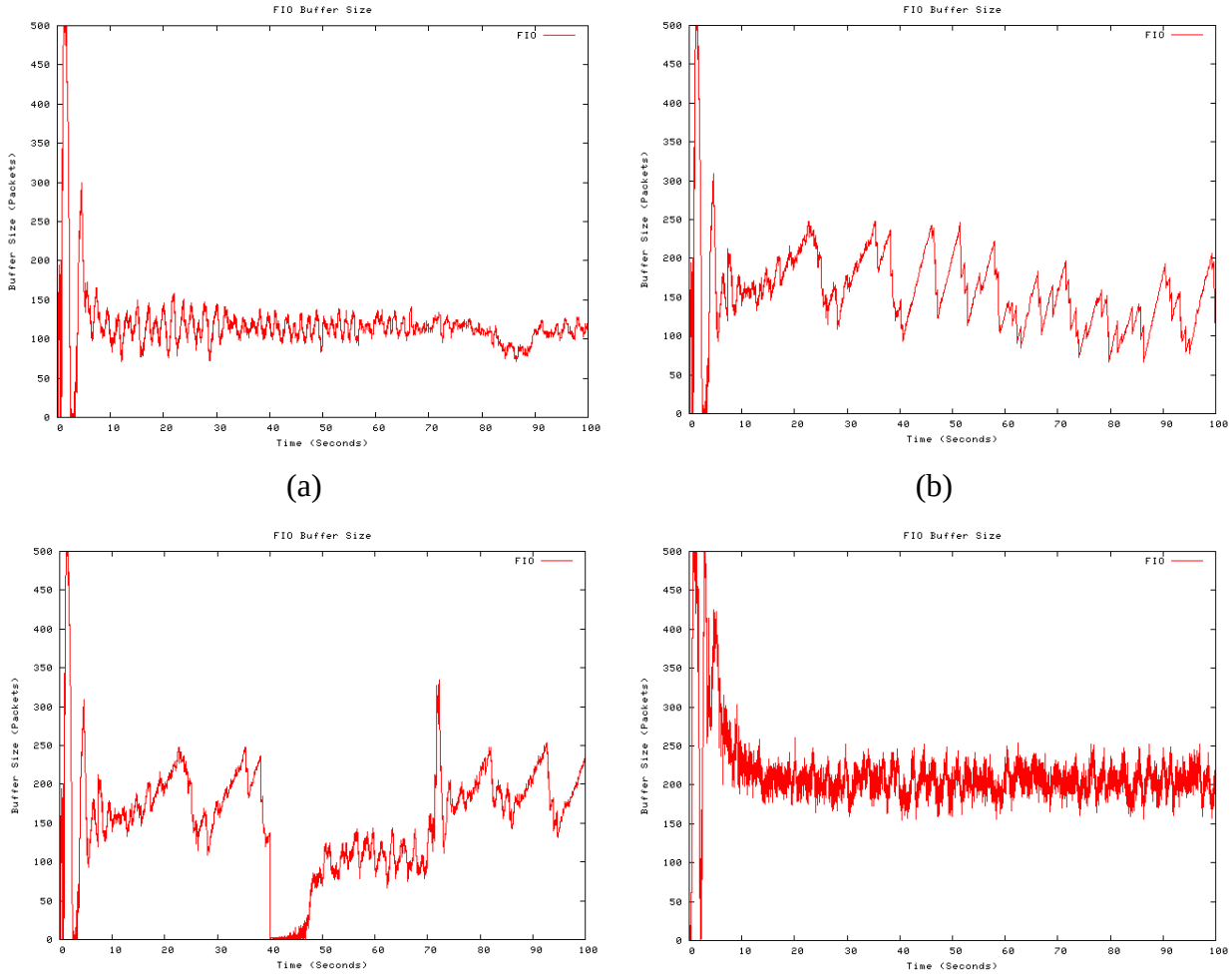


Figure 8.2 FIO queue evolution
(Illustrative examples of FIO operation)

dynamics by stopping the high-priority-tagged flows at time *t = 40 sec*, and resuming transmission at time *t = 70 sec*. The results (see Figure 8.2(c)) show that FIO is very robust against the dynamic traffic changes and keeps very good response. Between *t = 40 – 70 sec*, where only low-priority-tagged flows are active, FIO successfully manage to regulate the queue length at the TQL for low-priority.

Scenario (d) increases the number of flows tagged as high-priority traffic to 90, and also examines the effect of the RTT by having heterogeneous propagation delays of the links between the sources and their first-hop router (we separate the 100 flows

into groups of 10, and for each group - that consists of 9 high-priority-tagged flows and 1 low-priority-tagged flow – its propagation delay is increased by *5 msec*, starting from *5 msec* up to *50 msec*). The propagation delay of the bottleneck link has also changed to *60 msec*. In the presence of large amount of high-priority, compared with the low-priority traffic, FIO regulates its queue at the higher TQL (see Figure 8.2(d)) that results in high link utilization with minimal losses.

8.7 Ease of Implementation - Flexibility

The Diff-Serv architecture operates on the premise that complicated functionality should be moved at the edge of the network with simple functionality at the core. This requirement is met by the proposed fuzzy control methodology, as this is designed to be effective and simple at the same time (see practicability issues of fuzzy logic control methodology clearly explained in Chapter 6).

Note also that in a real implementation due to the use of identical FEM controllers only one fuzzy inference process is really required to operate at the router's output buffer. At each sampling period, depending on the type of the packet arrived, that is, whether it belongs to the high- or low-priority, a new/updated *mark* probability is computed, from the generic inference engine, based on the corresponding TQL.

Furthermore, even though we introduce FIO with two drop/mark precedence, it is easy to extend the fuzzy logic control methodology to multiple drop/mark precedence. The ease of this approach is the result of the design of a generic control methodology (as described in Chapter 6) that enables us to adopt easily the corresponding methodology and extend it in multiple-level related schemes.

8.8 Conclusions

We investigated the suitability of fuzzy logic to provide effective congestion for Diff-Serv environments, and particular the employment of a queue management discipline required to implement differential treatment among different traffic

priorities belonging to the Assured Forwarding PHB. Using the generic fuzzy logic based control methodology described earlier, we develop a simple Diff-Serv aware AQM controller that can realize multiple levels of drop precedence, and can offer an adequate differentiation among different priorities of traffic. At the same time, using the strengths of fuzzy logic, we can assure bounded delays and high link utilization, as well as low losses; thus addressing QoS issues to different traffic levels of precedence in TCP/IP networks. The proposed nonlinear fuzzy control methodology for Diff-Serv congestion control can offer significant improvement, on providing QoS, over the existing AQM schemes (Clark & Fang, 1998; Chait et al., 2002) due to the identified limitations they obey.

Chapter 9

Performance Evaluation of Fuzzy Explicit Marking In/Out in TCP/IP Diff-Serv Networks

9.1 Introduction

In this chapter we use simulative evaluation to demonstrate the effectiveness and robustness of the generic nonlinear fuzzy logic based AQM control methodology implemented also in TCP/IP Diff-Serv networks, namely *Fuzzy Explicit Marking In/Out* (FIO). Experiments with a wide range of network/traffic environments are conducted to compare FIO with other existing, well-known, AQM schemes for Diff-Serv congestion control, namely RIO (Clark & Fang, 1998), and two-level PI controller (Chait et al., 2002). Note that extensive simulations for the best-effort environment are included in Chapter 7.

9.2 Selection of Simulation Parameters

In order to evaluate the performance of the AQM schemes under comparison, we use the same simulation environments and parameters used in Chapter 7 for the evaluation of the proposed control methodology implemented in TCP/IP best-effort networks (see Section 7.2 for details).

We use the NS-2 simulator in both single- and multiple-congested links network environments, and we examine the effects on the AQM schemes of various network/traffic parameters, like time-varying dynamics, heterogeneous RTTs, short-lived flows that are of major significance in the investigation of the effectiveness and robustness of the algorithms under comparison. We also keep the same performance metrics, as in Chapter 7, for evaluating the performance of the fuzzy control methodology and the other selected AQM schemes; that is, bottleneck link utilization, loss rate, and mean queuing delay and its standard deviation, which provide us a QoS-aware evaluation.

Moreover, as we introduce the AQM schemes with two *mark* precedence levels: the low *mark* precedence level for the high-priority traffic ("IN" packets), and the high *mark* precedence level for the low-priority traffic ("OUT" packets), we are also interested to demonstrate adequate differentiation between these. Note that since in this thesis we focus on the core aspects of Diff-Serv, we assume that the classification is already done at the Diff-Serv boundary nodes, based on source-destination IP pairs.

9.3 Single-bottleneck Link

The network topology of a single-bottleneck link is shown in Figure 7.1 (in Chapter 7). We use TCP/Newreno. The buffer size of all queues is set to 500 packets (1000 bytes each). The sampling interval for FIO controller is set to *0,006 sec* (same as adopted by decision taken by Hollot, Misra, Towsley, and Gong, (2002)).

The TQL for FIO and PI controllers regarding the high-priority traffic is set to 40% of the buffer size, that is, 200 packets. The corresponding TQL for the low-priority traffic is set to 20% of the buffer size, that is, 100 packets. For RIO, we set, in the case of the high-priority traffic, the minimum threshold to 20% of buffer size (i.e., 100 packets) and the maximum to 60% of buffer size (i.e., 300 packets), giving an average TQL of 40% of buffer size, that is 200 packets. For low-priority traffic, we set the minimum threshold to 10% of buffer size (i.e., 50 packets) and the maximum to 30% of buffer size (i.e., 150 packets), giving an average TQL of 20% of buffer

size, that is 100 packets. For both FIO and RIO, the maximum mark probability for high-priority traffic is set to 0.02, whereas for the low-priority traffic is set to 0.1. Above settings are selected so as to provide an adequate differentiation between the high- and low-priority traffic, when congestion occurs. All nodes and algorithms are ECN-enabled. The simulation time is 100 sec.

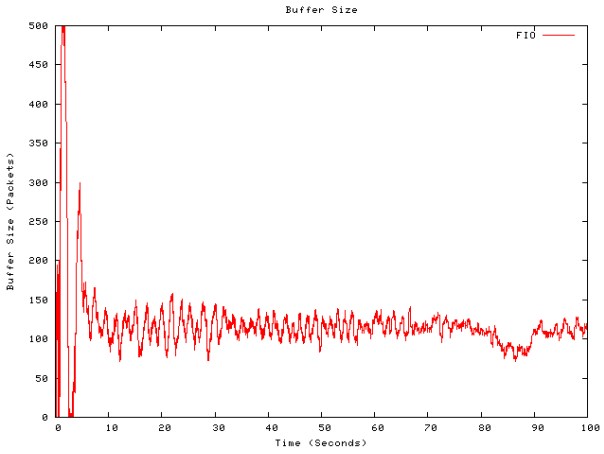
9.3.1 Scenarios I

Table C.1 (see Appendix C) contains the statistical results of the conducted experiments (mean queuing delay and its standard deviation, loss rate, and bottleneck link utilization).

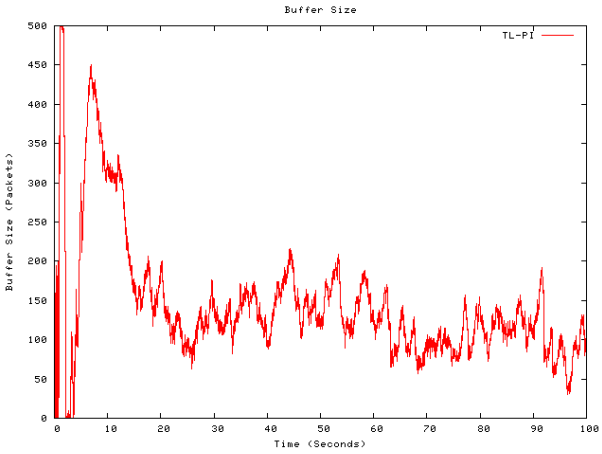
9.3.1.1 Scenario I-1-3: Effect of Increase of High-priority Traffic

In these scenarios we investigate the performance of the AQM schemes as the high-priority-tagged traffic increases in comparison to the low-priority traffic. The link capacities and propagation delays are set as follows: (C_1, d_1) = (100Mbps, 5ms), (C_2, d_2) = (15Mbps, 120ms), which is the bottleneck link, and (C_3, d_3) = (200Mbps, 5ms). All sources are greedy sustained FTP applications, and set to $N = 100$.

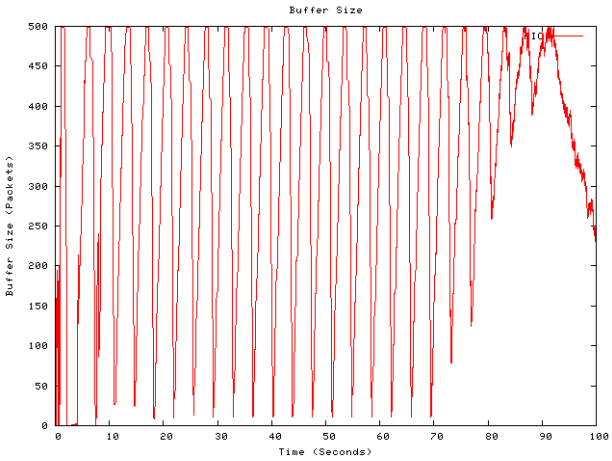
Scenario I-1 considers a limited number of flows tagged as high-priority traffic; 2% of the flows are considered belonging to high-priority, whereas the rest, 98%, are tagged as low-priority. Figure 9.1 shows the queues of FIO, RIO, and two-level-PI (we designate it as *TL-PI*, for easier convenience) where we can observe that FIO regulates its queue to the lower TQL (100 packets) that corresponds to the low-priority traffic, whereas RIO cannot regulate, at all, the queue and exhibits very large queue fluctuations that results in degraded utilization, losses and high variance of queuing delay (see Table C.1). TL-PI tries to regulate the queue at a low level; however it spends considerably long time with a considerably long delay variation. Furthermore, FIO achieves an adequate differentiation between the two traffic priority classes, despite the fact that only 2% of the traffic is tagged as of high-priority, in contrast with RIO and TL-PI that cannot provide sufficient differentiation for high-priority traffic.



(a) FIO



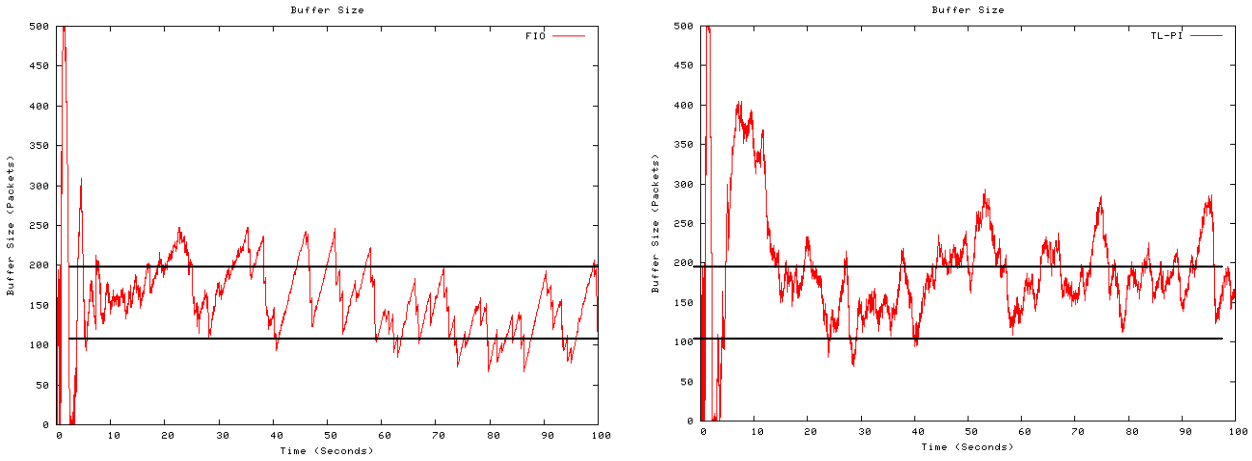
(b) two-level PI



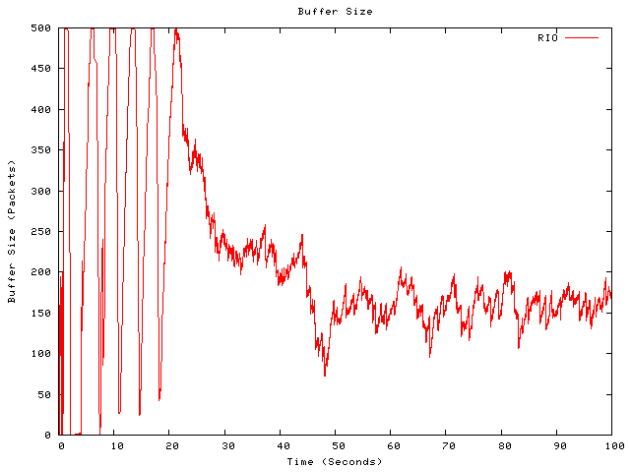
(c) RIO

Figure 9.1 Scenario I-1: Queue lengths

Scenario I-2 increases the number of flows tagged as high-priority traffic to 10% of the flows. FIO accomplishes a bounded queuing delay, between the two TQLs, that result in high link utilization and minimal losses (see Figure 9.2 and Table C.1). On the other hand, TL-PI and RIO slowly regulate their queue (especially in the case of RIO), after a significant transient period with large overshoots that result in lower utilization and higher losses than FIO has. Furthermore, FIO achieves a much higher differentiation between the two traffic priority classes, as compared with RIO and TL-PI, thus can provide adequate QoS differentiation.



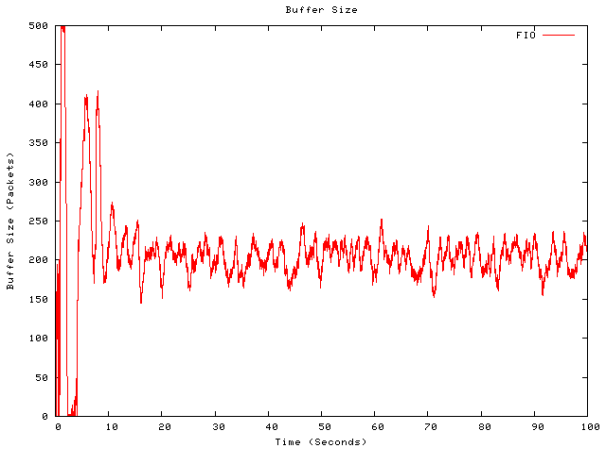
(a) FIO (b) two-level PI



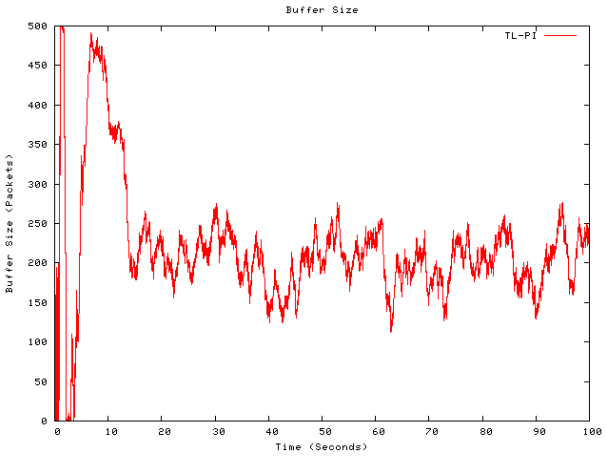
(c) RIO

Figure 9.2 Scenario I-2: Queue lengths

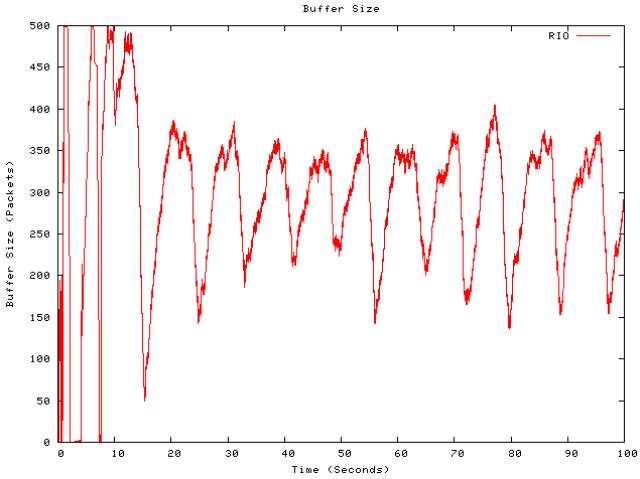
Scenario I-3 increases the number of flows tagged as high-priority traffic to 90% of the flows. In the presence of such large amount of high-priority traffic, compared with the low-priority traffic, FIO regulates its queue at the higher TQL (see Figure 9.3), and at the same time it offers the necessary differentiation needed among the two traffic of priority. RIO, on the other hand, exhibits large queue fluctuations that result in lower utilization and higher losses than FIO and TL-PI have. TL-PI shows a sluggish response to regulate the queue and less tight control than FIO.



