Techniques in Internet Congestion Control

Bartek Peter Wydrowski

Submitted for examination for the fulfilment of the degree of
Doctor of Philosophy

February 2003

Electrical and Electronic Engineering Department
The University of Melbourne

Dedicated to my late Father, Marek Wydrowski, an engineer and mariner, who valued honest work and technical ingenuity, and who nurtured my technical career.

Also Dedicated to my Mother, Teresa Wydrowski, also an engineer, who gave me her support and faith throughout my candidature and helped throughout this time allowing me to concentrate on my work.

As well as my Supervisor Professor Moshe Zukerman, who has great perseverance, kindness and dedication to his students.

Abstract

This dissertation develops and analyses techniques for the control of congestion on IP networks. The two dimensions of congestion control are explored: 1) *Load Control*: control of amount of traffic transmitted onto the network 2) *Capacity Dimensioning*: provisioning enough capacity to meet the anticipated load to avoid congestion.

We begin the work on load control by examining the design of Active Queue Management (AQM) algorithms. We focus on AQMs that are based on an integrator rate control structure. Unlike some previous AQMs, which measure congestion by measuring backlog at the link, this structure is based on the measurement of packet arrival rate at the link. The new AQMs are able to control link utilisation, and therefore control and reduce the amount of queuing and delay in the network. The rate-based AQMs can be used to provide extremely low queuing delays for IP networks, and enable a multi-service best-effort network that can support real-time and non-real-time applications.

In the first part of Chapter 3, this dissertation develops a new rate-based AQM and a performance evaluation is performed comparing its performance to some key existing AQM proposals. In the second part of Chapter 3, the deployment of the rate-based class of AQMs is investigated experimentally and analytically, and critical efficiency issues with TCP/IP inter-operation are uncovered and addressed. In Chapter 4, the implementation of rate-based AQMs in multi-queue systems is also investigated. Rate-based control is analysed and applied to a multi-class Differentiated Services (DiffServ) scheduler, and a Combined-Input-Output-Queued Switch.

In Chapter 5, the capacity dimensioning aspect of congestion control is addressed by the development of analytical tools for the performance evaluation of a multi-service link. The analytical tools predict the delay performance of traffic given link capacity parameters, and are useful for determining the required capacity to meet Quality of Service (QoS) requirements, such as for a convergent access link with Voice-Over-IP and Data traffic. The analytical tool uses matrix geometric methods and an iterative technique to solve the underlying M/G/1 queuing problem.

In Chapter 6, we re-examine the congestion signalling architecture of the existing Internet. Currently congestion information is communicated to sources by links dropping packets or explicit congestion notification (ECN) marking. It has previously been shown that source-flow controlled networks with this signalling technique optimises the global utility of sources, and posit certain fairness properties on source rates. This thesis investigates a novel architecture for signalling congestion on best-effort networks. The new architecture is proven analytically to be scalable and robust. It is able to ensure stable congestion control across arbitrary network topologies and arbitrary amounts of packet delay. The fairness properties of the new architecture are examined analytically, and the architecture is shown to be able to provide Max-Min fairness. Simulations are performed to examine its operation and inter-operation with TCP/IP.

Declaration

This is to certify that:

- The thesis comprises only my original work towards the PhD except where indicated in the Preface.
- Due acknowledgement has been made in the text to all other material used.
- The thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Signature: Bartek Wydrowski.

Preface

This is to declare collaborative work in this thesis:

- Work carried out in Chapter 5 was carried out with the collaboration of Beatrice Meini of the University of Pisa. Her collaboration resulted in the development of the iterative procedure for computing the matrix R in Section 5.4.3.

Table of Content

List of Figures

Acronyms

ACK	Acknowledgement
AF	Assured Forwarding
AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
ARPANET	Advanced Research Projects Agency Network
ATM	Asynchronous Transfer Mode
BA	Behaviour Aggregate
BB AQM	Backlog-Based Active Queue Management
CAC	Call Admission Control
CBR	Constant Bit Rate
CIOQ	Combined Input Output Queue (switch structure)
CSFQ	Core Stateless Fair Queuing
DiffServ	Differentiated Services
DSCP	DiffServ Code Point
ECN	Explicit Congestion Notification
ECT	ECN Capable Transport
EF	Expedited Forwarding
FIFO	First In First Out
FTP	File Transfer Protocol
HTML	Hyper Text Mark-up Language
I/O	Input/Output
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IOS	Internetworking Operating System (CISCO)
IP	Internet Protocol
IQ	Input Queued (switch structure)
LRD	Long Range Dependence
MMPP	Markov Modulated Poisson Process
MRED	Multi-level RED

OQ	Output Queued (switch structure)
PHB	Per Hop Behaviour
QoS	Quality of Service
RB AQM	Rate-Based Active Queue Management
RED	Random Early Detection
REM	Random Exponential Marking
RSVP	Resource reSerVation Protocol
RTO	Retransmit Time Out
RTT	Round Trip Time
SFB	Stochastic fair BLUE
SRD	Short Range Dependence
SSQ	Single Server Queue
TCP	Transport Control Protocol
UDP	Universal Datagram Protocol
VBR	Variable Bit Rate
VC	Virtual Circuit
VoIP	Voice over IP
WFQ	Weighted Fair Queue
WWW	World Wide Web

CHAPTER 1

Chapter 1: Introduction

1.1 Looking Forward

The Internet arose out of an inter-connection of disparate computer systems, and has evolved to be one of mankind's largest engineering structures. It has grown from connecting 562 hosts in 1983, to over 162 Million in 2002 [124], and from a mere 193 World-Wide-Web (WWW) sites [124] in 1993 to over 38 Million sites in 2002. Not only has it grown in size, but also in diversity of the application-layer as well as capacity and diversity of the physical-layer. Some of the major applications that have evolved include WWW, File-Transfer-Protocol (FTP), Email, Video and Audio streaming, Voice-over-IP (VoIP), Multi-player games and e-commerce traffic. The way the information is communicated, that is the physical-layer, has also increased in diversity and capacity, from wire-line links that support kilobit capacities, to wireless systems and dense wavelength division multiplexing (DWDM) promising multi-gigabit capacities.

The underlying protocols which constitute the Internet network have had to scale to the dimensions of the present network, and the diversity of applications and physical layers. The fact that the Internet actually works, despite the rapid growth and change, is a tremendous tribute to the Internet Protocol (IP). Though it was designed to work, it was not designed to work optimally. The size of the Internet and increasing demands of applications are creating motivation to re-examine the fundamental mechanisms of the Internet. Current research examines how the performance of the Internet can be improved, as well as what the fundamental limitations are for large-scale networks. The Internet will continue to grow, both in size, capacity and the demands of applications as well as importance in our lives.

(Indeed, interest already exists for an Interplanetary Internet for scientific data [31]). Therefore a solid architecture is necessary to ensure robustness and performance of this important infrastructure.

The Internet is essentially a network of interconnected queues. The most fundamental experiences of a packet whilst traversing this network is delay and the Internet is facing increasing packet delays. Just like the road system, the Internet is prone to traffic congestion, which causes large queues of packets to build up inside the network, awaiting transmission to the next link along the path. This makes it difficult for demanding applications such as telephony, video-conferencing and interactive games to be deployed, since they require packets to be delivered quickly, without perceptible delay.

The area of congestion control studies how performance can be improved when the demand for the finite transmission capacity exceeds the supply. Research has taken two approaches to congestion control. The first approach of capacity provisioning is about how to make the roads of the Internet wide enough to cope with traffic demand. The second approach, of load control, investigates how the users of the Internet can be controlled so that the demand they place on the network is within the capacity of the network. Both of these approaches are developed in this dissertation.

The fundamental re-examination of Internet congestion control by researchers has involved the application of control theory, queuing and probability theory, as well as simulation, to determine the characteristics of this large system. Not only does this work help us understand the way the Internet works, such research continues to uncover ‘Squeaky wheels’ in the mechanisms of the Internet, and even predict critical weaknesses in design.

This thesis makes a number of contributions to congestion control. The work of Chapters 3 and 4 focuses on one of the squeakiest wheels of Internet congestion control, the active queue management algorithm, and develops recent advances in the design of this component into a form which can be deployed in practical routers and switches. A tool for predicting the performance of traffic, which helps in designing

links, is developed in Chapter 5. Finally in Chapter 6 a fundamental re-examination of the way the Internet communicates information about congestion is undertaken.

1.2 Focus of this Thesis

This Thesis studies two aspects of congestion control, load control and capacity provisioning and develops new techniques in both of these areas. There are two parts to the work on load control. Firstly, in Chapters 3 and 4, architectures for the application of modern active queue management (AQM) algorithms in network equipment are developed. Secondly, in Chapter 6, Internet congestion signalling and the resultant fairness properties are re-examined. The work on capacity provisioning in Chapter 5 results in an analytical tool for traffic engineering of a multi-service link.

The focus of the re-examination of Internet congestion control signalling, in Chapter 6, stems from the fundamental question “what if the Internet had been built differently? What other sorts of structures also allow for distributed load control with source flow control?” The proposal examined is one possibility and results in different fairness properties than the Internet. The proposal is derived analytically and tested by simulation. Analysis shows that the proposed network structure is stable and the stability is preserved over arbitrary topologies and network delays. One of the advantages of the proposal is that, unlike the existing network, the stability can be preserved for arbitrary topologies without having to gauge the number of bottleneck links on the end-to-end path. Although the origins of the proposal are as an alternative to the current IP methods of signalling congestion, it turns out it may also be used to enhance the performance of the existing network, especially in areas such as multi-hop ad-hoc networks. Simulations are performed to examine the protocol’s interoperation with IP.

In the area of AQM design, in Chapters 3 and 4, this thesis bridges some theoretical work in the field of congestion control with experimental work and physical implementation. To bring theoretical work closer to real systems, many results have been obtained by simulation, which allow for complexities otherwise not

readily modelled by analysis. A network test bed for results in Chapter 3 was also developed on LINUX-based routers. Some previously unreported results in this thesis demonstrate that such experimental work is an essential part of research.

The experimental approach of this thesis has uncovered a result for AQM design that theoretical work had overlooked, that ECN capable flows must be split into a separate queue from non-ECN capable flows for optimal delay and loss performance. The result of this work is an architecture for the implementation of rate-based (RB) AQM algorithms for networks with ECN and non-ECN flows.

Contributions of this Thesis to load control include the application of RB AQMs to multi-queue systems. Previous work has looked at the AQM algorithm in isolation, as controlling a single queue. Here, RB AQMs are applied to DiffServ scheduling algorithms with multiple queues as well as Combined-Input-Output-Queued switches. This brings the application of modern AQMs closer to implementation.

There is a large body of theoretical work on AQM in literature that explores the control-theoretic aspects of flow control. Various works investigate the stability properties, transient performance, and optimal structure of AQM design. Rather than adding to the populated area of control analysis of congestion control, this thesis examines the architectural aspects of AQM design. The rate-based (RB) AQM integral (I) controller is identified as a key structure for future AQM design; other possible structures such as a proportional integral controller (PI) are considered as descendents from this basic form. By basing the work in this thesis on the basic rate-based Integral AQM structure, the work in this thesis can leverage off new results in AQM designs. This thesis takes the approach of developing the architectures for the application of the RB AQMs, rather than focusing in detail on the RB AQM algorithm itself, which will continue to evolve.

The work on capacity provisioning in Chapter 5 looks at the other dimension of congestion control, that of providing adequate capacity so that congestion does not occur, or at least, has predictable and limited impact on performance. To facilitate this, an analytical tool is developed for determining the delay performance of packets in a multi-service environment. Such a tool models packet delay performance given a

certain capacity, and is therefore useful for finding if this capacity is adequate for a given QoS requirement. Matrix-geometric techniques, stemming from work of Marcel Neuts, are applied to solve this queuing problem. This work serves as an adjunct to work on multi-service links in Chapter 3, where load control techniques were developed for a DiffServ link.

1.3 Subdivision of this Thesis by Chapter

Chapter 2: Background on Internet Congestion Control. In this chapter, the two aspects of congestion control, load control and capacity provisioning are introduced. The Internet source flow control model of load control is explored. The TCP algorithm is described as a specific example of source flow control, and the general system of source flow control as a utility optimisation problem is introduced. Active queue management (AQM) is introduced as a computational element in the utility optimisation problem, and the broad aims of AQM design are listed and discussed. Current AQM designs are described, along with their shortcomings. Different fairness criteria for source rate allocation, asides from utility optimisation, are also introduced. The issue of TCP unfriendly flows is introduced, and some policing schemes for mitigating this problem are described. The future implications of Capacity provisioning as a method of controlling congestion are explored.

Chapter 3: Rate-Based Active Queue Management. This chapter investigates the design of AQM algorithms from a structural perspective. A background on the control theoretic issues of AQM design is given, and the design of different classes of AQM algorithms is discussed. The class of rate-based (RB) AQM algorithms is explored. A new RB AQM algorithm is introduced and its performance is compared to several key AQMs by simulation. Some control theoretic properties of this algorithm are discussed. The deployment of RB AQMs in a network with ECN and non-ECN capable source algorithms is analysed and an important consideration in the optimal deployment of the RB AQM class of algorithms is uncovered analytically and experimentally. A new structure for the deployment of RB AQMs in networks that allows improved loss and delay characteristics is presented.