(a) FIO



(b) two-level PI



(c) RIO

Figure 9.3 Scenario I-3: Queue lengths

Figure 9.4 shows the utilization of the link regarding only the high-priority traffic as the traffic of high-priority increases (based on the above scenarios). It is clearly shown that FIO outperforms the other two schemes in terms of better differentiation provided between the two drop precedence levels, in favor of the low drop precedence level. RIO and TL-PI schemes, on the other hand, cannot give adequate differentiation in the presence of significant amount of low-priority traffic.

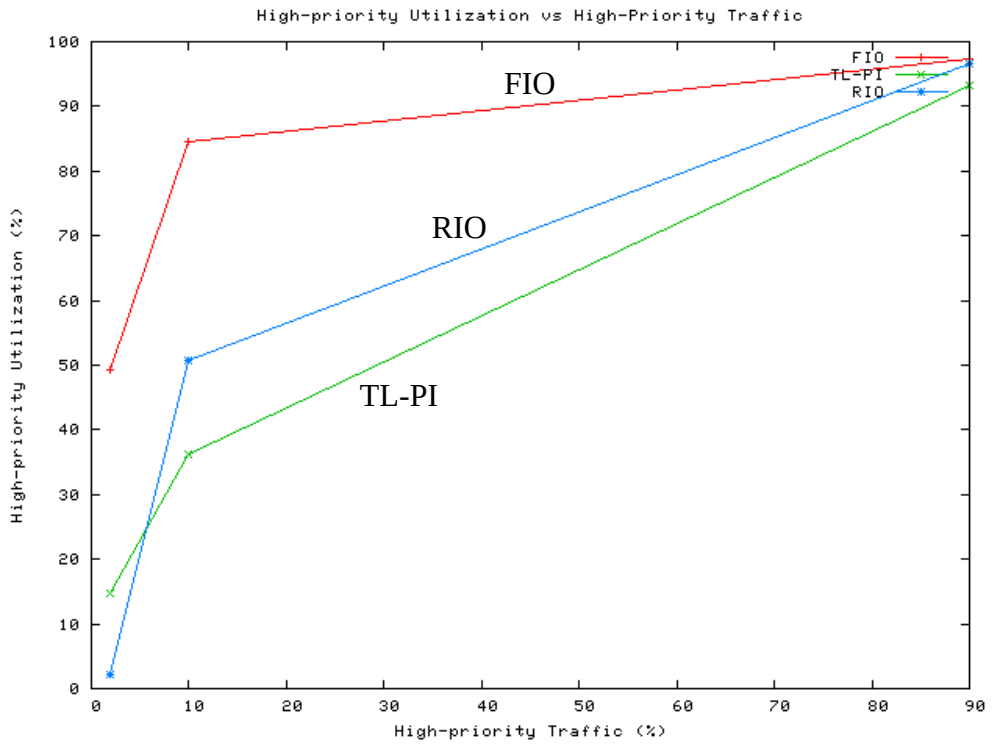
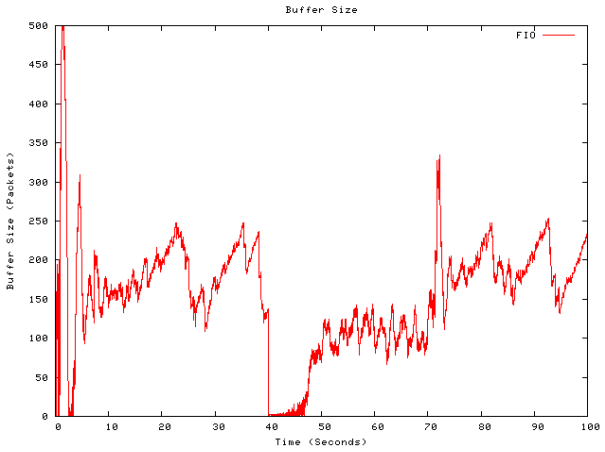


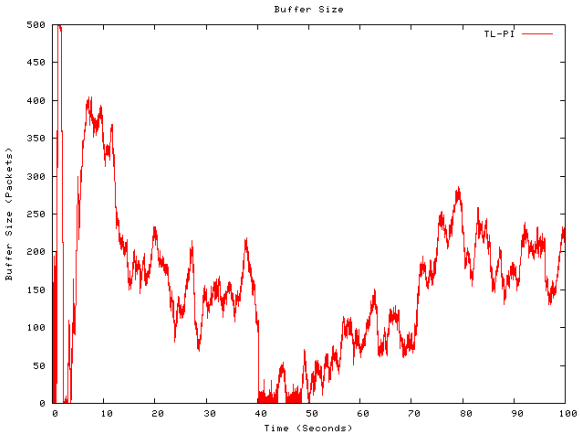
Figure 9.4 Scenarios I-1-3: Utilization of high-priority traffic vs percentage of high-priority traffic
(high-priority traffic increases from 2%, 10%, and 90% of the total traffic)

9.3.1.2 Scenario I-4: Effect of Time-varying Dynamics

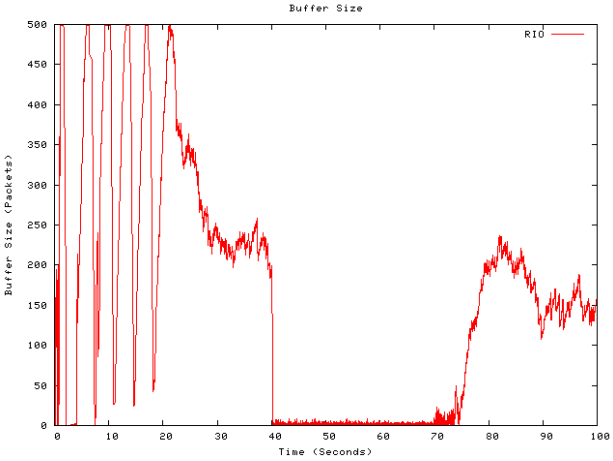
Scenario I-4 examines the behavior of the AQM schemes under dynamic traffic changes. We use *Scenario I-2*, and provide some time-varying dynamics by stopping the high-priority-tagged flows at time *t = 40 sec*, and resuming transmission at time *t = 70 sec*. The results (see Figure 9.5) show that FIO is very robust against the dynamic traffic changes and provides a very good response. Between *t = 40 – 70 sec*, where only low-priority-tagged flows are active, FIO successfully manage to regulate the queue length at the TQL for low-priority, whereas RIO fails to do so. TL-PI has a slower response, than FIO has, to regulate the queue at the specified values, when sudden change in the traffic happens. FIO also achieves better differentiation between the two traffic priorities, and provides the highest utilization and the lowest losses compared with the other AQM schemes.



(a) FIO



(b) two-level PI

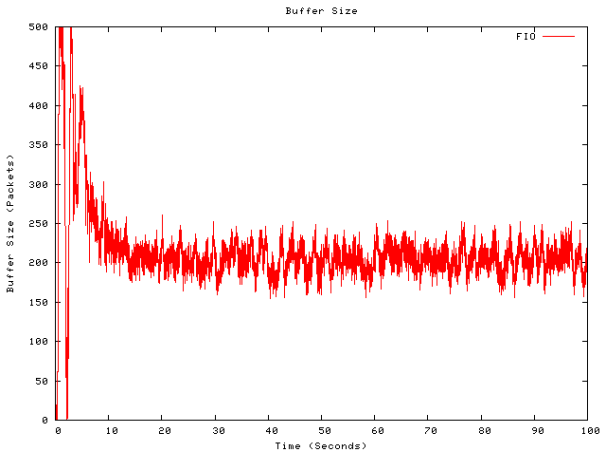


(c) RIO

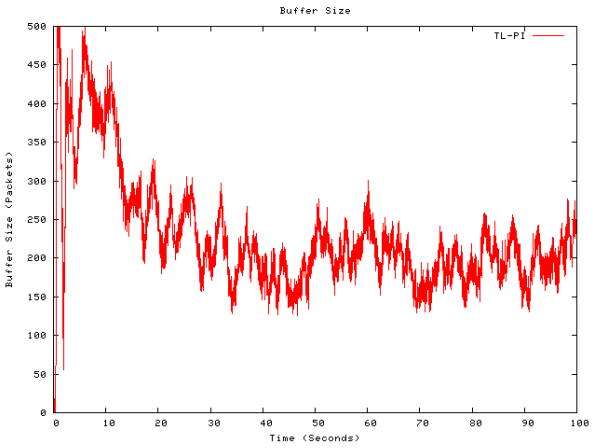
Figure 9.5 Scenario I-4: Queue lengths

9.3.1.3 Scenario I-5: Effect of Heterogeneous Propagation Delays

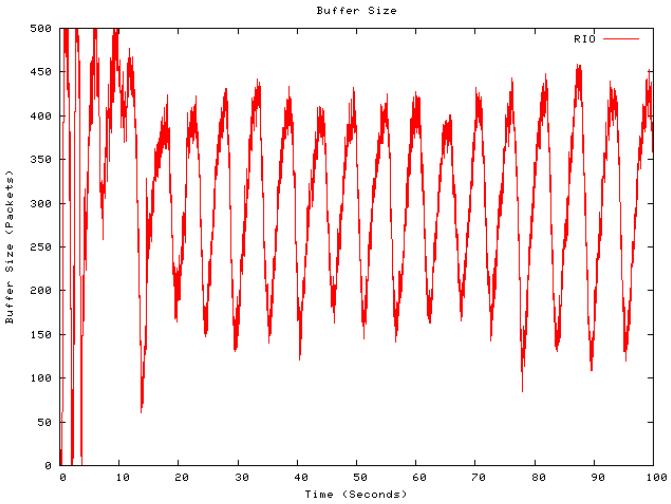
Scenario I-5 uses *Scenario I-3* and examines the effect of the RTT by having heterogeneous propagation delays of the links between the sources and router-A (we separate the 100 flows into groups of 10, and for each group - that consists of 9 high-priority-tagged flows and 1 low-priority-tagged flow – its propagation delay is increased by *5 msec*, starting from *5 msec* up to *50 msec*). The results (see Figure 9.6) show the superior steady performance of FIO with stable queue length dynamics



(a) FIO



(b) two-level PI



(c) RIO

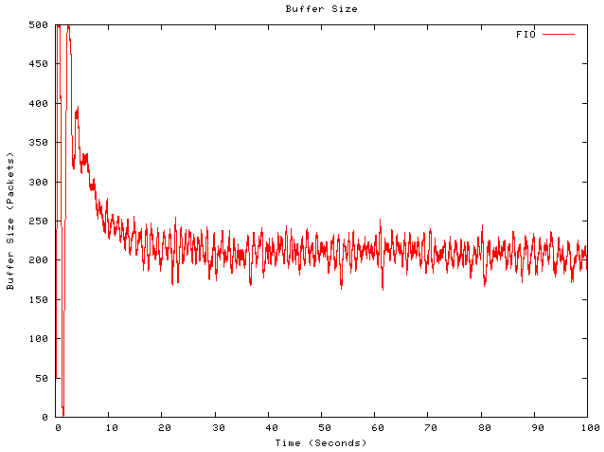
Figure 9.6 Scenario I-5: Queue lengths

that result in a high link utilization with minimal losses, while RIO exhibits large queue fluctuations, worst than in *Scenario I-3*, that result in a significant amount of losses and high variance of queuing delay. TL-PI shows again a slow transient response, with larger delay variation, than FIO. Further, FIO accomplishes an even stronger differentiation between the two priority traffic, in favour of the high-priority, by having an increased utilization for the high-priority. This is in contrast with TL-PI that exhibits an increase for the low-priority (see Table C.1).

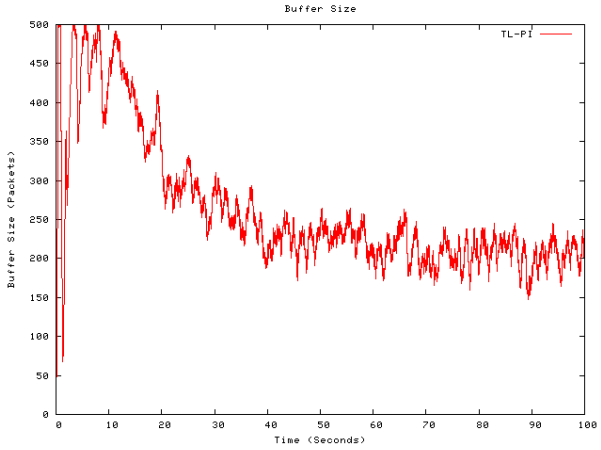
9.3.1.4 Scenario I-6: Effect of Delays

In *Scenario I-6* we investigate the performance of AQM schemes under variation of bottleneck link propagation delays. We specifically examine the effect of the round-trip time by decreasing the propagation delay from *120*, to *60*, and *30 msec*, by using the *Scenario I-3* as the basis. The results are shown in Figure 9.7 and Figure 9.8 (for *30msec* and 60msec, respectively – note that for a propagation delay of *120 msec*, Scenario I-3 applies), and Table C.1. From the results, we can observe the superior steady performance of FIO with stable queue dynamics, irrespective of the increase of RTT (as in the best-effort environment, a graceful degradation is observed as RTT increases). FIO has the highest utilization, and the lowest losses and the shortest delay variation. On the other hand, RIO exhibits large queue fluctuations as the RTT increases that result in degraded utilization and high variance of queuing delay. Also, TL-PI suffers from a slow response to regulate the queue, and has higher delay variation than FIO has. Thus, these mechanisms are shown to be more sensitive with variations of RTT.

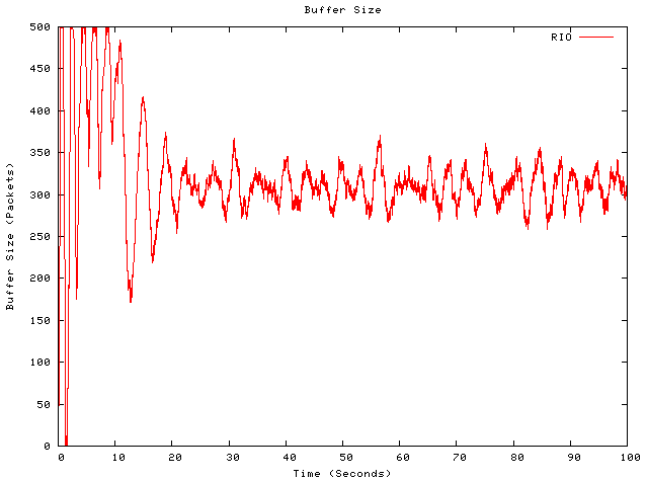
In Figure 9.9, we show the utilization of the bottleneck link regarding the high-priority traffic with respect to the mean queuing delay. FIO outperforms the other AQMs, in managing to achieve high utilization for the high-priority; thus it achieves a much higher differentiation between the two drop precedence levels, and at the same time regulating the queue and thus providing bounded mean delay, and delay variation. On the other hand, the other schemes exhibit much larger delays, and provide less differentiation between high- and low-priority traffic.



(a) FIO

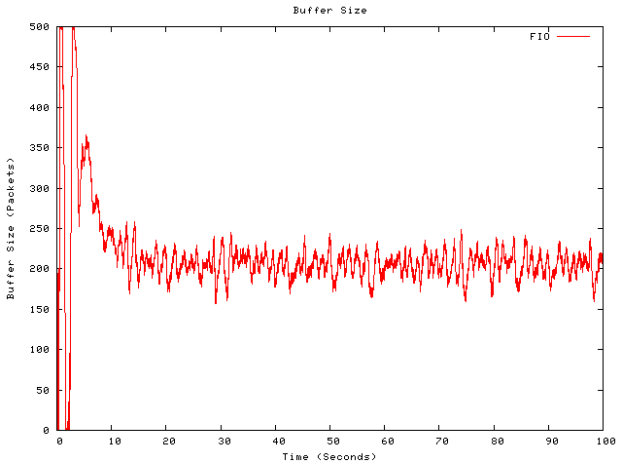


(b) two-level PI

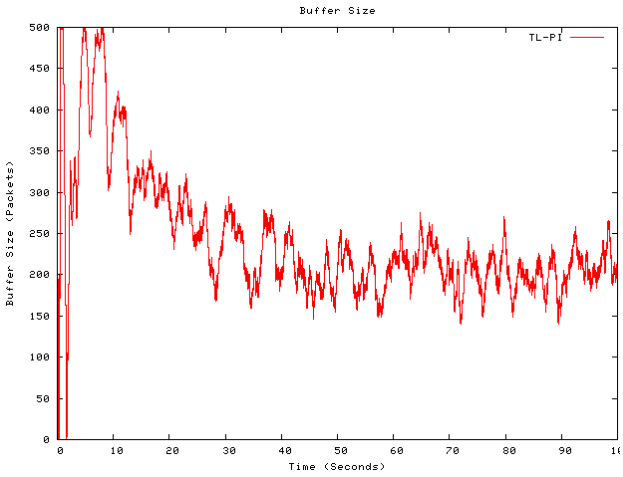


(c) RIO

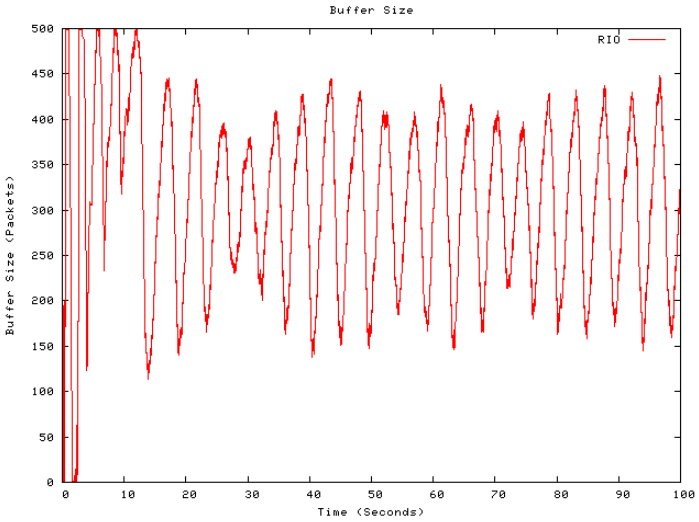
Figure 9.7 Scenario I-6: Queue lengths (bottleneck link propagation delay = 30 msec)



(a) FIO



(b) two-level PI



(c) RIO

Figure 9.8 Scenario I-6: Queue lengths (bottleneck link propagation delay = 60 msec)

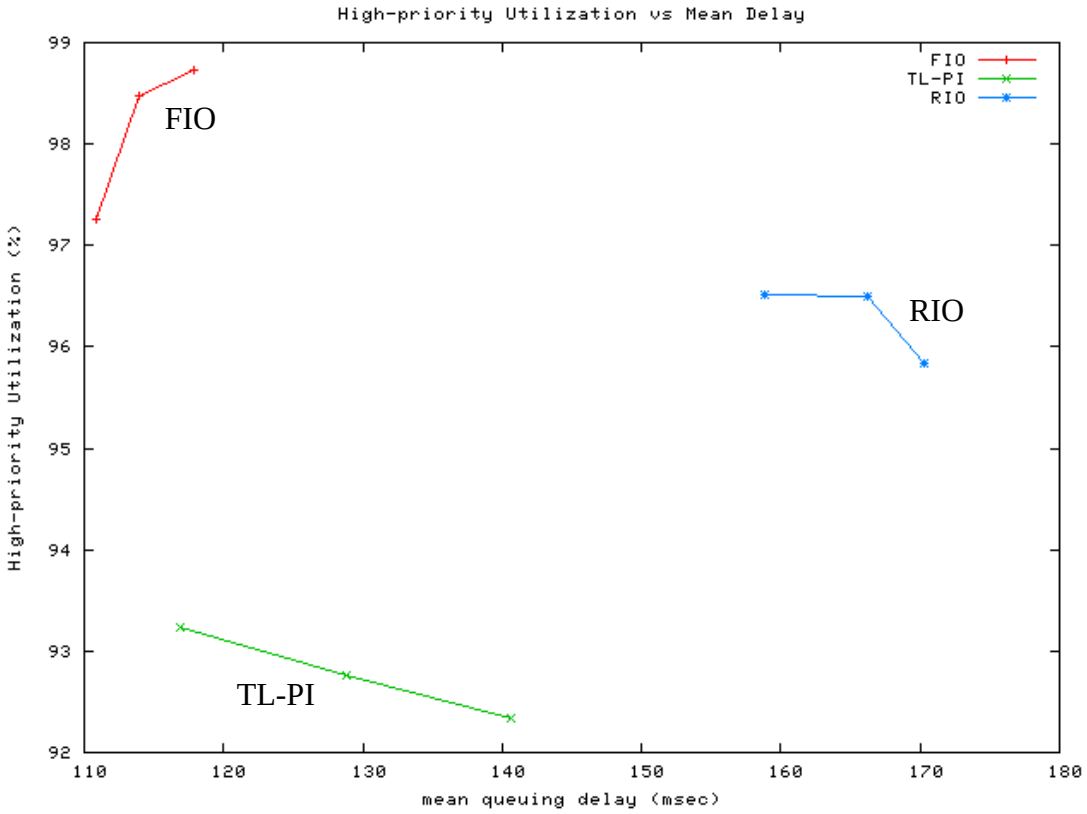
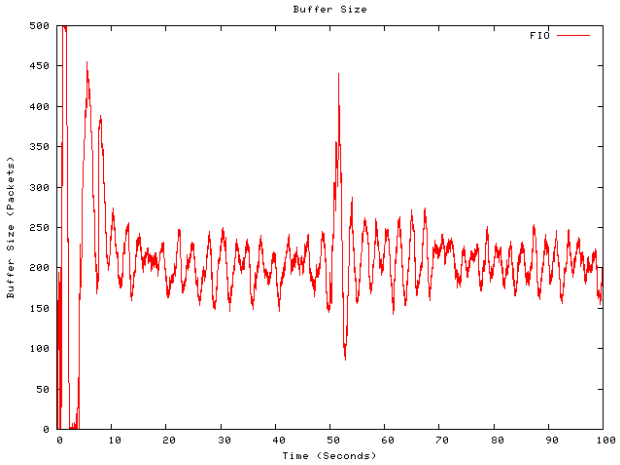


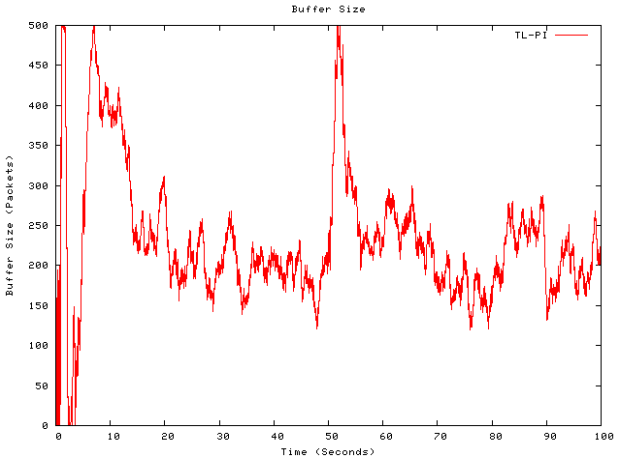
Figure 9.9 Scenario I-6: Utilization of high-priority traffic vs mean queuing delay (bottleneck propagation delay varies from 30, 60, and 120 msec)

9.3.1.5 Scenario I-7: Performance in the Presence of Short-lived Flows

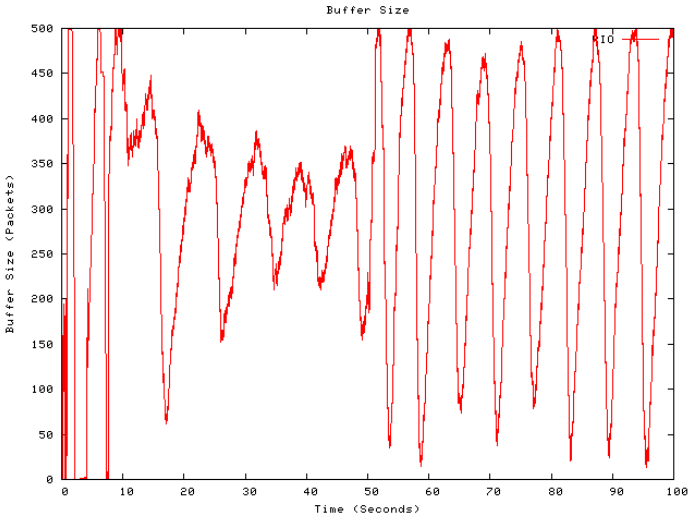
Scenario I-7 investigates the performance of AQM schemes by introducing additional web-like, short-lived traffic that can be seen as noise-disturbance to the network. In particular, we keep *Scenario I-3* as is, and at the middle of the simulation (i.e., *t=50 sec*) we introduce short-lived flows, which arrive at the link at the rate of 30 flows per second (of 20 packets each), as suggested by Kunniyur and Srikant (2004). The web-like traffic is categorized to belong in the high-priority traffic. Figure 9.10 shows the queue length evolution of the AQM schemes. We can observe the robustness of the FIO controller that adequately controls the queue, and retains the regulation of the queue length, quickly after the sudden introduction of short flows. It exhibits the highest utilization, with the lowest losses, and the shortest delay variation (see Table C.1).



(a) FIO



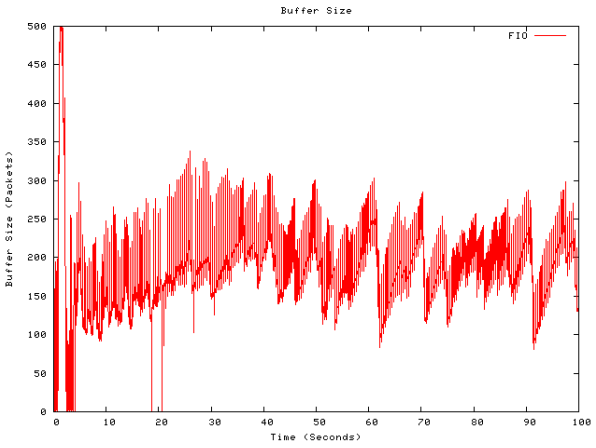
(b) two-level PI



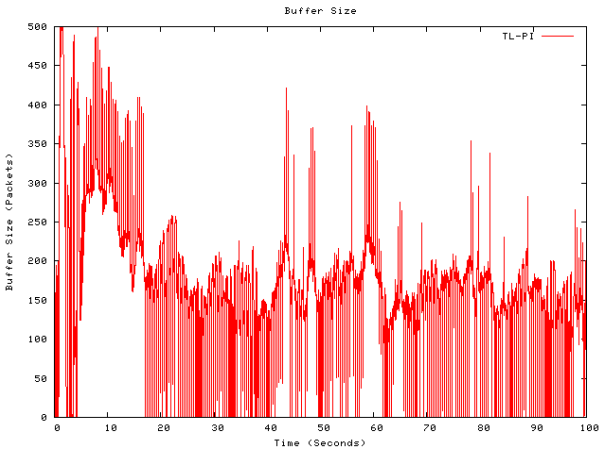
(c) RIO

Figure 9.10 Scenario I-7: Queue lengths

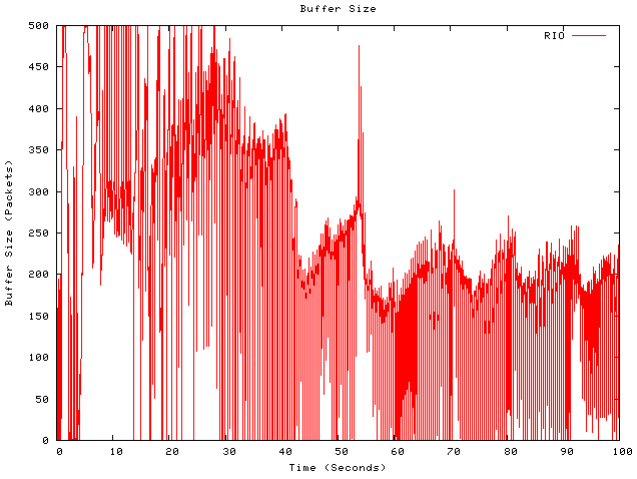
This is in contrast with the other AQM schemes, which at *t=50sec* are greatly influenced by the introduction of short flows. RIO and TL-PI suffers from a significant transient response with large overshoots (especially in the case of RIO that maintains the large oscillations). Further, once more, FIO exhibits the highest differentiation between the two precedence levels.



(a) FIO



(b) two-level PI



(c) RIO

Figure 9.11 Scenario I-8: Queue lengths

9.3.1.6 Scenario I-8: Effect of Reverse-path Traffic

Scenario I-8 investigates the effect of reverse traffic on the behavior of the AQM schemes. We use the *Scenario I-3* as the basis, and we additionally introduce 5 FTP flows sending data in the reverse path, which are considered to belong to the high-priority traffic aggregate (see Figure 9.11 and Table C.1).

We can observe that in the presence of the reverse-path traffic, the FIO controller responds adequately, and manages to keep the queue around the desired value, while exhibiting the highest utilization and the lowest – minimal – losses. On the other

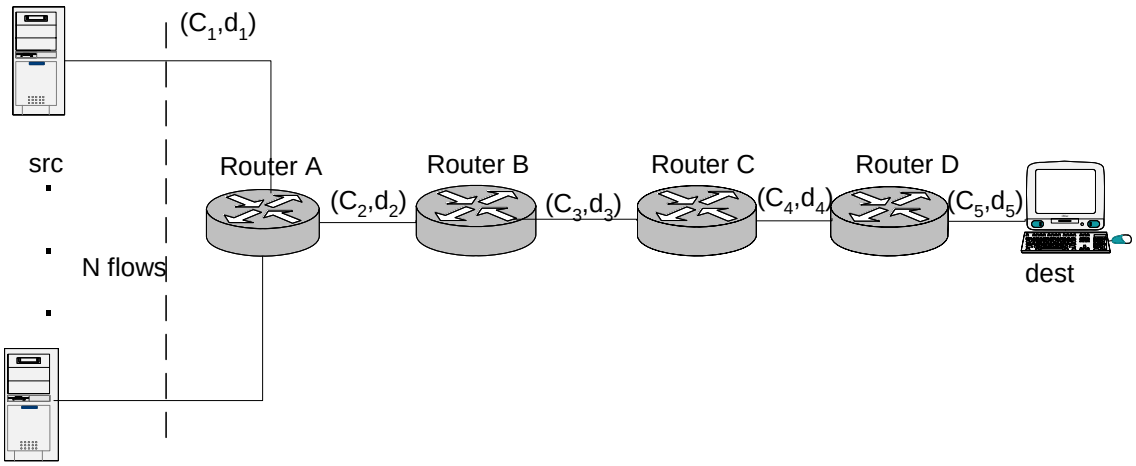


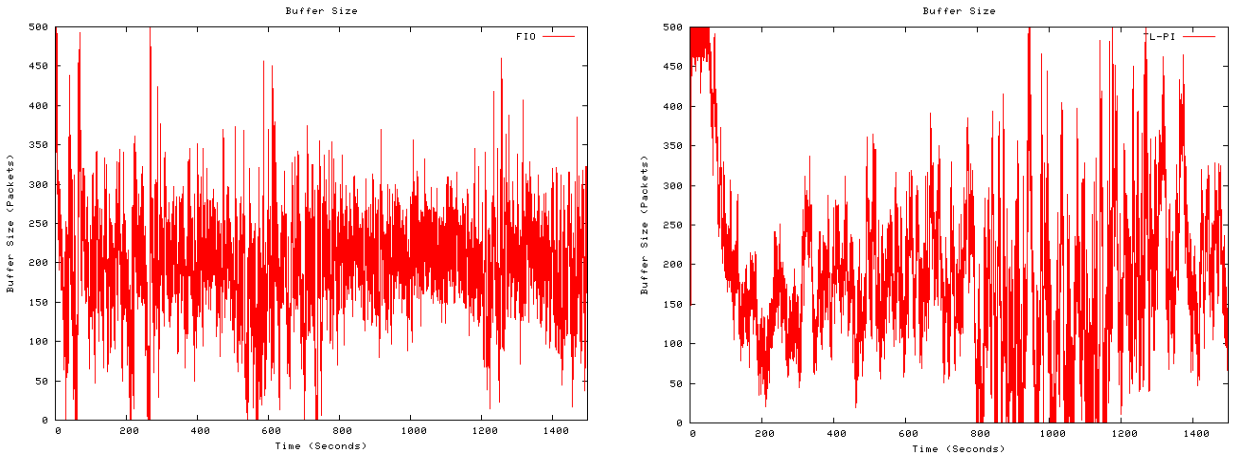
Figure 9.12 Single-bottleneck network topology II

hand, the other AQM schemes have responded badly to the presence of reverse-path traffic. Specifically, RIO and TL-PI cannot regulate the queue at all, and have shown large oscillations, with no sign of "driving" the queue to the reference value. This has as a result to have larger loss rate and lower link utilization than FIO achieves.

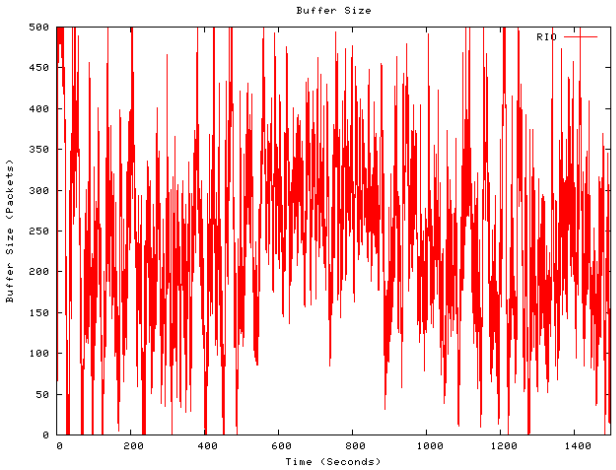
Further, FIO can adequately provide the necessary differentiation between the two levels of precedence, irrespective of the presence of the reverse-path traffic. However, RIO and especially TL-PI have shown to be very sensitive to such conditions, and have not given enough differentiation to the two priority levels (see Table C.1). For example, TL-PI provides 65% of link utilization to low-priority and 30% to high-priority traffic, which is very poor performance, whereas FIO shows its superiority by providing 81% to high-priority and only 18% to the low-priority traffic.

9.3.1.7 Scenario I-9: Effect of Intense Web Traffic

Scenario I-9 uses the network topology shown in Figure 9.12 with the introduction of TCP/Web-like traffic too. This network topology, as well as the distributions and parameters for web traffic (summarized in Table B.2 in Appendix B) are introduced by Iannaccon et al. (2001). We use TCP/SACK with an advertised window of 240 packets. The size of each packet is 1514 bytes. The buffer size of all queues is 500



(a) FIO (b) two-level PI



(c) RIO

Figure 9.13 Scenario I-9: Queue lengths

packets. We use AQM in the queues of the bottleneck link between router-B and router-C (of *10Mbps* capacity). The link capacities and propagation delays are set as follows: (C_1, d_1) = (500Mbps, 30ms), (C_2, d_2) = (100Mbps, 10ms), (C_3, d_3) = (10Mbps, 30ms), (C_4, d_4) = (100Mbps, 10ms), and (C_5, d_5) = (500Mbps, 1ms), while $N = 100$. The simulation time is *1500 sec*.

In this experiment web traffic is considered belonging to high-priority traffic, whereas FTP traffic is considered belonging to low-priority traffic class. The number of flows tagged as high-priority traffic class is set to 10, while the rest (90 flows) are tagged as low-priority traffic class. We create intense web traffic, based on the

suggestions provided by Iannaccon et al. (2001). We simulate 1000 web sessions with distributions and parameters as given in Table B.2.

As stated by Iannaccon et al. (2001), in case of web traffic the standard deviation of the queuing delay is mainly determined by the burstiness of the arriving traffic. Due to the high traffic variability, there exists a large possibility for high queueing delays. Even with these circumstances, FIO manages to maintain better the queue around the higher TQL (200 packets), in contrast with RIO and TL-PI (see Figure 9.13). Furthermore, FIO appears to control better the flow rate across the network and provide a more stable behaviour. From the results (see Table C.1) it can be seen that FIO can provide the necessary congestion control and differentiation and ensure acceptable QoS in a Diff-Serv network. It achieves a better discrimination between the two traffic classes than RIO and TL-PI do, whilst maintaining high utilization and minimal losses.

9.4 Multiple-bottleneck Links

The network topology of multiple-bottleneck links is shown in Figure 7.24 (in Chapter 7). We have considered network topologies with multiple bottleneck links in order to examine the performance of the AQM schemes in more realistic scenarios.

We use AQM in the queues of all core links from router-A to router-F. All other links (access links) have a simple Drop Tail queue. The link capacities and propagation delays are set as follows: (C_1, d_1) = (C_8, d_8) = (C_9, d_9) = (100Mbps, 5ms), (C_2, d_2) = (C_4, d_4) = (C_6, d_6) = (15Mbps, 10ms), (C_3, d_3) = (15Mbps, 60ms), (C_5, d_5) = (15Mbps, 30ms), and (C_7, d_7) = (C_{10}, d_{10}) = (C_{11}, d_{11}) = (200Mbps, 5ms). N_1 flows end up at destination 1, whereas N_2 flows end up at destination 2, and N_3 flows end up at destination 3 creating cross traffic. The results show that both bottleneck links, where the cross traffic exists, (i.e., between router-B and router-C, and between router-D and router-E) exhibit similar behaviour, as far as the performance comparison is concerned. Therefore, we have chosen the bottleneck link between router-B and router-C to show the results obtained.