Chapter 4: Rate-Based AQM in Multi-Queue Systems. In this chapter, structures for deployment of RB AQM algorithms in multi-queue systems, such as DiffServ and a Combined-Input-and-Output-Queued (CIOQ) Switch are developed. A technique for deployment of RB AQM with DiffServ replaces the existing Weighted-Random Early Detection (WRED) AQM, and enables low latency links. Simulation studies of WRED and RB-AQM based DiffServ are performed. Also a structure, which only requires one AQM per output port, for implementing RB-AQM in a CIOQ switch is developed. This replaces the existing tail-drop queue management in latest CIOQ switches and provides the same benefit of low latency. Analysis of transients and simulation of performance is performed on a model of such as switch.

Chapter 5: Multi-Service Network Provisioning. In this chapter an analytical performance evaluation of a two class DiffServ link is made. Whereas in Chapter 4, load control techniques for DiffServ were developed, this Chapter develops the capacity provisioning aspect for such links. A model of a voice and data convergent access link is developed to determine the performance of low priority data in the presence of high priority voice connections. This model gives the queuing performance of data traffic and is useful for capacity provisioning. Matrix geometric techniques are employed to solve this queuing problem and an iterative procedure to evaluate the results is developed.

Chapter 6: A New Congestion Control Architecture. In this chapter, the Internet architecture for signalling network congestion is re-examined. The Internet uses packet dropping or ECN packet marking to communicate the intensity of congestion, and this locks the network into certain fairness properties that are discussed in Chapter 2, such that the network always optimises the global utility. In this chapter, a new method of signalling congestion for a best-effort network is proposed, which results in Max-Min Fairness. Steady-state analysis is performed to determine its fairness properties, and stability and robustness are studied analytically. Transient

conditions are also studied by simulation. Inter-operation of the new protocol with TCP is explored.

1.4 Contributions of this Thesis

In this section the key contributions of this thesis are summarized. They are not listed in order of importance, but in order of appearance in this thesis.

The work in Chapter 2,3 led to publications of [1] [2] and [3]. The contributions in Chapters 2,3 can be summarized by the following points:

1. Design of new active queue management (AQM) algorithm GREEN. Unlike many previous AQMs which measure packet backlog at a link to determine congestion (backlog-based), GREEN measures the packet arrival rate at the link (rate-based).
2. Simulation study of GREEN and comparison of several leading backlog-based and rate-based AQM proposals.
3. Experimental implementation of rate-based AQM in LINUX-based router.
4. Discovery, through experimental implementation and analysis, of the low latency efficiency collapse phenomenon. This occurs from the deployment of rate-based AQMs in the present Internet that has both ECN and non-ECN capable sources.
5. Design of ‘congestion-notification-splitter’ congestion control component to allow deployment of rate-based AQMs (such as GREEN or REM) in current Internet with heterogenous sources that are ECN and non-ECN capable. This component prevents low latency efficiency collapse, and makes it possible to have the maximum amount of packets receive minimum loss and minimum delay.

The work in Chapter 4 led to publications of [4] and [5]. The contributions in Chapter 4 can be summarized by the following points.

6. Development of architecture for rate-based AQM controlled DiffServ link. Algorithm design which couples class-based scheduling to rate-based AQM control, making low-latency DiffServ links possible.
7. Simulation study of rate-based AQM DiffServ, and comparison with existing backlog-based AQM DiffServ link.
8. Development of architecture for rate-based AQM controlled combined input-output queued Router/Switch.
9. Simulation study of rate-based AQM CIOQ Router/Switch and comparison with existing backlog-based CIOQ Router/Switch architecture.

The work in Chapter 5 led to publication of [6]. The contributions in Chapter 5 can be summarized by the following points:

10. Analytical performance evaluation of a two-class DiffServ link. This work is particularly for capacity provisioning of convergent access links which carry voice and data traffic.
11. Application of matrix-geometric results to evaluation of multi-service network problem. Application of recent iterative techniques to solve M/G/1 queuing problem.

The work in Chapter 6 led to publications of [7] and [8]. The contributions in Chapter 6 can be summarized by the following points:

12. Re-examination of Internet congestion control signalling, and proposal of alternative inter-networking signalling architecture called MaxNet. MaxNet is a distributed and state-less source flow control based scheme for rate allocation which can achieve Max-Min fairness.
13. Analytical steady state evaluation of MaxNet. Proof of Max-Min property of MaxNet for homogenous sources. Unlike the Internet, which has been shown to maximise global utility, MaxNet is shown here to result in Max-Min.
14. Analytical study of MaxNet stability and robustness properties. Proof that MaxNet is stable and robust for arbitrary network topologies and network delays.

15. Proof that unlike the Internet, MaxNet does not need to gauge the number of bottleneck links on the transmission path of a source to achieve stability for arbitrary networks topologies.
16. Verification of MaxNet through Fluid-flow simulation. Simulation study of MaxNet in transient conditions.
17. Simulation study of Inter-operation of MaxNet with TCP/IP when MaxNet is used as an IP encapsulation protocol.
18. Analysis of congestion signal communication efficiency of MaxNet and Internet.

A patent, registered both in the USA and Australia was generated from the work in this thesis [9]. Throughout his PhD candidature, the author also contributed to the publications [10] and [11] which were not reported on in this thesis.

1.5 Publications by the Author

[1] B. Wydrowski and M. Zukerman, "GREEN: An Active Queue Management Algorithm for a Self Managed Internet" Proceedings of ICC 2002, New York, 2002. pp. 2631-2635.

[2] B. Wydrowski and M. Zukerman, "On the Transition to a Low Latency TCP/IP Internet" Proceedings of ICC 2002, New York, 2002. pp. 2631-2635.

[3] B. Wydrowski and M. Zukerman, "QoS in Best-Effort Networks" IEEE Communications Magazine, vol. 40, pp. 44-49, 2002.

[4] B. Wydrowski and M. Zukerman, "High Performance DiffServ Mechanism for Routers and Switches Packet Arrival Rate based Queue Management for Class Based Scheduling" Proceedings of NETWORKING 2002, Pisa, Italy, 2002. pp. 62-73.

[5] B. Wydrowski and M. Zukerman, "Implementation of Active Queue Management in a Combined Input and Output Queued Switch" Proceedings of Accepted for publication in ICC, Alaska, 2003.

[6] B. Wydrowski, M. Zukerman, C. H. Foh, and B. Meini, "Analytical Performance Evaluation of a Two Class DiffServ Link" Proceedings of ICCS, Singapore, 2002.

[7] B. Wydrowski and M. Zukerman, "MaxNet: A congestion control architecture for Max-Min fairness" IEEE Communications Letters, vol. 6, pp. 512-514, 2002.

[8] B. Wydrowski and M. Zukerman, "MAXNET: A New Network Congestion Control Architecture for MaxMin Fairness" Proceedings of Accepted for publication in ICC 2003, Anchorage, Alaska, 2003.

[9] B. Wydrowski and M. Zukerman, "An active queue management process". USA, Australia, 2002.