9.4.1 Scenarios II

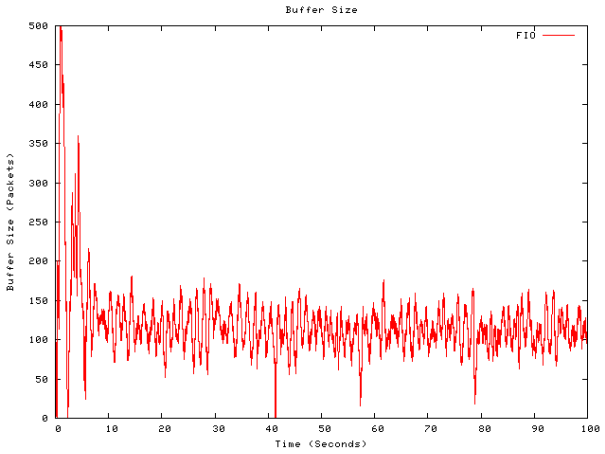
Table C.2 (see Appendix C) contains the statistical results of the conducted experiments (mean queuing delay and its standard deviation, loss rate, and bottleneck link utilization).

9.4.1.1 Scenario II-1-3: Effect of Increase of High-priority Traffic

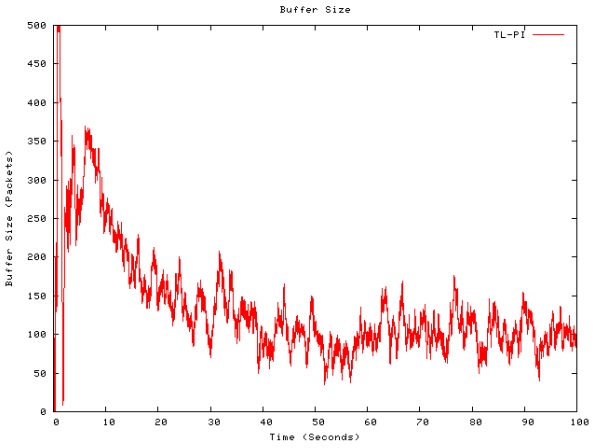
In these scenarios we investigate the performance of the AQM schemes as the high-priority-tagged traffic increases.

In *Scenario II-1* all sources ($N_1=100$, $N_2=50$, and $N_3=100$ flows) are greedy sustained FTP applications. We consider a limited number of flows tagged as high-priority traffic (2% of N_1 flows, whereas the rest 98% are tagged as low-priority). N_2 and N_3 flows are considered as being of low-priority (i.e., about 1.33% of the traffic passing through the bottleneck link, under consideration, belongs to the high-priority level). Figure 9.14 shows the queues of FIO, RIO, and TL-PI, where we can observe that FIO regulates its queue to the lower TQL (100 packets), whereas RIO exhibits very large queue fluctuations that results in degraded utilization, losses and high variance of queuing delay (see Table C.2). TL-PI shows a slow response to regulate the queue, with considerably delay variation. Furthermore, FIO achieves an adequate differentiation between the two traffic classes (it gives a considerable portion of the link utilization to the high-priority traffic), in contrast with RIO that cannot provide sufficient link utilization for high-priority class traffic. TL-PI has the worst behavior in terms of differentiation, as it only provides 1% of the link capacity to the high-priority traffic.

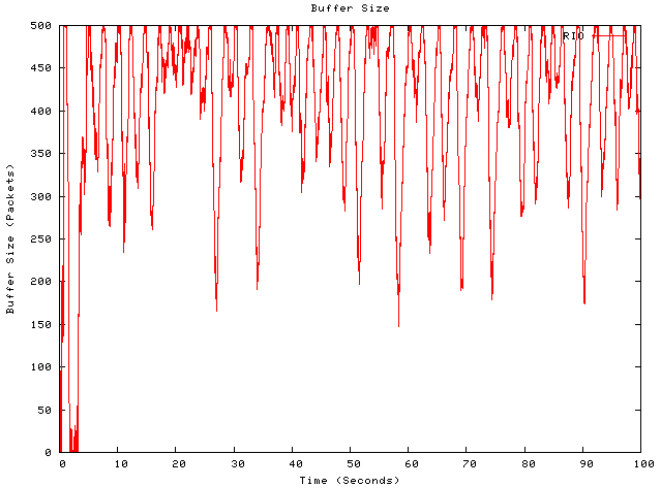
Scenario II-2 increases the number of flows tagged as high-priority traffic as follows: 15 out of 100 N_1 flows, 5 out of 50 N_2 flows, and 5 out of 100 N_3 flows (i.e., about 13.33% of the traffic passing through the bottleneck link under consideration). FIO accomplishes a bounded queuing delay, between the two TQLs, that result in high link utilization and minimal losses (see Figure 9.15 and Table C.2). On the other hand, RIO and TL-PI slowly regulate their queue, after a significant transient period with large overshoots that results in lower utilization and higher losses than FIO has.



(a) FIO



(b) two-level PI

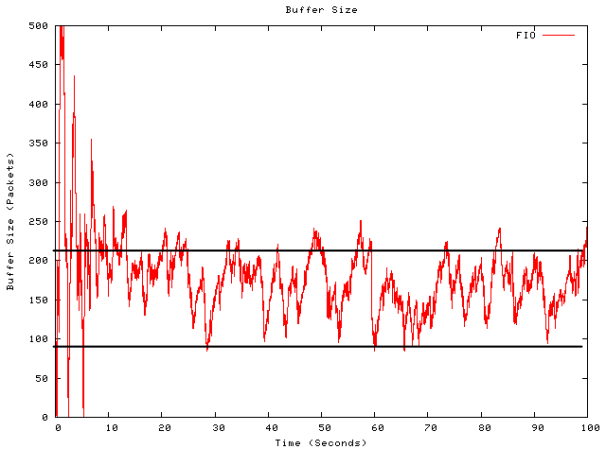


(c) RIO

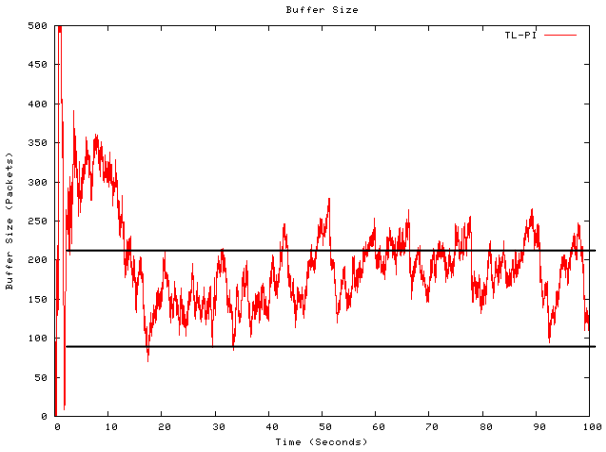
Figure 9.14 Scenario II-1: Queue lengths

Furthermore, FIO achieves a high differentiation between the two traffic classes (as it provides 93% utilization regarding the high-priority traffic and only 7% for low-priority), as compared with RIO and TL-PI (that provide only 34% and 66%, respectively), thus can provide adequate QoS differentiation.

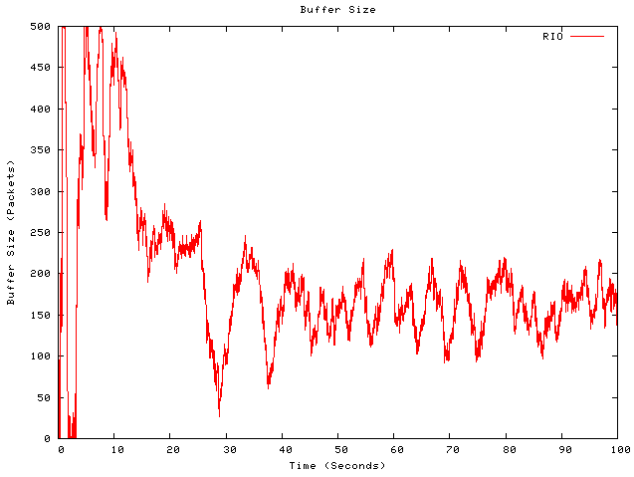
Scenario II-3 increases the number of flows tagged as high-priority traffic to 90% of N_1 flows, and uses all N_2 and N_3 flows as high-priority traffic (i.e., about 93.33% of the traffic passing through the bottleneck link under consideration). In the presence of large amount of high-priority traffic, FIO regulates its queue at the



(a) FIO



(b) two-level PI

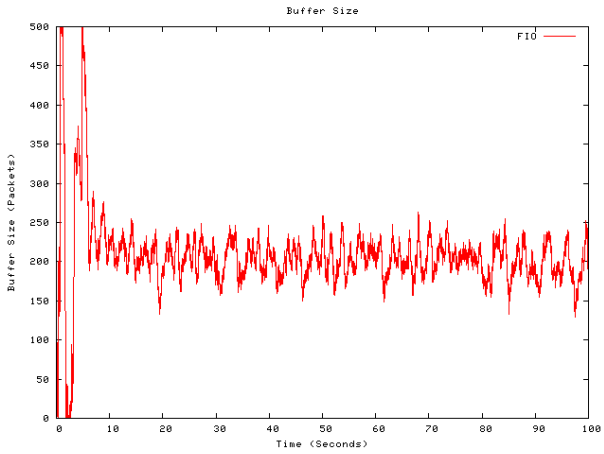


(c) RIO

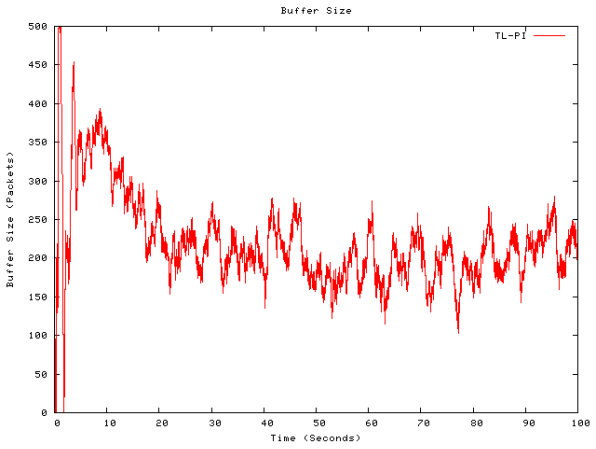
Figure 9.15 Scenario II-2: Queue lengths

higher TQL (see Figure 9.16). RIO, on the other hand, exhibits large queue fluctuations that result in lower utilization and higher losses than FIO has. TL-PI suffers from the sluggishness to regulate the queue.

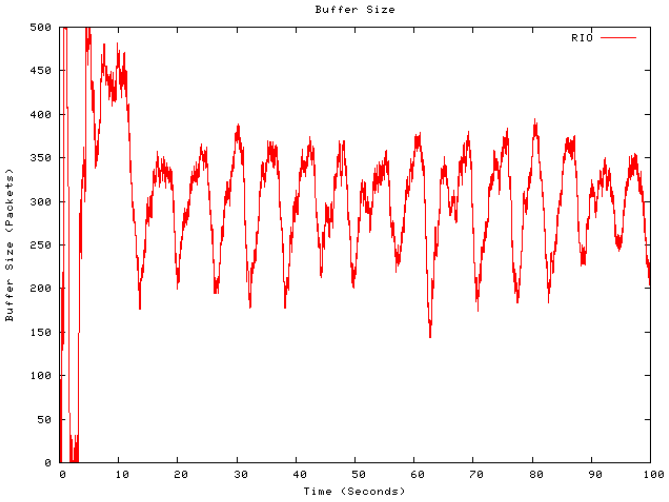
Figure 9.17 shows the utilization of the link regarding only the high-priority traffic as the traffic of high-priority increases (based on the above scenarios). It is clearly shown that FIO outperforms the other two schemes in terms of much better differentiation provided between the two drop precedence levels, in favor of the low drop precedence level (especially when lower amount of high-priority compared to



(a) FIO



(b) two-level PI



(c) RIO

Figure 9.16 Scenario II-3: Queue lengths

the low-priority traffic exists). RIO and TL-PI schemes, on the other hand, cannot give adequate differentiation in the presence of significant amount of low-priority traffic.

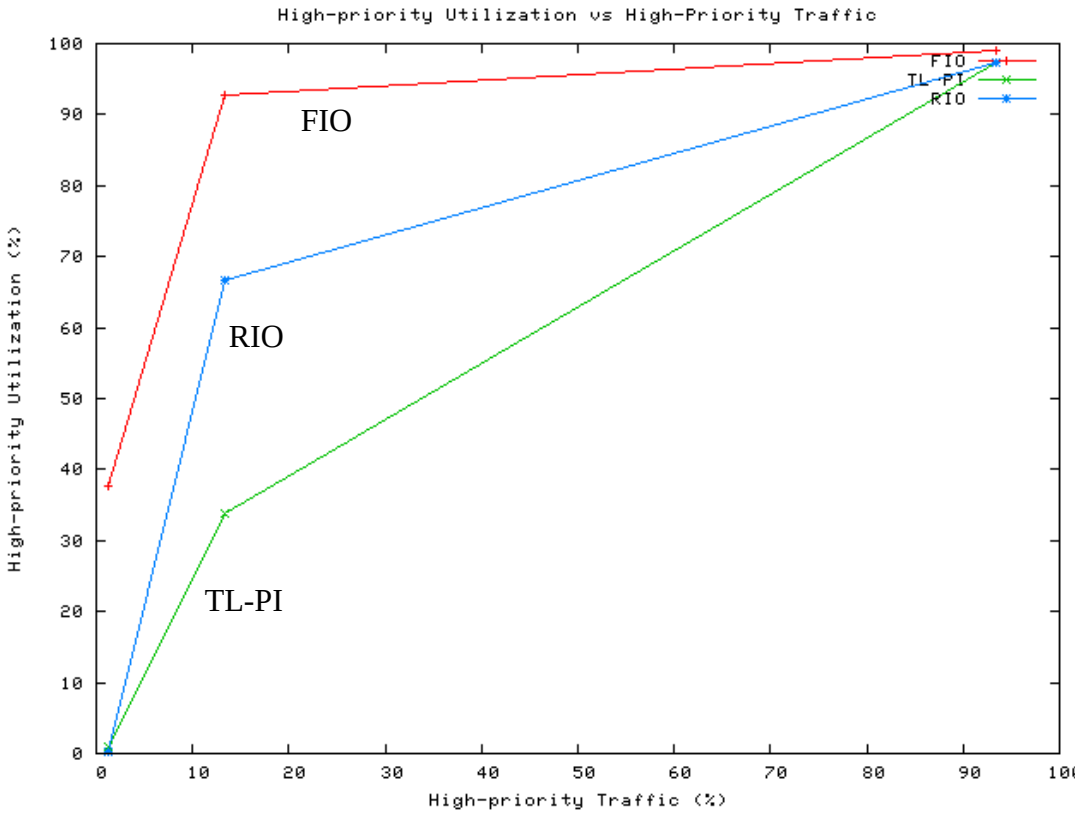
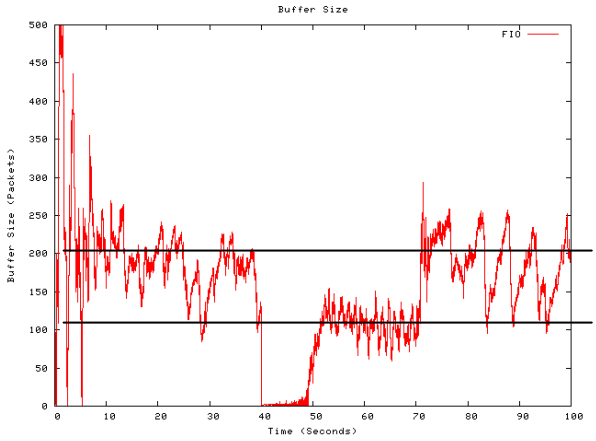


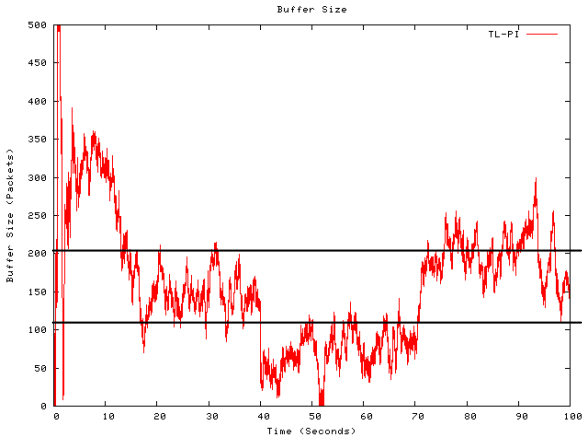
Figure 9.17 Scenarios II-1-3: Utilization of high-priority traffic vs percentage of high-priority traffic
(high-priority traffic increases from 1.33%, to 13.33%, and 93.33% of the total traffic passing through the bottleneck link)

9.4.1.2 Scenario II-4: Effect of Time-varying Dynamics

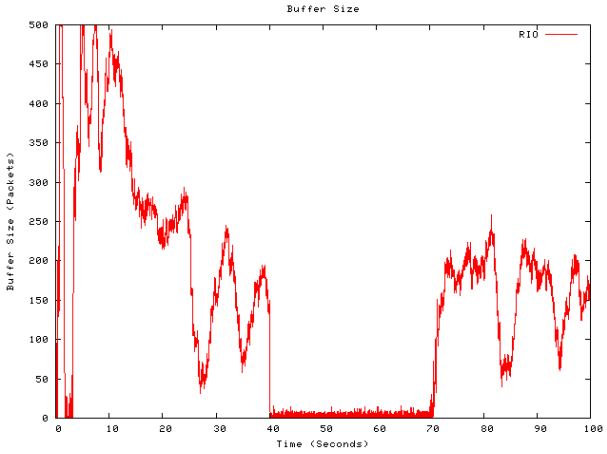
Scenario II-4 examines the behavior of the AQM schemes under dynamic traffic changes. We use *Scenario II-2*, and provide some time-varying dynamics by stopping the high-priority-tagged flows at time *t = 40 sec*, and resuming transmission at time *t = 70 sec*. The results (see Figure 9.18) show that FIO is very robust against the dynamic traffic changes and keeps very good response. Between *t = 40 – 70 sec*, where only low-priority-tagged flows are active, FIO successfully manages to regulate the queue length at the TQL for low-priority, whereas RIO fails to do so. At that interval, TL-PI cannot regulate the queue at its lower TQL, as it supposed to do, and it oscillates between empty and low TQL. FIO also achieves better differentiation between the two traffic priorities (whereas, TL-PI, once more, gives



(a) FIO



(b) two-level PI



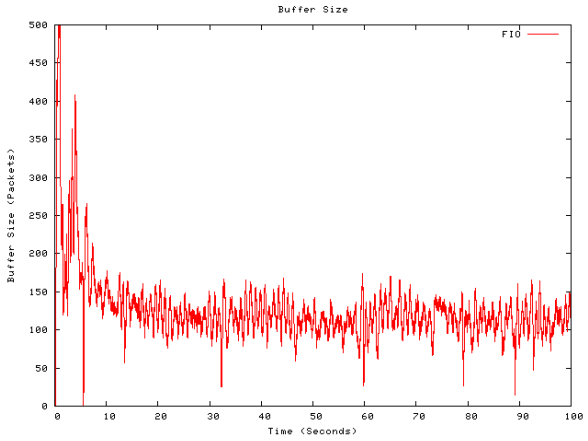
(c) RIO

Figure 9.18 Scenario II-4: Queue lengths

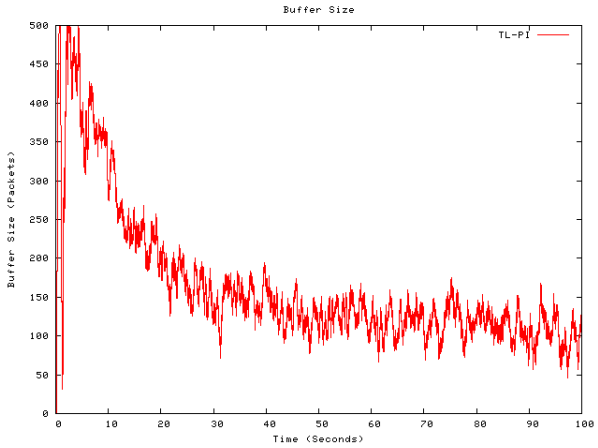
much more capacity to the low-priority traffic compared with the high-priority, instead of doing the opposite), and provides the lowest losses compared with the other AQM schemes.