[10] S. Chan, M. Zukerman, E. Wong, K. T. Ko, E. Yeung, and B. Wydrowski, "A Congestion Control Framework for Available Bit Rate Service in ATM Networks" International Journal of Communications Systems, vol. 15, pp. 341-357, 2002.

[11] C. H. Foh, B. Meini, B. Wydrowski, and M. Zukerman, "Modelling and Performance Evaluation of GPRS" Proceedings of IEEE VTC 2001, Rhodes, Greece, 2001. pp. 2108-2112.

CHAPTER 2

Chapter 2: Background on Internet Congestion Control

2.1 Introduction

The area of Internet congestion control was baptised in 1986-1987 when the then ARPANET suffered ‘congestion collapse’ [57]. Congestion collapse had been predicted by Nagel [89] in 1984. Congestion collapse occurs when mounting levels of traffic result in high packet loss inside the network, such that few or no packets are actually delivered to their destination, yet each link is highly loaded.

The initial response to ARPANET’s congestion collapse problem was to increase the capacity of the network. This helped temporarily, but the ARAPNET continued to suffer congestion collapses until a strategy to control the load of packets entering the network was developed. In 1988 Van Jackson enhanced the famous *Transport control protocol* (TCP) [57] so that the transmission rate was responsive to the level of network congestion. TCP was made to reduce the rate of transmission of hosts when it sensed the network load was nearing congestion collapse. Since the introduction of this enhanced TCP, congestion collapse did not reoccur.

This history of the Internet reflects the two fundamental approaches to the problem of controlling congestion in networks 1) Capacity Provisioning and 2) Load control. Since Congestion collapse occurs when the load of packets placed onto the network exceeds the network’s capacity to carry the packets, the capacity provisioning approach is to ensure that there is enough capacity to meet the load. The load control approach is to ensure that the load of packets placed onto the network is within the capacity of the network. Capacity provisioning is achieved either by accurate performance analysis and traffic modelling, or the brute force approach of

over provisioning. There is a range of load control strategies for networks, from connection admission control schemes through to best-effort flow control as on the Internet.

As is evident from the ARPANET experience, economic and technical reasons limit the capacity provisioning approach as a sufficient solution to the congestion control problem. Provisioning for peak loads is an expensive proposition, and in any case, it may not always be possible to anticipate what the peak load will be. Therefore, load control has a permanent role in the congestion control in networks. In this thesis, we will develop the areas of Internet congestion control by developing load control as well as provisioning techniques.

2.1.1 Load control mechanisms

When the capacity available is less than the demand for capacity, load control is the critical element which determines how many packets are allowed onto each link of the network, who gets to send them and when. This controls the *quality of service* (QoS) metrics such as bandwidth, latency and jitter experienced by users. There are a number of load control mechanisms, and these can be classified by the type of QoS guarantees they can deliver.

At one end of the spectrum are the connection admission control (CAC) schemes, such as the Resource Reservation Protocol (RSVP) [4]. Such schemes require the network to maintain information about each connection and arbitrate whether connections are admitted or rejected so that the connections that are admitted can be absolutely guaranteed their required bandwidth for the duration of the connection. When the load of requested connections is increased beyond the capacity of the network then some new users will be rejected in order to maintain the bandwidth guarantees made to already admitted users. CAC is good for honouring bandwidth supply contracts that specify minimum rates. The IntServ proposal [122] uses RSVP signalling protocol to extend Internet functionality to include CAC. However, CAC schemes on the Internet are not widely deployed and it is believed they cannot scale to widespread use [93] due to the amount of per-connection information required inside Internet routers and switches.

At the opposite end of the load control spectrum is the best-effort network. Unlike a CAC based network, the best-effort network does not need information about each connection to be stored in the routers and switches on the data path, because there is no resource reservation for a connection. The best-effort network sees each packet transmitted as independent of any other packet. The sources themselves decide how much they should send and on the Internet the sources are predominantly TCP/IP. All requested connections are admitted, and the available capacity is shared between the connections. As a result, no explicit guarantees can be given about the bandwidth available to each connection. However, the simplicity of the network infrastructure is compelling, since there is no concept of connection within the network, only the ability to forward packets is required of the network. This simplicity is a key reason for the success of IP networks.

Somewhere between CAC and best-effort networks, are flow aggregate schemes such as DiffServ [26] [10] [29]. With DiffServ, although individual connections are given no guarantees of minimum bandwidth, classes of connections are given minimum bandwidth guarantees. The bandwidth allocated to each connection within a class is essentially made by a best-effort mechanism, and each connection with a class competes for bandwidth. However, the aggregate of connections within the class is guaranteed a minimum bandwidth in the presence of other classes in the network. Such aggregate schemes require only a small amount of state information in the network, such as the relative bandwidth assignment between the classes, and this state information is proportional to the number of classes. Although not offering the per-connection guarantees of CAC systems, they allow the network operator to give important applications some protection from less important flows. We will discuss DiffServ in detail in Chapter 4.

2.2 TCP/IP

By far, the most dominant load control paradigm on the Internet is source flow control. The dominant source flow control protocol on today's Internet is TCP. In [116] measurements of traffic at core routers indicate that about 95% percent of traffic volume on the Internet is generated by the Transport Control Protocol *TCP*

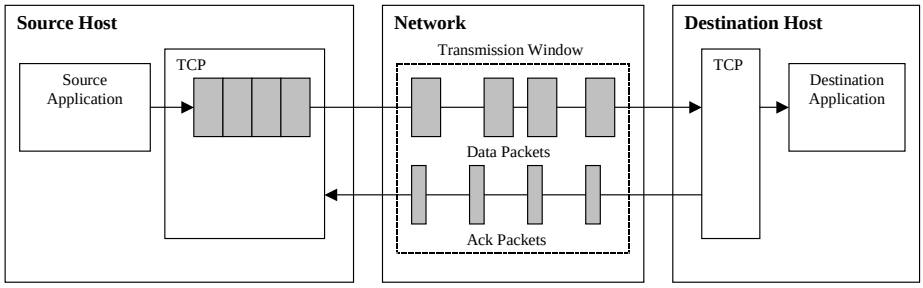
source algorithm. The remaining traffic is UDP (about 4%) and ICMP. There are various versions of TCP, including Reno [112, 113], SACK [82] and Vegas [18]. The most widely deployed version is TCP Reno. This modern version of TCP stems from the first congestion controlled TCP, TCP-Tahoe, as described by Jackobson and Karels [57].

The congestion control behaviour of TCP protocol is important to the techniques developed in this thesis and we will detail the congestion control mechanism of TCP in this section. The key function of TCP is to provide a reliable connection for the application layer across an unreliable best-effort network. TCP provides a number of guarantees to the application layer: (1) delivery of data (2) in order delivery of data and (3) error free delivery of data. These guarantees are achieved by error detection, buffering and retransmission. However, in the case of congestion collapse, none of these guarantees can be met. The congestion control mechanism in TCP is in place to protect the integrity of the network and prevents congestion collapse.

To ensure the integrity of the network, TCP is designed with the "conservation of packet principle". This principle demands that, in equilibrium, a packet must be removed from the network, for a new packet to be placed onto the network. The intuitive argument to explain this principle is that a successful delivery of a packet frees the network resources for the delivery of another packet. To support this conservation principle, TCP has a window based flow control.

Window based flow control limits how many packets TCP is able to transmit onto the network which have not been acknowledged as having been received. As shown in Fig. 2.1, only the amount of packets which fit into the window are allowed onto the network, and packets generated by the application which do not fit, wait at the source. TCP sends acknowledgement packets from the destination for packets received successfully. Once the acknowledgement for a packet is received at the source, the acknowledged packet and the acknowledgement packet are removed from the transmission window, leaving room for the transmission of the new packets awaiting transmission.

Fig. 2.1 TCP Window Based Flow Control



The window size *w* limits how many packets the TCP connection can have in flight in the network; including acknowledgement and data packets. This effectively controls the TCP throughput. The packets, by convention known as segments of data, have a maximum segment size *MSS*. If we consider that there are *w* packets in transit at time *t*, then after one *round-trip-time* (RTT) at time (*t + RTT)*, these packets would be delivered to the destination and acknowledged at the source and *w* new packets would be in the network. Therefore the transmission rate of the TCP source, in bits per second, can be described by:

$$T = \frac{w \cdot MSS}{RTT}(b/s) \tag{2.1}$$

The window size is controlled by TCP so that the transmission rate is matched to the most severe bottleneck, either the network or the receiving host's ability to consume the information. The actual window size is the minimum of the network's capacity to transmit, *cwnd*, and the receiver's capacity to receive information, *awnd*:

$$w = min\ (cwnd, awnd) \tag{2.2}$$

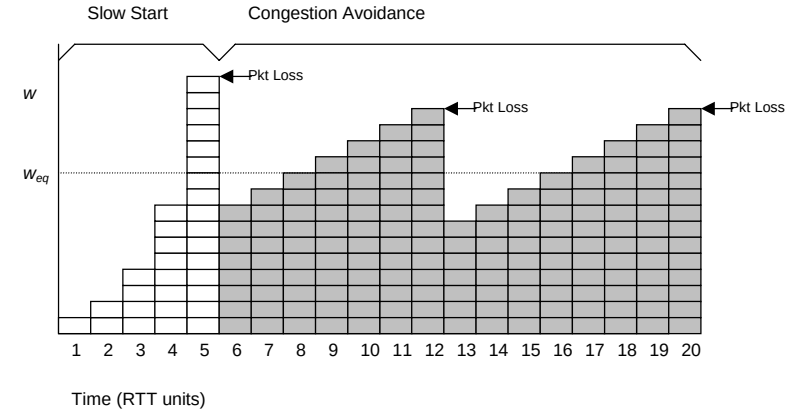
In practice, *awnd* is seldom the limiting factor, and the network is typically the bottleneck. Therefore we can assume *w=cwnd*. However, the question remains how TCP knows what the network capacity is. For a source to fully utilise the available capacity, *w* should be set to the bandwidth delay product of the bottleneck capacity of the end-to-end connection. In this way, TCP fills the 'pipe' between the

source and destination with packets. However, the best-effort Internet does not provide any explicit information about the available capacity between hosts.

TCP is able to gauge the available capacity by loading the network with traffic. TCP increases its transmission rate above the capacity, which causes backlog and packet loss to occur. The TCP source detects loss by sensing that the lost packets have not been acknowledged. When packet loss is detected, TCP knows it has exceeded the capacity, and reduces its rate. This increase/decrease cycle continues throughout the whole connection time to ensure TCP is fully utilising available capacity, and not exceeding network capacity.

TCP has two key modes of operation with different mechanisms for increasing and decreasing the window; 1) slow-start and 2) congestion avoidance. TCP begins the connection with window size of 1, and uses the slow start algorithm to increase the windows size until the first packet loss is detected. After the first loss, TCP uses the congestion avoidance mode to increase and decrease the window size about the operating point.

Fig. 2.2 TCP Congestion Control By Control of Window



In the counter-intuitively named slow-start mode, TCP increases the window size by 1 for each acknowledged received, which results in a very fast exponential growth of the window size. The slow-start mode is intended to bring the window size, and therefore the transmission rate of TCP, quickly to roughly the correct value. Once the window size grows over the bandwidth delay product of the transmission

path, packet loss will inevitably occur in the network, and then TCP switches to the congestion avoidance mode when the first loss is detected.

In congestion avoidance, TCP increases the window size by a factor of 1/*cwnd* for every packet acknowledged. However, for every packet loss, the window is decreased by a half. This additive-increase/multiplicative-decrease (AIMD) algorithm was first introduced in [58] and added to TCP by Van Jacobson in [57]. An important property of the AIMD mechanism is that it results in equal capacity sharing at links between TCP sessions [20] which traverse the same end-to-end path.

A further mechanism which occurs when there is catastrophic transmission failure is called Retransmission Timeout (RTO) [17]. Whenever TCP sends out a packet, it starts a timer. If the acknowledgement of the packet sent is not received with a certain time, a timeout occurs. TCP adjusts the time allowed for an acknowledgement in view of the RTT of the network. When a RTO occurs, it is a sign of very heavy congestion or link failure. RTO timeout should be rare, and when it happens, TCP reduces the congestion window to 1 and restarts in slow start.

2.3.1 Rethinking Best-Effort Networks

In the previous section, we started the description of best-effort networks with the specific example of the Internet's TCP protocol. In this section we will show that TCP is in fact a special case of a more general framework for best-effort networks as described by Kelly in [64].

To introduce Kelly's framework for describing best-effort networks, we will provide an example of how bandwidth allocation in a best effort network compares to bandwidth allocation in a CAC network. Let us consider an example where there are three users each requesting a 1 Mb/s connection across the same 2 Mb/s link. In a CAC network, one user will have to miss out. However, the best-effort network makes this situation less rigid by making the users' demand for bandwidth elastic. When users do not need strict guarantees of minimum bandwidth then blocking one user is not necessarily the best solution possible. Let us assume that a user is able to quantify, by a single number, its perceived quality of service (QoS) value of sending at a certain rate. Say, transmitting at a rate of 1 Mb/s gives the maximum possible

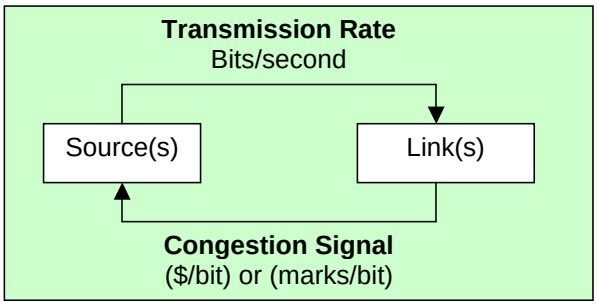
user perceived QoS value, but transmitting at a rate of less than 1 Mb/s still gives some, but less, QoS value to the user. Then, it is possible to conceive of a solution, where by making the three users transmit at 2/3 Mb/s each, the sum of the perceived QoS values of all three users is greater than the sum if only two users are allowed to transmit at the maximum 1 Mb/s and one user is blocked. In such a system, where the user demand has some flexibility, it is possible to achieve a compromise solution for sharing the available capacity which is better for the QoS of the whole community of users, despite giving less capacity to some users. This is exactly the solution that the best-effort network achieves.