9.4.1.3 Scenario II-5: Effect of Delays

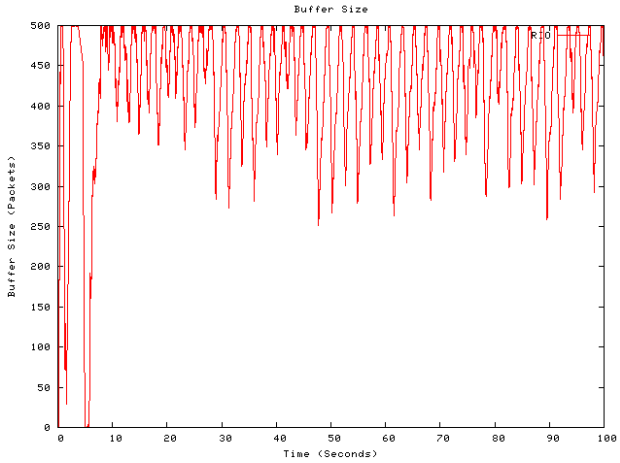
In *Scenario II-5* we investigate the performance of AQM schemes under variation of the bottleneck link propagation delays. We specifically examine the effect of the round-trip time by varying the propagation delay from *30*, to *60*, and *120 msec*, by



(a) FIO



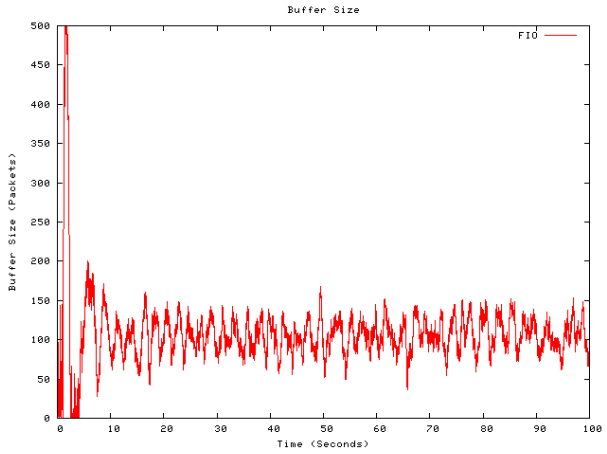
(b) two-level PI



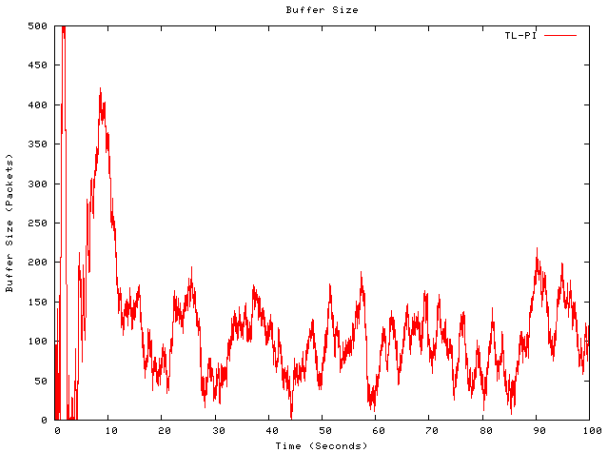
(c) RIO

Figure 9.19 Scenario II-5: Queue lengths (bottleneck link propagation delay = 30 msec)

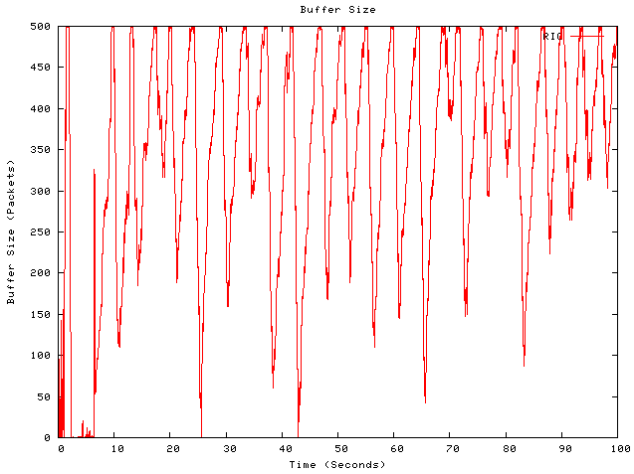
using the *Scenario II-1* as the basis (recall that in this scenario, the high-priority traffic consists of 1.33% of the total traffic passing through the particular link). The results are shown in Figure 9.19 and Figure 9.20 (for *30msec* and *120msec*, respectively – note that for a propagation delay of 6*0 msec*, Scenario II-1 applies), and Table C.2. From the results, we can observe the superior steady performance of FIO with stable queue dynamics, irrespective of the change in RTT. FIO has the highest utilization, and the lowest losses and the shortest delay variation. On the other hand, RIO exhibits larger queue fluctuations as the RTT increases that result in



(a) FIO



(b) two-level PI



(c) RIO

Figure 9.20 Scenario II-5: Queue lengths (bottleneck link propagation delay = 120 msec)

degraded utilization and high variance of queuing delay. Also, TL-PI suffers from a slow response to regulate the queue, and has higher delay variation than FIO has. Thus, these mechanisms are shown to be sensitive to variations of RTT.

In Figure 9.21, we show the utilization of the bottleneck link regarding the high-priority traffic with respect to the mean queuing delay. Despite the fact that the high-priority traffic consists only of 1.33% of the total traffic passing through the particular link, FIO outperforms the other AQMs, in managing to achieve a considerable amount of utilization for the high-priority; thus it achieves a much

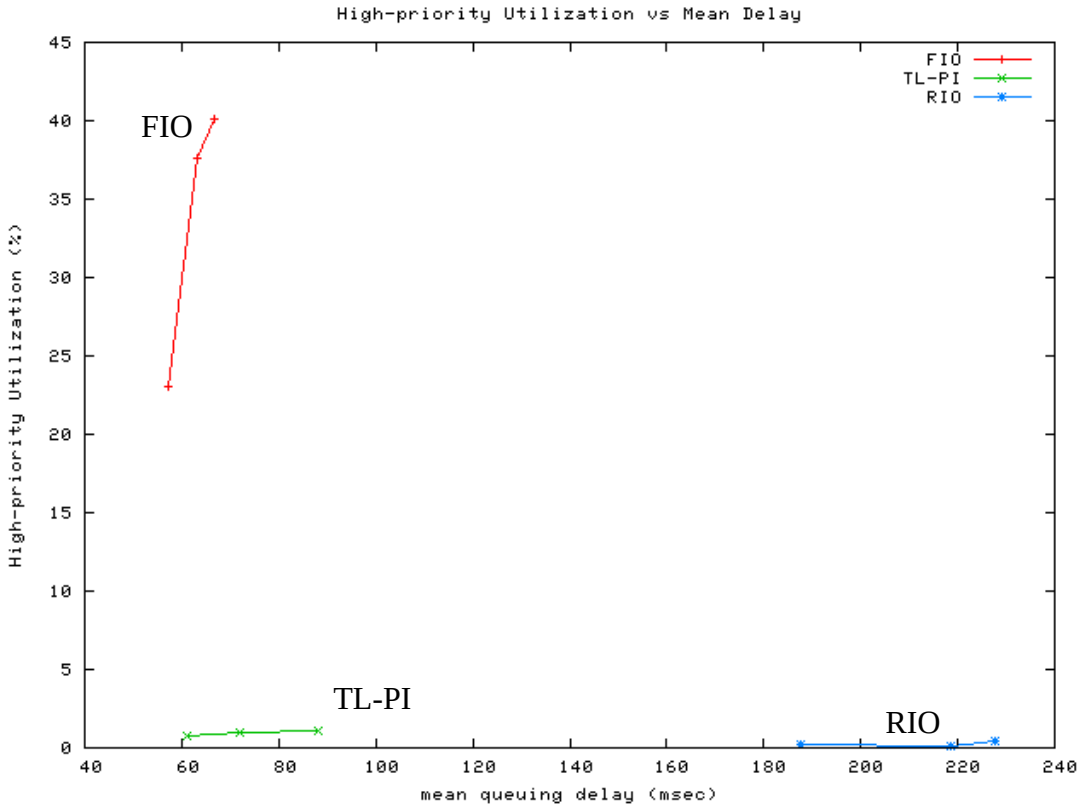
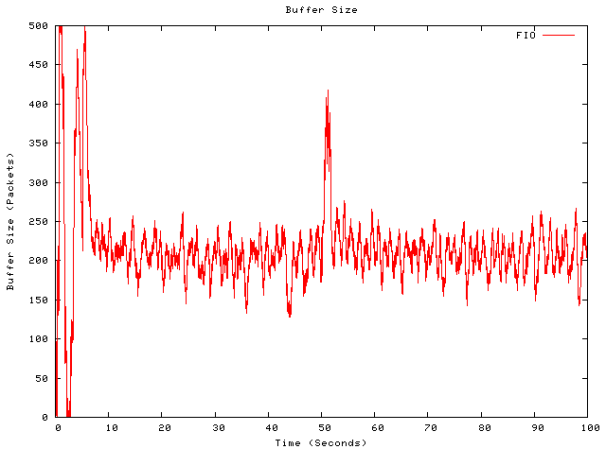


Figure 9.21 Scenario II-5: Utilization of high-priority traffic vs mean queuing delay (bottleneck propagation delay varies from 30, 60, and 120 msec – high-priority traffic consists of 1.33% of the total traffic passing through the bottleneck link)

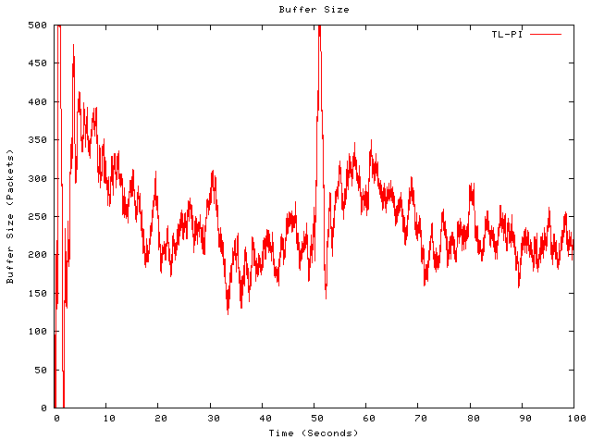
higher differentiation between the two drop precedence levels compared with the other schemes, and at the same time regulating the queue and thus providing bounded mean delay, and delay variation. On the other hand, the other schemes exhibit larger delays, and provide no differentiation between high- and low-priority traffic.

9.4.1.4 Scenario II-6: Performance in the Presence of Short-lived Flows

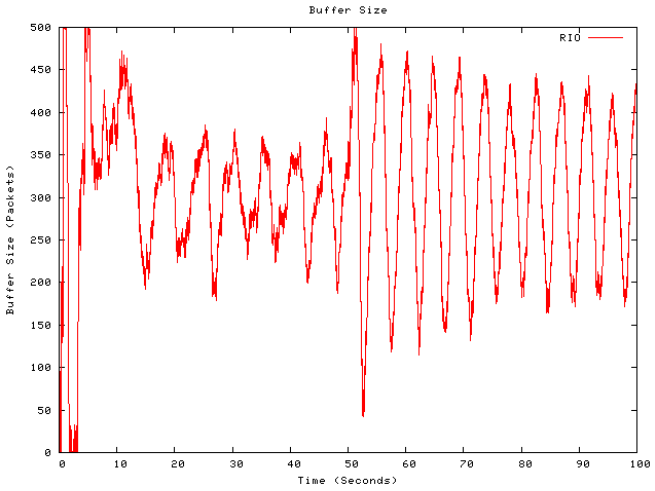
Scenario II-6 investigates the performance of AQM schemes by introducing additional web-like, short-lived, traffic that can be seen as noise-disturbance to the network. In particular, we keep *Scenario II-3* as is, and at the middle of the simulation (i.e., *t=50 sec*) we introduce short-lived flows, as previously explained (in *Scenario I-7*). The web-like traffic is introduced as high-priority traffic. These flows



(a) FIO



(b) two-level PI



(c) RIO

Figure 9.22 Scenario II-6: Queue lengths

are introduced as cross traffic passing through the bottleneck link under consideration. Figure 9.22 shows the queue length evolution of the AQM schemes. We can observe the robustness of the FIO controller that adequately controls the queue, and retains the regulation of the queue length, quick after the sudden introduction of short flows. It exhibits the highest utilization, with the lowest losses, and the shortest delay variation (see Table C.2).

This is in contrast with the other AQM schemes that it is evident at *t=50sec* that they are greatly influenced by the introduction of short flows. RIO maintains large

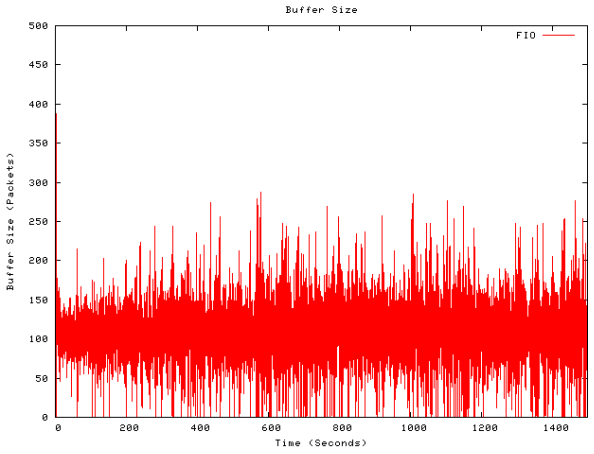
oscillations, after the critical time of *50 sec*, and TL-PI suffer from a significant transient response with large overshoots. Further, once more, FIO exhibits the highest differentiation among the two precedence levels.

9.4.1.5 Scenario II-7: Effect of Intense Web Traffic

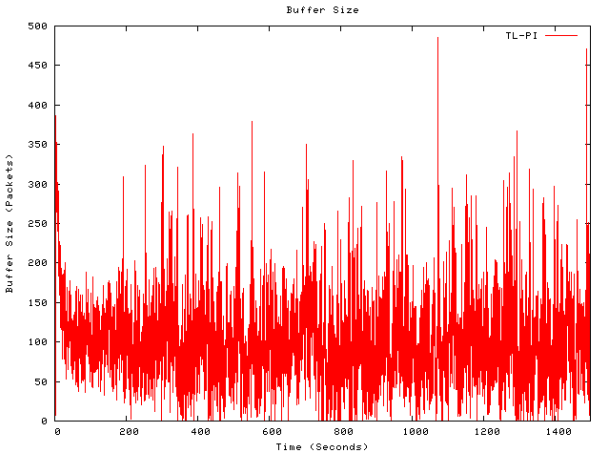
Scenario II-7 introduces TCP/Web traffic too. We use the distributions and parameters for web traffic as summarized in Table B.2 in Appendix B (introduced by Iannaccon et al. (2001)). In this experiment web traffic is considered belonging to high-priority traffic, whereas FTP traffic is considered belonging to low-priority traffic class. The number of flows tagged as high-priority traffic is set to 15 out of 100 N_1 flows, and the rest are tagged as low-priority traffic.

Even in the case of intense web traffic with high traffic variability – and possibly high queuing delays – FIO (see Figure 9.23) manages to maintain the queue around the lower TQL (100 packets), while RIO exhibits larger queue fluctuations that result in higher delays and losses. TL-PI shows a good performance, similar to FIO, however, it fails to give the appropriate differentiation to the two levels of precedence. It only offers 18% of link utilization to the high-priority traffic, whereas FIO achieves a high differentiation providing 82% of the link utilization to the high-priority traffic (see Table C.2).

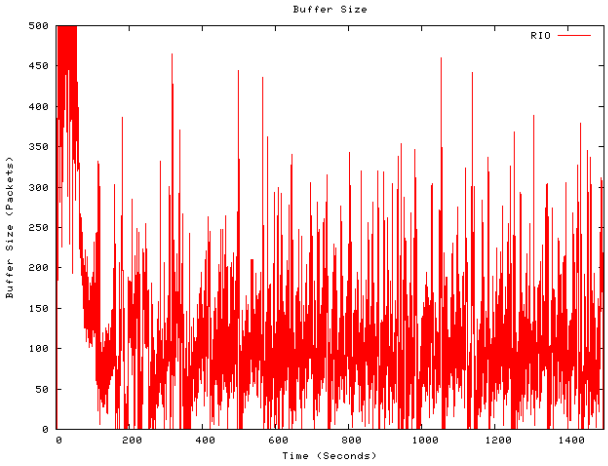
From the results (see Table C.2) it can be seen that FIO can ensure acceptable QoS in a Diff-Serv network by regulating the queue of the bottleneck link, while achieving the highest utilization, minimal losses and low delay, as compared to the other AQM schemes. The FIO controller also achieves an adequate differentiation between high- and low-priority traffic, in favor of the high-priority.



(a) FIO



(b) two-level PI



(c) RIO

Figure 9.23 Scenario II-7: Queue lengths

9.5 Conclusions

A detailed simulative evaluation and comparison of the proposed nonlinear fuzzy logic based AQM controller for Diff-Serv congestion control (FIO) with existing, well known AQM schemes (RIO, and two-level PI) has been carried out, using the NS-2 simulator. We specifically investigate the suitability of these AQM mechanisms to provide effective congestion at Diff-Serv core environments. The obtained results demonstrate that FIO can effectively offer adequate QoS, like high throughput, low losses, and bounded delays, combined with a satisfactory differentiation among different traffic levels of priority. FIO is shown to exhibit many desirable properties, like robustness and fast system response, with capabilities of adapting to highly variability and uncertainty in network.

RIO has been shown to result in under utilization of resources and a degradation in its performance under dynamic environments, like variation in RTTs, traffic variation in terms of priority traffic ratio, or where there are short-lived TCP flows. The delay bounds of RIO increase with increasing bandwidth demands (changing network levels). Moreover, the fluctuations in instantaneous queue lengths are also large in RIO leading to higher jitter variations.

The two-level PI controller suffers, in general, with a slow response to regulate the queue under dynamic environments, like time-varying dynamics and when short-lived flows are introduced. Further, it exhibits a poor differentiation between the two drop precedence levels, in the cases where a significant amount of low-priority traffic exists. This is in contrast with the FIO controller that manages the high-priority traffic to utilize as much from the link capacity, fulfilling completely this basic requirement for adequate Diff-Serv congestion control.

In overall, we have demonstrated the effectiveness and robustness of the proposed methodology for Diff-Serv congestion control with simulative evaluation, and at the same time we demonstrated limitations the other AQM schemes have. Thus, from the results presented, the fuzzy logic AQM-based control methodology offers significant improvements on controlling congestion in TCP/IP Diff-Serv networks under widely differing operating conditions, without the need for retuning.

Chapter 10

Concluding Remarks and Future Work

10.1 Concluding Remarks

The complex, but challenging, concept of TCP/AQM congestion control, in both best-effort and Diff-Serv environments, is the key issue of our research study.

A number of representative AQM mechanisms in TCP/IP networks are studied. It is widely accepted that they have serious limitations and drawbacks, including:

- The linearity of the control functions of proposed AQM mechanisms that cannot capture effectively the nonlinearities of the TCP network.
- The dependency of AQM control parameters on dynamic network parameters, like the number of flows and the round trip propagation delays.
- The linearization of the existing models to allow analysis and design of AQM-based controllers, often making stability bounds overly conservative and sluggish performance when dynamic changes occur.
- The accuracy of the existing TCP/AQM models, as they ignore the slow start phase of TCP and/or timeout events that are prominent conditions in today's Internet with the existence of short-lived TCP/Web flows.

It is our thesis that the use of nonlinear control in AQM could lead to efficient and effective control laws. Furthermore, due to the complexity of the dynamic system/network parameters, a robust intelligent control methodology is necessary to effectively control the system under widely varying operating conditions. Thus,

given the need for such control methodology – to capture the dynamics, highly bursty network traffic, and nonlinearities of the TCP/IP system, under widely differing operating conditions – we investigate the usefulness of fuzzy logic control to meet such objectives. Fuzzy logic control is particularly appealing in nonlinear complex systems where satisfactory analytic models are impractical to obtain, but where their behaviour is well understood and can be captured by linguistic models.

In this study we make a significant contribution in formulating an effective and generic AQM control methodology, using fuzzy logic based control, to solve the problem of congestion control in TCP/IP networks.

The potential of Fuzzy logic control methodology to incorporate human knowledge into such a control strategy is demonstrated, and the capability to qualitatively capture the attributes of a control system based on observable phenomena is a main feature of fuzzy logic control and has been shown in the extensive simulative evaluation. Results have been published in various journals and conferences. The proposed Fuzzy Control methodology offers significant improvements in controlling congestion in TCP/IP networks under widely differing operating conditions, without the need for retuning.

More specifically, the main contributions of the thesis in the problem of active queue management congestion control in TCP/IP networks for best-effort and differentiated services can be summarized as follows:

- Formulation of a generic AQM control methodology
    - efficient and effective nonlinear control law
    - simplicity, independent of mathematical models of the controlled system
    - inherent robustness with effective control of the system under widely differing operating conditions, without the need to (re)tune the settings
    - easily adopted in different network environments (best-effort and Diff-Serv architectures).
- Extensive simulative evaluation of proposed mechanism
    - formulated several scenarios with well-known topologies and traffic behavior, e.g. varying round trip delays and number of active flows, as

well as dynamic changes to traffic behavior, inclusion of short-lived and unresponsive flows, and reverse-path traffic.

- Demonstrated that fuzzy logic based AQM control methodology better handles the nonlinearities of the TCP/IP networks, in contrast with existing, well-known, conventional counterparts.
- Offer significant improvements in controlling congestion control in TCP/IP networks by achieving
    - regulated queues
    - bounded delay and delay variation
    - high link utilization
    - minimal packet losses
    - adequate and effective differentiation among different drop precedence's traffic in Diff-Serv networks
    - fast system response
    - robustness to varying system dynamics (differing topologies and traffic conditions).

10.2 Future Work

The design of a generic fuzzy logic control methodology for AQM that is shown to achieve its goals for both best-effort and Diff-Serv environments motivates the adoption of such methodology in other network environments, like mobile/wireless networks, and rate-based multimedia transport framework for TCP-friendly congestion control. Moreover, as currently there is an ongoing research towards enhancing TCP for high-speed networks, we can examine how the fuzzy logic control methodology correlates with these enhancements of TCP.

Braden et al. (1998) indicated the need for router queuing mechanisms to protect responsive flows from other non-responsive traffic. Further, the short-lived TCP flows are more sensitive to packet *marking*/*dropping* than long-lived TCP flows. Thus, the effectiveness of the AQM schemes at providing the necessary fairness on sharing the network resources can be investigated. A well-known fairness numerical

metric can be used proposed by Jain, Chiu, and Hawe (1984) that reflects the fair share distribution across the various connections.

Even though our methodology is demonstrated to behave adequately in a variety of network and traffic conditions, we can add some form of adaptivity to search for provable “optimal” performance. For example, the adaptation, based on formal adaptive control theory, of the output scaling gain can be a subject of future research to improve the responsiveness and accuracy of control, if required, and “ensure” optimum responsiveness under any operating condition.

Further, the use of other adaptive machine learning techniques that can be incorporated in the fuzzy logic based AQM system can be investigated. In particular, the use of neural network principles can be used for selecting the proper adjustment of the fuzzy logic system parameters. Fuzzy Logic and neural networks have been successfully combined in various applications, and can play an important role in the induction of rules and membership function parameters from observations.

An alternative procedure to set the parameters of the fuzzy logic controller (e.g., for deciding the optimal set of rules) can be utilized based on evolutionary algorithms. Evolutionary algorithms provide a universal optimization technique that imitates processes of genetic adaptation that occur in natural evolution. Unlike mathematically more rigorous optimization methods, they require, other than the objective itself, no particular knowledge about the problem structure, such as gradient information. This property makes them applicable to optimization tasks, in which the optimization function is evaluated through experiments or simulations rather than computed directly in closed form. The different approaches, genetic algorithms, evolution strategies, evolutionary programming and genetic programming, are distinguished by the genetic structures that undergo adaptation and the genetic operators by which they generate new variants (Di Fatta, Hoffmann, Lo Re, & Urso, 2003).

Much work remains for the analytical study of fuzzy logic, particularly in the area of stability and performance analysis. Most proposed fuzzy logic controllers in literature do not have any stability analysis because of the difficulty in analysis. This

is mainly due to the existence of the nonlinearity in the control structure that usually makes it difficult to conduct theoretical analysis to explain why fuzzy logic controllers in many instances achieve better performance than the conventional counterparts, especially for highly nonlinear processes. However, as elegantly pointed out by Mamdani (1993), overstressing the necessity of mathematically derived performance evaluations may be counter productive and contrary to normal industry approach (e.g. prototype testing may suffice for accepting the controlled systems performance). Nevertheless, a certain degree of safety concerning fuzzy logic based AQM can be examined.

List of Publications Stemming from the Thesis Study

INTERNATIONAL JOURNALS

1. **Chrysostomou**, C., Pitsillides, A., & Sekercioglu, A. (2006). Fuzzy Explicit Marking: A unified congestion controller for best effort and diff-serv networks. *Submitted for publication in the* ***Computer Networks Journal***.

2. **Chrysostomou**, C., Pitsillides, A., Rossides, L., Polycarpou, M., & Sekercioglu, A. (2003). Congestion Control in Differentiated Services Networks using Fuzzy-RED, ***IFAC Control Engineering Practice (CEP) Journal***, *Vol. 11, Issue 10*, 1153-1170, special Issue on "Control Methods for Telecommunication Networks".

3. **Chrysostomou**, C. Pitsillides, C., Rossides, L., & Sekercioglu, A. (2003). Fuzzy Logic Controlled RED: Congestion Control in TCP/IP Differentiated Services Networks. ***Soft Computing Journal*** *- A Fusion of Foundations, Methodologies and Applications, Vol 8, Number 2,* 79 – 92, special Issue on "The Management of Uncertainty in Computing Applications".

INTERNATIONAL CONFERENCES

1. **Chrysostomou**, C. & Pitsillides, A. (2006). Fuzzy logic congestion control in TCP/IP tandem networks, *Proceedings of the 11th IEEE Symposium on Computers and Communications (****IEEE ISCC 2006****)*, Cagliari, Italy.

2. **Chrysostomou**, C. & Pitsillides, A. (2005). Using Fuzzy Logic Control to Address Challenges in AQM Congestion Control in TCP/IP Networks. *Workshop*

*on Modeling and Control of Complex Systems (**MCCS'05**)*, (CD ROM Proceedings), Ayia Napa, Cyprus.

3. **Chrysostomou**, C., Pitsillides, A., Hadjipollas, G., Polycarpou, M., & Sekercioglu, A. (2004). Congestion Control in Differentiated Services Networks using Fuzzy Logic. *Proceedings of 43rd IEEE Conference on Decision and Control (**IEEE CDC 2004**)*, Bahamas, 549-556 (CD ROM Proceedings - ISBN: 0-7803-8683-3, IEEE Catalog Number: 04CH37601C).
4. **Chrysostomou**, C., Hadjipollas, G., Pitsillides, A. (2004). Fuzzy Logic Control in TCP/IP Networks. *Proceedings of the International Conference on Integrated Modeling & Analysis in Applied Control & Automation (**IMAACA'2004**)*, special session on "Network Control and Network Controlled Systems", Genoa, Italy, 260-269.
5. **Chrysostomou**, C., Pitsillides, A., Hadjipollas, G., Polycarpou, M., & Sekercioglu, A. (2004). Fuzzy Logic Control for Active Queue Management in TCP/IP Networks. *Proceedings of 12th IEEE Mediterranean Conference on Control and Automation (**IEEE MED'04**)*, Kusadasi, Aydin, Turkey, (CD ROM Proceedings).
6. **Chrysostomou**, C., Pitsillides, A., Hadjipollas, G., Sekercioglu, A., & Polycarpou, M. (2004). Fuzzy Logic Congestion Control in TCP/IP Differentiated Services Networks for Quality of Service Provisioning. *Proceedings of the 1st IEEE International Conference on Information & Communication Technologies: from Theory to Applications (**IEEE ICTTA'04**)*, Damascus, Syria, (CD ROM Proceedings – ISBN: 0-7803-8483-0, IEEE Catalog Number: 04EX852C).
7. **Chrysostomou**, C., Pitsillides, A., Hadjipollas, G., Polycarpou, M., & Sekercioglu, A. (2004). Fuzzy Logic Based Congestion Control in TCP/IP Networks for Quality of Service Provisioning. *Proceedings of the International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (**NEW2AN'04**)*, St. Petersburg, Russia, 235-243 (ISBN: 952-15-1132-X).

8. **Chrysostomou**, C., Pitsillides, A., Hadjipollas, G., Sekercioglu, A., & Polycarpou, M. (2003). Fuzzy Logic Congestion Control in TCP/IP Best-Effort Networks. *2003 Australian Telecommunications Networks and Applications Conference (**ATNAC 2003**)*, Melbourne, Australia (CD ROM - ISBN: 0-646-42229-4).
9. **Chrysostomou**, C., Pitsillides, A., Hadjipollas, G., Sekercioglu, A., & Polycarpou, M. (2003). Fuzzy Explicit Marking for Congestion Control in Differentiated Services Networks. *Proceedings of the 8th IEEE Symposium on Computers and Communications (**IEEE ISCC'2003**), Vol. 1*, 312-319, Antalya, Turkey.

Bibliography

Allman, M., Paxson, V., & Stevens, W. (1999). Transmission Control Protocol. *Request for Comments RFC 2581,* Internet Engineering Task Force.

Altman, E., Barakat, C., Mascolo, S., Moller, N., & Sun, J. (2006). Analysis of TCP Westwood+ in high speed networks. *Proceedings of Int. Workshop on protocols for fast long-distance networks, PFLDnet2006*, Nara, Japan.

Andrews, M., & Slivkins, A. (2006). Oscillations with TCP-like flow control in networks of queues. *Proceedings of IEEE Infocom 2006.*

Athuraliya, S. (2002). A note on parameter values of REM with Reno-like algorithms. *http://netlab.caltech.edu/pub/rem.htm*

Athuraliya, S., Li, V. H., Low, S. H., & Yin, Q. (2001). REM: Active Queue Management. *IEEE Network Magazine, 15*(3), 48-53.

Aul, Y.H., Nafaa, A., Negru, D., & Mehaoua, A. (2004). FAFC: Fast adaptive fuzzy AQM controller for TCP/IP netwoks. *Proceedings of Globecom 2004.*

Azvine, B., & Vasilakos, A. (2000). Application of soft computing techniques to the telecommunication domain. *ERUDIT Roadmap*, (G. Tselentis, Ed.), 89-110.

Babiarz, J., Chan, K., & Baker, F. (2006). Configuration guidelines for DiffServ service classes. *Internet Draft,* Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-diffserv-service-classes-02.txt

Bitorika, A., Robin, M., Huggard, M., and Goldrick Mc, C. (2004). A comparative study of active queue management schemes. *Proceedings of ICC'04*.

Blake, S., Black, D, Carlson, M., Davies, E., Wang, Z., & Weiss, W. (1998). An architecture for Differentiated Services. *Request For Comments RFC 2475*, Internet Engineering Task Force.

Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., & Zhang, L. (1998). Recommendations on queue management and congestion avoidance in the Internet. *Request for Comments RFC 2309,* Internet Engineering Task Force.

Braden, B., Zhang, L., Berson, S., Herzog, S., & Jamin, S. (1997). Resource Reservation Protocol (RSVP) – Version 1 Functional Specification". *Request for Comments RFC 2205,* Internet Engineering Task Force.

Braden, R., Clark, D., & Shenker, S. (1994). Integrated Services in the Internet Architecture: An overview. *Request for Comments RFC 1633,* Internet Engineering Task Force.

Brakmo, L., & Peterson, L. (1995). TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal of Selected Areas in Communications,* vol. 13, no. 8, 1465-1480.

Chait, Y., Hollot, C.V., Misra, V., Towsley, D., Zhang, H., &Lui, C.S. (2002). Providing throughput differentiation for TCP flows using adaptive two-color marking and two-level AQM. *Proceedings of INFOCOM'02,* New York.

Chan, K., Babiarz, J., & Baker, F. (2006). Aggregation of DiffServ service classes. *Internet Draft,* Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-chan-tsvwg-diffserv-service-class-aggr-03.txt

Chen, G., Pham, T.T., & Weiss, J.J. (1995). Fuzzy modeling of control systems. *IEEE Transactions on Aerospace and Electronic Systems, vol. 31, no. 1,* 414-429.

Chiu, D-M., & Jain, R. (1989). Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems, 17*, 1-14.

Christiansen, M., et al. (2001). Tuning RED for Web Traffic. *IEEE/ACM Transactions on Networking, vol. 9, no. 3*, 249-64.

Chrysostomou, C. & Pitsillides, A. (2005). Using Fuzzy Logic Control to Address Challenges in AQM Congestion Control in TCP/IP Networks. *Workshop on Modeling and Control of Complex Systems (MCCS'05)*, (CD ROM Proceedings), Ayia Napa, Cyprus.

Chrysostomou, C. & Pitsillides, A. (2006). Fuzzy logic congestion control in TCP/IP tandem networks, *Proceedings of the 11th IEEE Symposium on Computers and Communications (IEEE ISCC'2006),* Cagliari, Italy.

Chrysostomou, C. Pitsillides, C., Rossides, L., & Sekercioglu, A. (2003). Fuzzy Logic Controlled RED: Congestion Control in TCP/IP Differentiated Services Networks. *Soft Computing Journal - A Fusion of Foundations, Methodologies and Applications, Vol 8, Number 2*, 79 – 92, special Issue on "The Management of Uncertainty in Computing Applications".

Chrysostomou, C., Hadjipollas, G., Pitsillides, A. (2004). Fuzzy Logic Control in TCP/IP Networks. *Proceedings of the International Conference on Integrated Modeling & Analysis in Applied Control & Automation (IMAACA'2004),* special session on "Network Control and Network Controlled Systems", Genoa, Italy, 260-269.

Chrysostomou, C., Pitsillides, A., Hadjipollas, G., Polycarpou, M., & Sekercioglu, A. (2004b). Fuzzy Logic Control for Active Queue Management in TCP/IP Networks. *Proceedings of 12th IEEE Mediterranean Conference on Control and Automation (IEEE MED'04),* Kusadasi, Aydin, Turkey, (CD ROM Proceedings).

Chrysostomou, C., Pitsillides, A., Hadjipollas, G., Polycarpou, M., & Sekercioglu, A. (2004c). Congestion Control in Differentiated Services Networks using Fuzzy Logic. *Proceedings of 43rd IEEE Conference on Decision and Control (IEEE CDC 2004),* Bahamas, 549-556 (CD ROM Proceedings - ISBN: 0-7803-8683-3, IEEE Catalog Number: 04CH37601C).

Chrysostomou, C., Pitsillides, A., Hadjipollas, G., Polycarpou, M., & Sekercioglu, A. (2004a). Fuzzy Logic Based Congestion Control in TCP/IP Networks for Quality of Service Provisioning. *Proceedings of the International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN'04),* St. Petersburg, Russia, 235-243 (ISBN: 952-15-1132-X).

Chrysostomou, C., Pitsillides, A., Hadjipollas, G., Sekercioglu, A., & Polycarpou, M. (2003b). Fuzzy Logic Congestion Control in TCP/IP Best-Effort Networks. *2003 Australian Telecommunications Networks and Applications Conference (ATNAC 2003)*, Melbourne, Australia (CD ROM - ISBN: 0-646-42229-4).

Chrysostomou, C., Pitsillides, A., Hadjipollas, G., Sekercioglu, A., & Polycarpou, M. (2004). Fuzzy Logic Congestion Control in TCP/IP Differentiated Services Networks for Quality of Service Provisioning. *Proceedings of the 1st IEEE International Conference on Information & Communication Technologies: from Theory to Applications (IEEE ICTTA'04),* Damascus, Syria, (CD ROM Proceedings – ISBN: 0-7803-8483-0, IEEE Catalog Number: 04EX852C).

Chrysostomou, C., Pitsillides, A., Hadjipollas, G., Sekercioglu, A., & Polycarpou, M. (2003a). Fuzzy Explicit Marking for Congestion Control in Differentiated Services Networks. *Proceedings of the 8th IEEE Symposium on Computers and Communications (IEEE ISCC'2003), Vol. 1,* 312-319, Antalya, Turkey.

Chrysostomou, C., Pitsillides, A., Rossides, L., Polycarpou, M., & Sekercioglu, A. (2003). Congestion Control in Differentiated Services Networks using Fuzzy-RED, *IFAC Control Engineering Practice (CEP) Journal, Vol. 11, Issue 10*, 1153-1170, special Issue on "Control Methods for Telecommunication Networks".

Clark, D., & Fang, W. (1998). Explicit Allocation of Best Effort Packet Delivery Service. *IEEE/ACM Transactions on Networking, 6*(4), 362-373.

Davie, B., Charny, A., Benett, J.C.R., Benson, K., Le Boudec, J.Y., Courtney, W., Davari, S., Sierra, PMC, Firoiu, V., & Stiliadis, D. (2002). An Expedited Forwarding PHB (Per-Hop Behavior). *Request for Comments RFC 3246,* Internet Engineering Task Force.

DCCP-WG, (2001). Datagram Congestion Control Protocol Working Group. http://www.ietf.org/html.charters/dccp-charter.html

Deering, S. & Hinden, R. (1998). Internet Protocol, Version 6 (IPv6) Specification. *Request for Comments RFC 2460,* Internet Engineering Task Force.