Research [64] [76] has shown that the same processes found in supply and demand economic systems, which allow buyers and sellers to negotiate price, allow best-effort networks to ensures that the available capacity of network is utilised, and the QoS perceived by the users is maximised. To achieve these objectives, the machinery behind a best-effort network needs just two fundamental elements, a *source flow control* algorithm and a link *active queue management* (AQM) algorithm.

We have already introduced TCP as an example of a source flow control algorithm. In general, a source flow control algorithm resides in the host transmitting information, and decides how much information to transmit based on the congestion level of the network. The link AQM algorithm lives inside a router/switch, and monitors the queues that buffer packets awaiting transmission on the outgoing links connecting the router/switch to the next router/switch in the network. The AQM algorithm estimates the level of congestion the link is experiencing and generates a congestion signal, which communicates the congestion level to the source. Each AQM, on the source-to-destination path of the connection (e.g. TCP connection), contributes to the total congestion signal received by the source. The source algorithm can be thought of as the buyer of capacity, and the link AQM algorithm is the seller of capacity. Today, the source flow control algorithm that accounts for the majority of the traffic volume on the Internet is TCP, and the dominant AQM algorithm is the tail-drop queue, which we will describe later. The two elements are

connected in a closed loop system, as shown in Fig. 2.3, which allows the buyer and seller to negotiate.

Fig. 2.3 Logical Congestion Control Feedback Loop

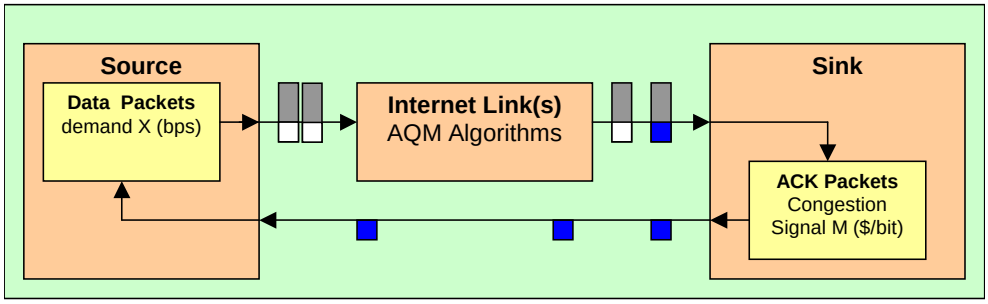


Consider a single source (say, TCP) transmitting information at a certain rate (b/s). The number of bits transmitted across the network is effectively the amount of resource purchased by the source from the links. The links that the TCP connection traverses generate a combined feedback signal which is a measure of the congestion level of the path. This signal controls the source, and can be thought of as a price per bit of transmission. If the transmission rate of the sources is above the physical capacity of the link (say, 56.6 kb/s for a modem) then the link increases the feedback signal to tell the source to slow down. In other words, if demand exceeds supply, then the price is increased. Conversely, if the sources transmit at less than the available capacity, then the feedback signal is decreased and the source increases its rate so that the available capacity is utilised. This is analogous to demand being below the supply resulting in a price decrease. This feedback mechanism allows the source to learn to send at the capacity of the network available between the source and destination, as each link on the end-to-end path in the network contributes to this signal.

As described in Section 2.2, the way this feedback signal, or price, is communicated is typically by dropping packets. Depending on the level of congestion at the link, the AQM drops packets at a rate that reflects the severity of the congestion. Packet dropping is a natural mechanism for signalling congestion, because congestion occurs when more packets arrive than can be transmitted on the

link, which leads to buffer overflow and inevitably packet loss. However, other modes of communicating the feedback signal have been developed, and one is explicit congestion notification (ECN) [105]. With ECN, to avoid packet loss, packets are marked by setting a single bit in the packet header, instead of being dropped, to communicate the presence of congestion. Fig. 2.4 shows the physical path that the ECN marking signal takes from the link AQM to the source.

Fig. 2.4 Physical Congestion Control Feedback Loop



2.3.2 Differentiated Bandwidth

To see how different sources attain different rates, let us explore the analogy to economics further. When the buyer and seller negotiate for a price of some quantity of a resource, they have different objectives. The buyer wants to maximise the *Net Benefit*, *B*, of the purchase, which is equal to the benefit, $U(x)$, of having x units of resource less the cost of purchase: $B = U(x) - x \times c$, where c is the price of one unit of resource. We assume that the seller's objective is to sell all of the resources. In the negotiations, the seller is able to control the price, and the buyers are able to control the amount of resource purchased. Each buyer will only purchase an amount of resource that gives it the maximum *Net Benefit*. If the price is too high, the buyers will reduce the amount purchased, and not all of the resource will be sold.

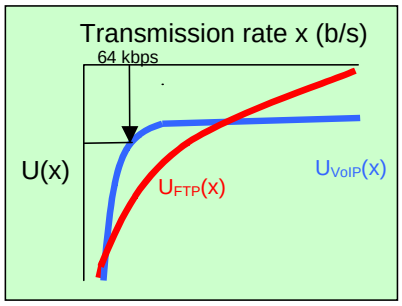
This will cause the seller to reduce the price, to motivate the buyers to buy all of the available resource. However, if the price is too low, than the demand will exceed the supply, and the seller will not be able to deliver the resources. The seller will tune the price up and down in this fashion until the right price is found which induces the group of buyers to buy all of the resource. Although the right price sells

all of the resource, not every buyer will necessarily have an equal share. The portion of the total resource that each buyer purchases is based on their own *Net Benefit* calculation, which is different if each buyer has a different *U(x)*.