Di Fatta, G., Hoffmann, F., Lo Re, G., & Urso, A. (2003). A Genetic Algorithm for the Design of a Fuzzy Controller for Active Queue Management. *IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Computational Intelligence in Telecommunications Networks and Internet Services: Part I, Vol.33, No.3*, 313-324.

DIFFSERV, (1998). Differentiated Services Working Group. Internet Engineering Task Force. https://www1.ietf.org/mailman/listinfo/diffserv

Douligeris, C., & Develekos, G. (1995). A fuzzy logic approach to congestion control in ATM networks. *Proceedings of IEEE ICC'95,* Washington, USA, 1969-1973.

Dualibe, C., Jespers, P., & Verleysen, M. (2000). A 5.26 mflips programmable analogue fuzzy logic controller in a standard cmos 2.4µ technology. *Proceedings of ISCAS 2000,* Geneva, Switzerland, 377-380.

Feng, W. (1999). Improving Internet Congestion Control and Queue Management Algorithms. *PhD Dissertation*, University of Michigan.

Feng, W., Kandlur, D., Saha, D., & Shin, K. (1999a). Blue: A New Class of Active Queue Management Algorithms. *Technical Report UM CSE-TR-387-99*.

Feng, W., Kandlur, D., Saha, D., & Shin, K. (1999b). A self-configuring RED gateway. *Proceedings of IEEE INFOCOM'99*, New York, USA.

Fengyuan, R., Yong, R., & Xiuming, S. (2002). Design of a fuzzy controller for active queue management. *Computer Commmunications, vol 25,* 874-883.

Firoiu, V., & Borden, M. (2000). A study of active queue management for congestion control. *Proceedings of IEEE Infocom'00*, Tel Aviv.

Floyd, S. (2000). Recommendation on using the "gentle_" variant of RED. *http://www.icir.org/floyd/red/gentle.html.*

Floyd, S. (2003). HighSpeed TCP for large congestion windows. *Request for Comments RFC 3649,* Internet Engineering Task Force.

Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, *1*(4), 397-413.

Floyd, S., & Kohler, E. (2006). Profile for datagram congestion control protocol (DCCP). Congestion control ID 2: TCP-like congestion control. *Request for Comments RFC 4341,* Internet Engineering Task Force.

Floyd, S., Allman, M., Jain, A., & Sarolahti, P. (2006). Quick-Start for TCP and IP. *Internet-Draft,* Internet Engineering Task Force.

Floyd, S., Gummadi, R., & Shenker, S. (2001). Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management. *Technical report, ICSI*.

Floyd, S., Handley, M., & Kohler, E. (2006). Problem statement for the datagram congestion control protocol (DCCP). *Request for Comments RFC 4336,* Internet Engineering Task Force.

Floyd, S., Henderson, T., & Gurtov, E.A. (2004). The NewReno modification to TCP's fast recovery algorithm. *Request for Comments RFC 3782,* Internet Engineering Task Force.

Floyd, S., Kohler, E., & Padhye, J. (2006). Profile for datagram congestion control protocol (DCCP). Congestion control ID 3: TCP-friendly rate control. *Request for Comments RFC 4342,* Internet Engineering Task Force.

Fuzzy Logic Toolbox. (2002). User's Guide, *by The MathWorks, Inc.* http://www.mathworks.com/access/helpdesk/help/toolbox/fuzzy/

Grieco, L.A. & Mascolo, S. (2003). End-to-end bandwidth estimation for congestion control in packet networks. *International Workshop QoS-IP,* Milano, Italy.

Guirguis, M., Bestavros, A., & Matta, I. (2003). Exogenous-Loss Awareness in Queue Management – Towards Global Fairness. *Techical Report*, Computer Science Departrment, Boston University.

Guo, L., & Matta, I. (2001). The War Between Mice and Elephants. *IEEE ICNP'01*.

Habetha, J., & Walke, B. (2002). Fuzzy rule-based mobility and load management for self-organizing wireless networks. *International journal of wireless information networks, vol. 9, no. 2,* 119-140.

Handley, M., Floyd, S., Padhye, J., & Widmer, J. (2003). TCP Friendly Rate control (TFRC): Protocol Specification. *Request for Comments RFC 3448,* Internet Engineering Task Force.

Hassan, M., & Sirisena, H. (2001). Optimal control of queues in computer networks. *IEEE International Conference on Communications.*

Heinanen, J., Baker, F., Weiss, W., & Wroclawski (1999). Assured Forwarding PHB Group. *Request for Comments RFC 2597,* Internet Engineering Task Force.

Hengartner, U., Bolliger, J., & Gross, T. (2000). TCP Vegas revisited. *Proceedings of IEEE Infocom 2000.*

Herzog, S. (2000). RSVP extensions for policy control. *Request for Comments RFC 2750,* Internet Engineering Task Force.

Hollot, C. V., Misra, V., Towsley, D., & Gong, W.B. (2001). A control theoretic analysis of RED. *Proceedings of IEEE Infocom'01.*

Hollot, C. V., Misra, V., Towsley, D., & Gong, W.-B. (2002). Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. *IEEE Transactions on Automatic Control, vol. 47, no. 6*, 945-959.

Iannaccon, G., Brandauer, C., Ziegler, T., Diot, C., Fdida, S., & May, M. (2001). Tail Drop and Active Queue Management Performance for bulk-data and Web-like Internet Traffic. *6th IEEE Symposium on Computers and Communications.* Hammamet, Tynisia.

ICCRG, (2006). Internet Congestion Control Research Group. http://oakham.cs.ucl.ac.uk/mailman/listinfo/iccrg

Jacobson, V. (1988). Congestion avoidance and control. *Proceedings of ACM SIGCOMM 1998,* 314-329.

Jacobson, V., Nichols, K., & Poduri, K. (1999). An Expedited Forwarding PHB. *Request for Comments RFC 2598,* Internet Engineering Task Force.

Jain, R., Chiu, D, & Hawe, W. (1984). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Digital Equipment Corporation, Technical Report DEC-TR-301*.

Jin, C., Wei, D.X., & Low, S.H. (2004). FAST TCP: motivation, architecture, algorithms, performance. *IEEE Infocom 2004.*

Katabi, D., Handley, M., & Rohrs, C. (2002). Congestion control for high bandwidth-delay product networks. *Proceedings of ACM SIGCOMM 2002*.

Kelly, T. (2003). Scalable TCP: Improving performance on highspeed wide area networks. *ACM computer communication review*.

Keshav, S. (1991). Congestion Control in Computer Networks. *Ph.D. thesis*, University of California Berkeley.

Khalifa, I., & Trajkovic, L. (2004). An Overview and Comparison of Analytical TCP Models. *ISCAS 2004.*

Kohler, E., Handley, M., & Floyd, S. (2006). Datagram congestion control protocol (DCCP). *Request for Comments RFC 4340,* Internet Engineering Task Force.

Kohler, S., Menth, M., & Vicari, N. (2000). Analytic Performance Evaluation of the RED Algorithm for QoS in TCP/IP. *Research Report Series no. 259*, University of Wurzburg, Institute of Computer Science.

Kunniyur, S., & Srikant, R. (2003). End-to-end congestion control: utility functions, random losses and ECN marks. *IEEE/ACM Transactions on Networking*.

Kunniyur, S., & Srikant, R. (2004). An adaptive virtual queue (AVQ) algorithm for active queue management. *IEEE/ACM Transactions on Networking, vol 12, no. 2,* 286-299.

Kurose, J.F., & Ross, K.W. (2005). Computer networking: a top-down approach featuring the Inernet. *Addison-Wesley.* ISBN: 0-321-26976-4

Lakshman, T.V., & Madhow, U. (1997). The performance of TCP/IP for networks with high bandwidth delay products and random loss. *IEEE/ACM Transactions on Networking, vol. 5,* 336-350.

Lee, C. (1990). Fuzzy logic in control systems: Fuzzy logic controller – Parts I-II. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 20, no. 2,* 404-435.

Leith, D., & Shorten, R. (2004). H-TCP protocol for high-speed long distance networks. *Proceedings of Int. Workshop on protocols for fast long-distance networks (PFLDnet 2004).*

Lestas, M.C. (2005). Intelligent congestion control for computer networks. *PhD Thesis.* University of Southern California.

Li, Z., Zhang, Z., Addie, R., & Clerot, F. (2003). Improving the Adaptability of AQM Algorithms to Traffic Load Using Fuzzy Logic. *2003 Australian Telecommunications Networks and Applications Conference (ATNAC 2003)*, Melbourne, Australia.

Low, S., Paganini, F., Wang, J., Adlakha, S., & Doyle, J. (2002). Dynamics of TCP/RED and a Scalable Control. *IEEE Infocom 2002*, New York.

Low, S., Paganini, F., Wang, J., Adlakha, S., & Doyle, J. (2003). Linear Stability of TCP/RED and a Scalable Control. *Computer Networks Journal, 43(5)*, 633-647.

Mamdani, E.H. (1974). Applications of fuzzy algorithms for simple dynamic plant. *Proceedings of IEE, 121(12)*, 1585-1588.

Mamdani, E.H. (1993). Twenty years of fuzzy logic: experiences gained and lessons learned. *IEEE International conference on fuzzy systems,* 339-344, San Franscisco.

Mamdani, E.H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, *vol. 7, no. 1,* 1-13.

Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M., & Wang, R. (2001). TCP Westwood: End-to-end bandwidth estimation for efficient transport over wired and wireless networks. *Proceedings of ACM Mobicom 2001.*

Mathis, M., Mahdavi, J., Floyd, S., & Romanow, A. (1996). TCP Selective Acknowledgement options. *Request for Comments RFC 2018,* Internet Engineering Task Force.

May, M., Bolot, J., Diot, C., & Lyles, B. (1999). Reasons Not to Deploy RED. *7th International Workshop on Quality of Service, IWQoS'99*, 260-262.

May, M., Bolot, J.C., Jean-Marie, A., & Diot, C. (1999). Simple perfomance models of differentiated services schemes for the Internet. *Proceedings of INFOCOM'99*, New York.

May, M., Bonald, T., & Bolot, J.C. (2000). Analytic Evaluation of RED Performance. *Proceedings of IEEE Infocom 2000*. Tel Aviv.

Misra, V., Gong, W.B , & Towsley, D. (2000). Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. *ACM/SIGCOMM'00,* 151-160.

Morales, E., Polycarpou, M., Hemasilpin, N., & Bissler, J. (2001). Hierarchical Adaptive and Supervisory Control of Continuous Venovenous Hemofiltration. *IEEE Transactions on Control Systems Technology, Vol. 9, No. 3*, 445-457.

Network Simulator. (1989). *NS-2*, http://nsnam.isi.edu/nsnam/.

Nichols, K., & Carpenter, B. (2001). Definition of differentiated services per domain behaviors and rules for their specification. *Request for Comments RFC 3086,* Internet Engineering Task Force.

Nichols, K., Blake, S., Baker, F., & Black, D. (1998). Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 headers. *Request for Comments RFC 2474,* Internet Engineering Task Force.

Oliveira, R. & Braun, T. (2004). A delay-based approach using fuzzy logic to improve TCP error detection in ad hoc networks. *In Proceedings of IEEE Wireless Communications and Networking conference (WCNC04),* Atlanta, USA.

Ott, T.J., Lakshman, T.V., & Wong, L.H. (1999). SRED: Stabilized RED. *Proceedings of IEEE INFOCOM'99*, New York, USA.

Padhye, J. (2000). Model-based approach to TCP-friendly congestion control. *PhD Thesis,* University of Massachusetts.

Padhye, J., Firoiu, V., Towsley, D.F., & Kurose, J.F. (2000). Modeling TCP Reno Performance: A Simple Model and its Empirical Validation. *IEEE/ACM transactions on Networking, vol. 8, no. 2*, 133-145.

Passino, K., & Yurkovich, M. (1998). Fuzzy Control. *Ed. Prentice Hall, ISBN 0-201-18074-X*.

Peeters, S., & Blondia, C. (1999). A discrete time analysis of random early detection with responsive best-effort traffic. *Technical Report 257TD(99)29*, COST-257, MC meeting, Cyprus.

Pentikousis, K. (2001). Active Queue Management. *ACM student magazine Crossroads, Connector columns, Tutorials.*

Pitsillides, A., & Sekercioglu, A. (2000). Congestion Control. In Pedrycz, W. & Vasilakos, A. V. (Eds.), *Computational Intelligence in Telecommunications Networks* (pp. 109-158). Boca Raton, FL: CRC Press, ISBN: 0-8493-1075-X.

Pitsillides, A., Sekercioglou, A., & Ramamurthy, G. (1997). Effective Control of Traffic Flow in ATM Networks Using Fuzzy Explicit Rate Marking (FERM). *IEEE Journal on Selected Areas in Communications (JSAC), vol. 15, issue 2,* 209-225.

Plasser, E., & Ziegler, T. (2004). A RED Function Design Targeting Link Utilization and Stable Queue Size Behaviour. *Computer Networks Journal, issue 44,* 383-410.

Postel, J. (1981). Internet Protocol. *Request for Comments RFC 791,* Internet Engineering Task Force.

Ramakrishnan, K., Floyd, S., & Black, D. (2001). The addition of explicit congestion notification (ECN) to IP. *Request for Comments RFC 3168,* Internet Engineering Task Force.

Ranjan, P., Abed, E.H., & La, R.J. (2004). Nonlinear Instabilities in TCP-RED. *IEEE/ACM Transactions on Networking, vol. 12*, no. 6.

Rossides, L., Chrysostomou, C., Pitsillides, A., & Sekercioglu, A. (2002). Overview of Fuzzy-RED in Diff-Serv Networks. *Lecture Notes in Computer Science, Publisher: Springer-Verlag Heidelberg, ISSN: 0302-9743, Volume 2311 / 2002, Title: Soft-Ware 2002: Computing in an Imperfect World : First International Conference (Soft-Ware), D. Bustard, W. Liu, R. Sterritt (Eds)*, Belfast, Northern Ireland, 1-13.

Rossides, L., Sekercioglu, A., Kohler, S., Pitsillides, A., Phuoc, T-G., & Vassilakos, A. (2000). Fuzzy Logic Controlled RED: Congestion Control for TCP/IP Diff-Serv Architecture. *ESIT2000, 8th European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany, 263-269.

Ryu, S., Rump, C., & Qiao, C. (2003). Advances in Internet congestion control. *IEEE Communications Surveys & Tutorials,* Third Quarter 2003, vol 5, no 1, 28-39.

Savoric, M. (2003). Fuzzy explicit window adaptation: a method to further enhance TCP performance, *Technical Report TKN-03-010,* Telecommunication Networks Group, Technical University Berlin.

Schwartz, M. (1988). Telecommunication networks: Protocols, modelling, analysis. *Addison Wesley*.

Sekercioglou, A., Pitsillides, A., & Egan, G.K. (1994). Study of an adaptive fuzzy controller based on the adaptation of relative rule weights. *Proceedings of ANZIIS'94*, 204-208, Brisbane, Queensland, Australia,.

Sekercioglu, A., Pitsillides, A., & Vasilakos, A. (2001). Computational intelligence in management of ATM networks. *Soft Computing Journal, 5(4),* 257-263.

Shenker, S., Zhang, L., & Clark, D.D. (1990). Some observation on the dynamics of a congestion control algorithm. *Computer Communications Review,* 30-39.

Siripongwutikorn, P., Banerjee, S., & Tipper, D. (2002). Adaptive bandwidth control for efficient aggregate QoS provisioning. *Proceedings of Globecom 2002.*

Stevens, W. (1994). TCP/IP illustrated, Volume I: the protocols. *Reading, MA: Addison-Wesley.*

Stevens, W. (1997). TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *Request for Comments RFC 2001,* Internet Engineering Task Force.

TMRG, (2006). The Transport Modeling Research Group. http://www.icir.org/tmrg/

Wang Y. T. (2004). A Dynamic Channel Borrowing Approach with Fuzzy Logic Control in Distributed Cellular Networks. *In the special issue of Simulation Modeling Practice and Theory, vol. 12*, 287–303

Wang, C., Li, B., Hou, Y.T., Sohraby, K., & Lin, Y. (2004). LRED: A Robust Active Queue Management Scheme Based on Packet Loss Ratio. *IEEE Infocom'04.*

Wang, C., Li, B., Sohraby, K., & Peng, Y. (2003). AFRED: An adaptive fuzzy-based control algorithm for active queue management. *Proceedings of the 28th IEEE International Conference on Local Computer Networks (LCN'03).*

Yang, C.Q., & Reddy, A.V.S. (1995). A taxonomy for congestion control algorithms in packet switching networks. *IEEE Network Magazine*.

Yasunobu, S., & Miyamoto, S. (1985). Automatic Train Operation by Predictive Fuzzy Control. *In Industrial Applicatons of Fuzzy Control*, (M. Sugeno, Ed.), 1-18, Elsevier Science Publishers.

Zadeh, L. A. (1965). Fuzzy Sets. *Information And Control, vol. 8*, 338-353.

Zadeh, L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics, 3(1),* 28-44.

Ziegler, T., Fdida, S., & Brandauer, C. (2001). Stability Criteria of RED with TCP Traffic. *IFIP ATM & IP Working Conference*, Budapest.

Appendix A

Linguistic Rules of the Fuzzy Logic based Control Methodology

1. IF queue-error is negative-very-big and the previous-queue-error is negative-very-big THEN mark-probability is huge

2. IF queue-error is negative-very-big and the previous-queue-error is negative-big THEN mark-probability is huge

3. IF queue-error is negative-very-big and the previous-queue-error is negative-small THEN mark-probability is huge

4. IF queue-error is negative-very-big and the previous-queue-error is zero THEN mark-probability is huge

5. IF queue-error is negative-very-big and the previous-queue-error is positive-small THEN mark-probability is huge

6. IF queue-error is negative-very-big and the previous-queue-error is positive-big THEN mark-probability is huge

7. IF queue-error is negative-very-big and the previous-queue-error is positive-very-big THEN mark-probability is huge

8. IF queue-error is negative-big and the previous-queue-error is negative-very-big
THEN mark-probability is big

9. IF queue-error is negative-big and the previous-queue-error is negative-big
THEN mark-probability is big

10. IF queue-error is negative-big and the previous-queue-error is negative-small
THEN mark-probability is big

11. IF queue-error is negative-big and the previous-queue-error is zero
THEN mark-probability is very-big

12. IF queue-error is negative-big and the previous-queue-error is positive-small
THEN mark-probability is very-big

13. IF queue-error is negative-big and the previous-queue-error is positive-big
THEN mark-probability is huge

14. IF queue-error is negative-big and the previous-queue-error is positive-very-big
THEN mark-probability is huge

15. IF queue-error is negative-small and the previous-queue-error is negative-very-big
THEN mark-probability is tiny

16. IF queue-error is negative-small and the previous-queue-error is negative-big
THEN mark-probability is very-small

17. IF queue-error is negative-small and the previous-queue-error is negative-small
THEN mark-probability is small

18. IF queue-error is negative-small and the previous-queue-error is zero
THEN mark-probability is small

19. IF queue-error is negative-small and the previous-queue-error is positive-small

THEN mark-probability is big

20. IF queue-error is negative-small and the previous-queue-error is positive-big
THEN mark-probability is very-big

21. IF queue-error is negative-small and the previous-queue-error is positive-very-big
THEN mark-probability is very-big

22. IF queue-error is zero and the previous-queue-error is negative-very-big
THEN mark-probability is zero

23. IF queue-error is zero and the previous-queue-error is negative-big
THEN mark-probability is zero

24. IF queue-error is zero and the previous-queue-error is negative-small
THEN mark-probability is zero

25. IF queue-error is zero and the previous-queue-error is zero
THEN mark-probability is tiny

26. IF queue-error is zero and the previous-queue-error is positive-small
THEN mark-probability is very-small

27. IF queue-error is zero and the previous-queue-error is positive-big
THEN mark-probability is small

28. IF queue-error is zero and the previous-queue-error is positive-very-big
THEN mark-probability is big

29. IF queue-error is positive-small and the previous-queue-error is negative-very-big
THEN mark-probability is zero

30. IF queue-error is positive-small and the previous-queue-error is negative-big
THEN mark-probability is zero

31. IF queue-error is positive-small and the previous-queue-error is negative-small
THEN mark-probability is zero

32. IF queue-error is positive-small and the previous-queue-error is zero
THEN mark-probability is zero

33. IF queue-error is positive-small and the previous-queue-error is positive-small
THEN mark-probability is tiny

34. IF queue-error is positive-small and the previous-queue-error is positive-big
THEN mark-probability is tiny

35. IF queue-error is positive-small and the previous-queue-error is positive-very-big
THEN mark-probability is very-small

36. IF queue-error is positive-big and the previous-queue-error is negative-very-big
THEN mark-probability is zero

37. IF queue-error is positive-big and the previous-queue-error is negative-big
THEN mark-probability is zero

38. IF queue-error is positive-big and the previous-queue-error is negative-small
THEN mark-probability is zero

39. IF queue-error is positive-big and the previous-queue-error is zero
THEN mark-probability is zero

40. IF queue-error is positive-big and the previous-queue-error is positive-small
THEN mark-probability is zero

41. IF queue-error is positive-big and the previous-queue-error is positive-big
THEN mark-probability is zero

42. IF queue-error is positive-big and the previous-queue-error is positive-very-big

THEN mark-probability is tiny

43. IF queue-error is positive-very-big and the previous-queue-error is negative-very-big
THEN mark-probability is zero

44. IF queue-error is positive-very-big and the previous-queue-error is negative-big
THEN mark-probability is zero

45. IF queue-error is positive-very-big and the previous-queue-error is negative-small
THEN mark-probability is zero

46. IF queue-error is positive-very-big and the previous-queue-error is zero
THEN mark-probability is zero

47. IF queue-error is positive-very-big and the previous-queue-error is positive-small
THEN mark-probability is zero

48. IF queue-error is positive-very-big and the previous-queue-error is positive-big
THEN mark-probability is zero

49. IF queue-error is positive-very-big and the previous-queue-error is positive-very-big
THEN mark-probability is zero

Appendix B

FEM Simulation Results

Table B.1 Control parameter values of selected AQM mechanisms

A-RED Floyd, Gummadi, and Shenker (2001)	**PI** Hollot, Misra, Towsley, and Gong (2002)	**REM** Athuraliya, Li, Low, and Yin (2001)	**AVQ** Kunniyur and Srikant (2004)
use *gentle* mode	$a = 1.822(10)^{-5}$ $b = 1.816(10)^{-5}$	$\alpha = 0.1$ $\gamma = 0.001$ $\varphi = 1.001$	$\gamma = 1.0$ $\alpha = 0.15$

Table B.2 Distributions and parameters for Web Traffic

	Inter-page Time	**Objects per page**	**InterObject time**	**Object Size**
Distribution	Pareto	Pareto	Pareto	Pareto
Mean	50 msec	4 msec	0.5 msec	12 KB
Shape	2	1.2	1.5	1.2

Table B.3 Summary of statistical results – Scenarios I

TQL=200: expected mean delay = 106.67 msec					
Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**	**Utilization (%)**
		Mean-Delay	Std-Deviation		
	FEM	106.12	12.56	0	99.82
	PI	119.87	32.94	0.42	99.02
I-1	**A-RED**	107.06	22.74	0.39	98.97
	REM	109.85	27.11	0.4	99.05
	AVQ	47.84	36.7	0.26	98.32
	FEM	106.18	18.14	0.009	99.85
	PI	132.37	65.02	0.49	99.19
I-2	**A-RED**	110.06	29.13	0.55	99.12
	REM	113.52	39.57	0.45	99.21
	AVQ	52.37	58.22	0.24	98.32
	FEM	106.27	12.69	0	99.97
	PI	142.27	66.68	0.54	99.43
I-3	**A-RED**	111.37	22.25	3.69	96.36
	REM	116.5	44.09	0.38	99.58
	AVQ	37.84	41.81	0.18	99.02

Table B.3 Summary of statistical results – Scenarios I (continued)

TQL=200: expected mean delay = 106.67 msec					
Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**	**Utilization (%)**
		Mean-Delay	Std-Deviation		
I-4 (Bottleneck prop. delay = 30 msec)	**FEM**	108.61	13.61	0	99.92
	PI	136.31	63.28	0.49	99.36
	A-RED	111.65	26.19	0.55	99.28
	REM	114.67	41.08	0.41	99.43
	AVQ	49.69	50.56	0.23	98.47
I-4 (Bottleneck prop. delay = 60 msec)	**FEM**	106.52	18.17	0	99.81
	PI	127.85	65.31	0.48	98.8
	A-RED	105.56	31.54	0.47	98.88
	REM	112.31	39.06	0.44	98.84
	AVQ	81.97	69.78	0.31	98.07
I-4 (Bottleneck prop. delay =120 msec)	**FEM**	101.8	24.59	0.05	99.18
	PI	119.51	54.8	0.57	96.67
	A-RED	106.53	72.84	0.66	97.24
	REM	108.77	47.79	0.54	96.85
	AVQ	112.99	78.34	0.5	97.37

Table B.3 Summary of statistical results – Scenarios I (continued)

TQL=200: expected mean delay = 106.67 msec					
Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**	**Utilization (%)**
		Mean-Delay	Std-Deviation		
I-5 (Traffic Load=200)	**FEM**	106.98	21.32	0	99.35
	PI	144.99	85.65	1	96.25
	A-RED	113.78	53.32	1.73	97.13
	REM	116.29	50.17	0.82	97.59
	AVQ	99.89	82.42	0.71	97.45
I-5 (Traffic Load=300)	**FEM**	111.53	24.93	0.046	99.47
	PI	168.22	96.26	1.68	96.85
	A-RED	121	53.78	7.17	92.33
	REM	125.4	63.1	0.99	96.66
	AVQ	84.86	82.31	0.67	97.68
I-5 (Traffic Load=400)	**FEM**	117.47	30.81	0.27	99.21
	PI	183.28	99.53	2.8	96.68
	A-RED	136.37	58.47	9.63	89.96
	REM	134.92	75.57	1.52	95.65
	AVQ	91.99	85.98	0.82	97.63

Table B.3 Summary of statistical results – Scenarios I (continued)

TQL=200: expected mean delay = 106.67 msec					
Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**	**Utilization (%)**
		Mean-Delay	Std-Deviation		
I-5 (Traffic Load=500)	**FEM**	119.87	32.56	0.44	99.11
	PI	194.9	94.08	4.24	95.55
	A-RED	136.98	62.24	12.79	86.81
	REM	143.33	82.23	2.19	93.87
	AVQ	97.97	92.82	0.74	97.96
I-6	**FEM**	106.77	18.22	0.04	99.61
	PI	134.9	41.26	0.62	98.25
	A-RED	112.07	46.43	0.81	97.89
	REM	112.96	34.11	0.62	98.2
	AVQ	110.67	77.38	0.45	97.92

Table B.3 Summary of statistical results – Scenarios I (continued)

TQL=200: expected mean delay = 106.67 msec					
Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**	**Utilization (%)**
		Mean-Delay	Std-Deviation		
I-7 (reverse web-traffic)	**FEM**	100.33	24.6	0.07	99.12
	PI	115.25	55.6	0.7	96.58
	A-RED	104.3	73.7	0.68	97.17
	REM	105.86	50.44	0.56	96.78
	AVQ	69.79	61.25	0.43	94.96
I-7 (reverse web-traffic + FTP)	**FEM**	95.93	26.61	0.1	98.95
	PI	98.67	44.98	0.96	95.6
	A-RED	78.91	57.5	1.52	87.95
	REM	80.61	43.51	0.78	93.59
	AVQ	31.95	44.21	0.43	81.58
I-8	**FEM**	92.59	24.81	0.23	99.12
	PI	95.86	45.64	1.04	95.3
	A-RED	81.68	64.33	1.59	83.6
	REM	78.3	44.29	0.82	92.44
	AVQ	5.86	25.14	0.4	43.92

Table B.4 Summary of statistical results – Scenarios II

TQL=200: expected mean delay = 35.56 msec					
Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**	**Utilization (%)**
		Mean-Delay	Std-Deviation		
II-1 (congestion at peripheral link)	**FEM**	32.68	7.87	0	99.61
	PI	44.33	17.01	0.46	99.31
	A-RED	37.2	7.64	0.28	99.46
	REM	37.3	12.34	0.4	99.4
	AVQ	22.12	20.02	0.22	98.51
II-2 (congestion at peripheral link + short-lived flows)	**FEM**	32.55	7.87	0	99.61
	PI	45.62	17.11	0.49	99.23
	A-RED	34.87	11.83	0.34	99.39
	REM	37.54	8.32	0.44	99.34
	AVQ	19.45	20.73	0.22	98.42

Table B.5 Summary of statistical results – Scenarios III

TQL=200: expected mean delay = 106.67 msec					
Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**	**Utilization (%)**
		Mean-Delay	Std-Deviation		
III-1	**FEM**	109.61	15.42	0.074	99.84
	PI	152.58	75.93	0.91	98.96
	A-RED	112.12	25.14	5.73	94.25
	REM	119.17	52.0	0.62	99.23
	AVQ	37.11	40.88	0.25	98.8
III-2	**FEM**	111.88	21.05	0.12	99.71
	PI	203.81	74.91	2.27	97.94
	A-RED	154.90	43.65	12.51	87.33
	REM	137.73	78.98	1.31	98.07
	AVQ	75.46	85.59	0.17	99.0
III-3	**FEM**	114.56	20.27	0.19	99.91
	PI	224.65	55.35	3.56	96.9
	A-RED	156.46	46.63	13.47	86.73
	REM	143.95	81.82	1.65	97.89
	AVQ	52.18	45.9	0.29	97.89

Table B.5 Summary of statistical results – Scenarios III (continued)

TQL=200: expected mean delay = 106.67 msec					
Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**	**Utilization (%)**
		Mean-Delay	Std-Deviation		
III-4 (Bottleneck prop. delay at router-D and router-E = 120 msec)	**FEM**	116.83	36.32	0.16	99.4
	PI	186.39	91.92	2.26	97.91
	A-RED	134.14	60.13	9.68	90.25
	REM	134.88	76.02	1.43	96.46
	AVQ	82.92	71.08	0.64	98.15
III-4 (Bottleneck prop. Delay at router-D and router-E = 200 msec)	**FEM**	110.66	41.83	0.04	99.04
	PI	169.37	101.56	2.36	95.2
	A-RED	135.48	79.91	7.65	90.99
	REM	125.97	72.89	1.71	95.23
	AVQ	95.9	81.07	0.66	97.53
III-5	**FEM**	117.87	27.11	0.09	98.55
	PI	210.95	65.70	1.86	95.2
	A-RED	177.53	33.73	12.39	90.99
	REM	135.5	51.81	1.08	95.24
	AVQ	92.67	51.17	0.24	97.53

Appendix C

FIO Simulation Results

Table C.1 Summary of statistical results – Scenarios I

TQL=100: expected mean delay = 53.33 msec

TQL=200: expected mean delay = 106.67 msec

Scenarios	AQM*	Delay (ms)		Loss Rate (%)			Utilization (%)		
		Mean-Delay	Std-Deviation	Low-priority	High-priority	Total	Low-priority	High-priority	Total
	FIO	62.61	23.05	0.47	0.019	0.25	49.69	49.33	99.02
I-1	**TL-PI**	77.32	41.56	0.57	0.11	0.5	83.89	14.83	98.72
	RIO	178.52	79.72	1.66	1.72	1.67	94.31	2.27	96.58
	FIO	84.27	28.59	1.285	0.084	0.26	14.42	84.6	99.02
I-2	**TL-PI**	103.04	36.44	0.65	0.23	0.5	62.49	36.23	98.72
	RIO	112.86	55.97	5.77	0.22	2.97	44.67	50.8	95.47
	FIO	110.89	25.15	2.24	0.45	0.47	0.93	97.25	98.18
I-3	**TL-PI**	116.86	39.15	0.72	0.49	0.5	4.79	93.23	98.02
	RIO	158.8	46.22	15.12	1.12	1.22	0.72	96.52	97.24

* we designate the two-level PI AQM controller as TL-PI, for easier convenience

Table C.1 Summary of statistical results – Scenarios I (continued)

TQL=100: expected mean delay = 53.33 msec

TQL=200: expected mean delay = 106.67 msec

Scenarios	AQM	Delay (ms)		Loss Rate (%)			Utilization (%)		
		Mean-Delay	Std-Deviation	Low-priority	High-priority	Total	Low-priority	High-priority	Total
I-4	**FIO**	84.55	34.93	0.39	0.15	0.27	48.78	46.58	95.36
	TL-PI	82.35	49.59	0.5	0.5	0.5	78.55	15.76	94.31
	RIO	87.17	79.7	3.7	0.435	2.84	65.61	25.43	91.04
I-5	**FIO**	113.74	25.12	1.8	0.41	0.41	0.28	99.07	99.35
	TL-PI	121.99	39.04	0.59	0.41	0.42	6.43	92.6	99.03
	RIO	164.71	51.87	18.89	2.02	2.12	0.47	97.28	97.75

Table C.1 Summary of statistical results – Scenarios I (continued)

TQL=100: expected mean delay = 53.33 msec

TQL=200: expected mean delay = 106.67 msec

Scenarios	AQM	Delay (ms)		Loss Rate (%)			Utilization (%)		
		Mean-Delay	Std-Deviation	Low-priority	High-priority	Total	Low-priority	High-priority	Total
I-6 (bottleneck prop. delay = 30msec)	**FIO**	117.85	25.16	3.13	0.30	0.32	0.78	98.73	99.51
	TL-PI	140.59	45.81	0.5	0.33	0.35	6.79	92.34	99.13
	RIO	170.28	30.21	15.3	3.53	3.59	0.43	95.84	96.27
I-6 (bottleneck prop. delay = 60msec)	**FIO**	113.94	25.42	4.1	0.41	0.43	0.65	98.48	99.13
	TL-PI	128.83	39.76	0.54	0.43	0.44	6.25	92.76	99.01
	RIO	166.2	50.78	16.2	2.42	2.5	0.64	96.49	97.13
I-7	**FIO**	112.88	28.56	2.21	0.45	0.47	0.69	97.16	98.03
	TL-PI	123.29	42.47	0.79	0.58	0.59	4.31	93.34	97.65
	RIO	160.61	69.2	15.42	1.88	1.98	0.87	95.79	96.66

Table C.1 Summary of statistical results – Scenarios I (continued)

TQL=100: expected mean delay = 53.33 msec

TQL=200: expected mean delay = 106.67 msec

Scenarios	AQM	Delay (ms)		Loss Rate (%)			Utilization (%)		
		Mean-Delay	Std-Deviation	Low-priority	High-priority	Total	Low-priority	High-priority	Total
I-8	**FIO**	85.69	28.57	4.83	0.11	0.91	18.11	80.7	98.81
	TL-PI	75.05	40.51	1.92	0.35	1.54	65.78	29.8	95.58
	RIO	92.03	56.32	7.5	0.55	5.19	45.89	46.97	92.86
I-9	**FIO**	188.38	68.07	1.81	0.47	0.48	1.0	98.3	99.8
	TL-PI	201.90	116.21	0.72	1.59	1.52	12.29	87.3	99.6
	RIO	231.37	108.35	17.1	0.56	2.17	10.4	89.4	99.8

Table C.2 Summary of statistical results – Scenarios II

TQL=100: expected mean delay = 53.33 msec

TQL=200: expected mean delay = 106.67 msec

Scenarios	AQM	Delay (ms)		Loss Rate (%)			Utilization (%)		
		Mean-Delay	Std-Deviation	Low-priority	High-priority	Total	Low-priority	High-priority	Total
	FIO	63.29	24.07	0.13	0	0.09	62.1	37.6	99.7
II-1	**TL-PI**	72.07	37.02	0.32	0	0.32	98.45	0.96	99.41
	RIO	218.52	48.85	1.68	0	1.68	97.38	0.16	97.54
	FIO	94.56	26.71	0.87	0.03	0.09	6.85	92.82	99.67
II-2	**TL-PI**	103.87	31.43	0.43	0.11	0.32	65.56	33.85	99.41
	RIO	105.07	49.94	8.65	0.07	3.05	29.53	66.78	96.3
	FIO	110.36	24.8	0	0.34	0.34	0.39	99.09	99.48
II-3	**TL-PI**	117.68	30.74	0.08	0.36	0.35	2.04	97.33	99.37
	RIO	162.38	37.68	21.21	1.33	1.34	0.58	97.29	97.87

Table C.2 Summary of statistical results – Scenarios II (continued)

TQL=100: expected mean delay = 53.33 msec

TQL=200: expected mean delay = 106.67 msec

Scenarios	**AQM**	**Delay (ms)**		**Loss Rate (%)**			**Utilization (%)**		
		Mean-Delay	Std-Deviation	Low-priority	High-priority	Total	Low-priority	High-priority	Total
II-4	**FIO**	86.75	36.08	0.18	0.049	0.099	37.15	59.5	96.65
	TL-PI	84.8	43.46	0.34	0.2	0.32	81.66	14.73	96.39
	RIO	80.78	70.90	5.87	0.11	3.37	50.2	43.24	93.44
II-5 (bottleneck prop. delay = 30msec)	**FIO**	66.54	24.46	0.23	0	0.14	59.61	40.14	99.75
	TL-PI	88.05	47.23	0.33	0	0.33	98.5	1.04	99.54
	RIO	227.77	42.53	1.85	3.65	1.86	97.43	0.41	97.84
II-5 (bottleneck prop. delay = 120msec)	**FIO**	57.19	22.94	0.29	0	0.24	75.8	23.0	98. 80
	TL-PI	61.02	39.12	0.51	0.07	0.51	97.59	0.76	98.35
	RIO	187.48	65.64	1.35	0	1.35	94.65	0.18	94.83

Table C.2 Summary of statistical results – Scenarios II (continued)

TQL=100: expected mean delay = 53.33 msec

TQL=200: expected mean delay = 106.67 msec

Scenarios	AQM	Delay (ms)		Loss Rate (%)			Utilization (%)		
		Mean-Delay	Std-Deviation	Low-priority	High-priority	Total	Low-priority	High-priority	Total
II-6	**FIO**	113.79	27.43	0	0.32	0.32	0.09	99.21	99.30
	TL-PI	130.44	32.01	0	0.53	0.53	2.10	97.04	99.14
	RIO	164.89	48.27	20.48	2.46	2.48	0.62	96.69	97.31
II-7	**FIO**	57.15	16.94	0.009	0	0.007	16.36	82.37	98.73
	TL-PI	55.79	27.0	0.03	0.01	0.02	80.99	17.66	98.65
	RIO	78.80	61.03	0.24	0.01	0.19	20.66	77.59	98.25