2.3.3 Utility, a measure of user's perceived QoS

In the best-effort QoS model, the buyer's *U(x)* function, defines how much QoS value a source attributes to sending at a particular transmission rate *x*. It is known as a *utility function* as it describes of what utility or usefulness a particular transmission rate is to the user. Fig. 2.5 shows two examples of different utility functions for quite different applications. The utility function for a Voice over IP (VoIP) application is described by curve $U_{VoIP}(x)$. For uncompressed voice, acceptable voice quality requires a transmission rate of about 64 kb/s to transmit an 8 kHz 8 bit signal. It is possible to cope with a little less quality than this, but as the graph of $U_{VoIP}(x)$ shows the utility of transmitting voice at less than 64 kb/s reduces sharply as the transmission rate is decreased. Also, the quality of voice does not increase much if the transmission rate is increased beyond 64 kb/s, so the utility function remains almost flat above 64 kb/s, indicating there is little utility to be gained by transmitting faster than 64 kb/s. With the second example $U_{FTP}(x)$, which represents a file transfer FTP application, the utility function reflects a different requirement. Naturally, the user is happier the faster the file is transferred, so the utility functions shows how the utility increases as the transmission rate is increased. However, note that the user will be significantly more pleased if the file is transferred within a few seconds, rather than a few minutes or hours, but they may not even notice if the file is transferred in a few milliseconds rather than seconds. Therefore, the utility increases steeply until an acceptable transmission rate is achieved, and continues to increase above this transmission rate, but by a diminishing amount. In [77], it has been shown that source flow control algorithms like TCP embody a particular utility function. Indeed, TCP's (Reno version) utility function is much like $U_{FTP}(x)$ in Fig. 2.5, and for the reasons we have described, it is useful for file transfer.

Fig. 2.5 Utility Functions



As described previously, when there are many sources competing for the same capacity, this is analogous to having many buyers competing for the same resource. All of the sources transmitting on a given path contribute to the total demand, and the same congestion signal, is communicated to all of the sources competing for the same capacity. Each source will transmit at a rate, which maximises its *Net Benefit* given the current price of bandwidth. Users with different utility function will be willing to transmit (purchase) different amounts, since their *Net Benefit* calculation is different. A user who has a large gain in utility for purchasing some bandwidth will be willing to pay more for this bandwidth then a user whose utility is not improved much by the purchase. So in the above example, the VoIP session will be willing to pay more for the first 64 kb/s than the FTP session, since the increase in utility is greater. In a situation where the capacity is limited to say 100 kb/s, and the FTP and VoIP applications compete for capacity, the VoIP session will out-bid for the first 64 kb/s, and the FTP session will be left with the rest. In practice this means that given a certain packet loss rate, a VoIP session should be more aggressive in sending at its required 64 kb/s than an FTP session. This is indeed true for VoIP protocols based on UDP, which have very limited rate back-off in response to packet loss or ECN marking. Such protocols are called TCP unfriendly, since they will squeeze TCP rate below an equal share on a bottlenecked link. However, one can view this in a different light.

In a landmark result for best-effort networks [76], it turns out that when each user transmits at a rate that maximises only their own *Net Benefit*, the supply-demand pricing process will drive the price to a point which regulates all of the users rates

such that the total utility of all of the users is the maximum possible. This means so long as the users define their utility function true to their applications' needs, the supply-demand pricing process will divide the capacity so that the maximum possible QoS is experienced by the whole community of users. A best-effort network makes this optimum allocation in a completely distributed way. Each source algorithm only needs to select a transmission rate that improves its own Net Benefit, and each link only has to adjust the price so that the capacity demand matches the capacity supply.

2.3.4 QoS through Supply Demand Pricing

The analogy to economics we have been using to describe congestion control can also be taken quite literally as a pricing mechanism for charging users [66]. In this proposal, the congestion feedback signal generated by the network, which we have already described as a *price per bit ($/bit)***,** can be used to charge the user for the volume of traffic they transmit. To achieve this, the access provider needs only to be able to count the number of ECN marked packets that the user receives, as this is in proportion to the price and volume of traffic.

This is a good pricing policy for a network access provider as it regulates the usage of available capacity and allows flexible contracts. The more bits the user transmits, the higher the charge. More importantly, if there is heavy congestion, users are charged more, creating incentive to reduce demand in a busy period. However, from the perspective of the user, if they have a particularly important application that is bandwidth hungry, they are able to receive the capacity they need by out-bidding other users. Therefore, tying user charging to the congestion control mechanism extends the self-regulating dynamics found in the source and link algorithm, to the behaviour of the user.

2.3.5 The Formal Utility Optimisation Problem

In previous sections, we introduced an intuitive explanation of best-effort network economics analogy. In this section, we will describe the formal optimisation problem that describes the system as set out in [62] and reformulated in [76]. These

papers are pivotal to our understanding of best-effort networks, and further work in Chapter 6, where we redefine the network to solve a different optimisation problem. The complete *System* is comprised of the *Network* and the *User*, and we will now define the parameters of each:

The Network

The network is defined as a set of J resources with finite capacities C_j. A route r is a non-empty subset of J. R is the set of all possible routes. Let $A_{jr} = 1$ if j is on a particular route r, otherwise $A_{jr} = 0$. The A_{jr} matrix defines which resources a route r passes through. Let S be the set of all possible source-sink, and s be a particular source-sink. The matrix H defines which route serves a particular source. The element H_{sr} is set to 1 if the route r serves the source s, otherwise $H_{sr} = 0$.

The User

Each source s relates to a particular user who transmits at a rate x_s to a particular sink. The total flow x_s between a source-sink pair is supported by a number of flows over different routes. The flow pattern y describes how much of flow, y_r, there is over a particular route r. The total flow, described by the vector x of all sources, is then equal to

$$x = Hy, \tag{2.3}$$

the sum of all the flow on all the routes between each source-sink. For the flows to be feasible, the sum of all the flows through each resource must be at or below the capacity. This condition is $A \cdot y \le C$, as $A \cdot y$ sums all the flows for all the routes using a particular resource. This is a feasibility condition for the system (2.3).

Each user s that transmits at a rate x_s, and has a utility function $U_s(x_s)$. The utility function must be increasing, strictly concave and continuously differentiable function of x_s over the range $x_s>0$. The aggregate utility of the network is the sum of the individual utilities of each of the sources.

The System

The formal objective of the system process is to maximise the total utility of all the users (2.4) subject to the feasibility constraints (2.5) and (2.6). By maximising the *aggregate utility* function (2.4), the system apportions the maximum capacity to each user whilst minimising the compromises to other users.

SYSTEM(U,H,A,C):

$$\text{maximise} \quad \sum U_s(x_s) \tag{2.4}$$

$$\text{subject to} \quad H \cdot y = x,\ A \cdot y \le C \tag{2.5}$$

$$\text{over:} \quad x, y \ge 0 \tag{2.6}$$

The network determined *cost* of sending a unit of information (say, a bit) for a particular user s is λ_s. The *benefit* to the user of sending at rate x_s, is the utility (value of sending at x_s) minus the cost at rate x_s (2.7). Each user adjusts their transmission rate x_s to maximise their benefit.

USERs(U_s,λ_s):

$$\text{maximise} \quad U_s(x_s) - \lambda_s \cdot x_s \tag{2.7}$$

$$\text{over} \quad x_s \ge 0 \tag{2.8}$$

The network attempts to maximise its *profit*. The *profit* is the product of the cost of bandwidth and the bandwidth served (2.9). The network adjusts the price so that profit is maximised. If the price is too high, users will not send as much and profit will suffer. Conversely, if the price is too low, the maximum profit possible is not generated and users may send more than there is capacity to serve.

NETWORK(H,A,C,λ)

$$\text{maximise} \quad \sum_s \lambda_s \cdot x_s \tag{2.9}$$

$$\text{over} \quad x_s \ge 0 \tag{2.10}$$

In [64] Kelly shows that given the User, Network and System processes as described above, there exists a unique price vector λ that solves the User, Network and System maximisation problem. The working in [64] takes a penalty function approach to solving the optimisation problem. In [62] Kelly proves that the maximised utility transmission rate x_s is a stable point of the system, to which all trajectories converge. In [62] Kelly extends the system to incorporate time lags and noise in the feedback mechanism.

In [76] Low *et al.* show the remarkable property that the Internet solves the above optimisation problem in a completely distributed and asynchronous manner. Low shows that the problem is solved separately by the interaction of the source and link algorithms, without the need to centrally coordinate between these elements. Total user utility is maximised, if each user maximises their own benefit, as in (2.7), and each link increases or decreases the link price vector p_l, so that in steady state, the aggregate arrival rate $x^l(t)$ is matched to the target link capacity c_l, such as by this control law:

$$p_l(t+1) = [\, p_l(t) - \gamma (x^l(t) - c_l)]^+$$

We will later show that this is the underlying structure of most AQMs. In [76], Low *et al.* show that such AQMs and the source algorithms can be interpreted as distributed computational elements in solving the network wide utility optimisation problem. Regardless of exactly how the $p_l(t)$ congestion notification signal is computed by a particular AQMs, so long as in the steady state the AQM controls the packet arrival rate to match the target capacity of the link, $x^l(t) = c_l$, the AQM is in effect solving the same utility maximisation problem.

This distributed optimisation problem as set out in [76] and [62] have inspired a great deal of research. For example, a number of papers on the idea of congestion based pricing have emerged from the Cambridge-Microsoft research centre. In [46] an empirical study of competing source control algorithms was performed, to explore source behaviour in a competitive game environment. The authors of [65] investigate the choice of utility functions based on the requirements

of the application. In [72] an exploration of ECN marking to support the economic framework is made. It is clear from numerous works that the utility optimisation property of the Internet is the current paradigm for understanding the network in recent research.

2.4 Active Queue Management - AQM

2.4.1 Introduction

In this section, we will review existing proposals for AQM algorithms. There is a large body of AQMs proposed and we will limit our review to some key AQMs which represent the key paradigms. In following chapter, we will analyse the key AQMs in more detail.

Let us have a closer look into the structure of an AQM algorithm as shown in Fig. 2.6. The AQM algorithm controls the arrival rate of packets into the queue, by ECN marking or packet dropping to generate the congestion signal that controls the source rate. In the optimisation problem analogy, determining the right feedback signal is akin to finding the right price that controls the demand of the sources to match to the target link capacity. It has been shown that different AQMs [77] [76] solve the underlying utility optimisation problem as defined in Section 2.3.5, however with differences in properties such as steady state backlog in the link, speed of converge, transient response, and stability regions.

We will now examine these differences and discuss the aims of AQMs in general. The aims of flow control include (a) fairness (b) utilisation (c) quality of service (d) good transient properties and (e) scalability.

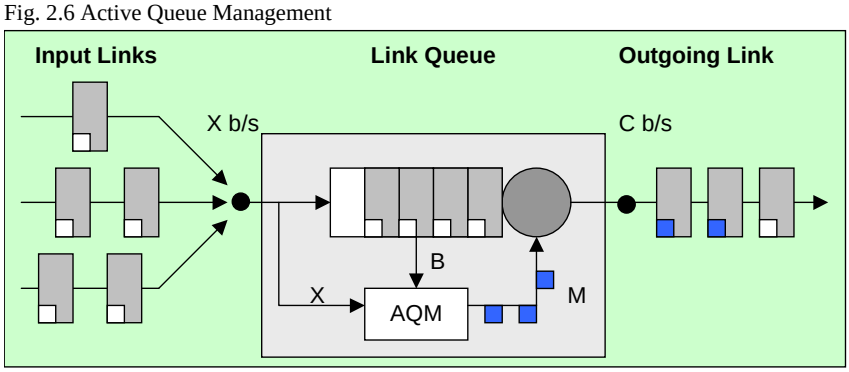
Fairness principally refers to the rate allocation that occurs in steady state, and we will see in Chapter 6 that this is chiefly controlled by the way links signal congestion, and the source algorithms. For the Internet, fairness is utility maximisation. We will explore this paradigm of fairness, and alternative ones later in this chapter in Section 2.5 and in Chapter 6. Utilisation refers to the AQM's ability to drive the link at some target capacity, and controls application throughput. However, the greater the utilisation, the greater the queuing delays experienced. In Chapter 3, we will discuss how several current AQMs inadvertently drive the network at very

high utilisation, causing large queuing delays, and explore new proposals for AQMs which can better control utilisation and delay.

In the context of AQM design, Quality of Service (QoS) means packet delay and packet loss. We have discussed the link between utilisation and delay. One way packet loss occurs is when the buffer overflows, so it is important for the AQM to have good dynamic properties, and quickly control the source rate to the target capacity. If the controller is too slow, the network may experience under-utilisation or buffer overflows when the transmission rate is not matched to the capacity. If the controller is too fast, it may become unstable, and cause rates to oscillate. We will explore stability and some dynamic properties of AQMs in Chapter 3.

For the Internet to be scalable and fully distributed, the AQM algorithm must only use local information to achieve the listed aims. Therefore to determine the appropriate dropping or marking rate *M*, the AQM can only observe information such as the packet arrival rate *X*, the backlog in the queue *B*, and link capacity *C*. The problem in designing an AQM algorithm is how to combine this state information to generate the appropriate feedback signal, so that the source rates can be controlled quickly to achieve the desired utilisation of the link.

Fig. 2.6 Active Queue Management



The design of the AQM algorithm has received particular attention of researchers, because it is currently the most significant bottleneck in the QoS performance of the best-effort Internet. As in [78], we will consider QoS, as the packet loss rate and delay. Currently, the tail-drop queue is effectively the default

and practically the only AQM on the Internet. Unfortunately, the tail-drop AQM requires unnecessarily large backlogs and packet loss in queues and research points out that replacing this component alone would significantly improve the performance of the network.

Today, applications that demand low packet delays, such as VoIP, receive poor performance on the best-effort network due to these delays that range from tens to hundreds of milliseconds. To highlight just how much latency is an issue on the current Internet, Fig. 2.7 is a measurement of RTTs experienced on September 11, 2001 [83], when a surge of communication occurred due to the tragic events of the day. Some links experienced delays in the order of seconds. However, we will demonstrate that large latency in the network during times of heavy load is not necessary. Modern AQMs remove the large backlog that the tail-drop AQM induces in the network.

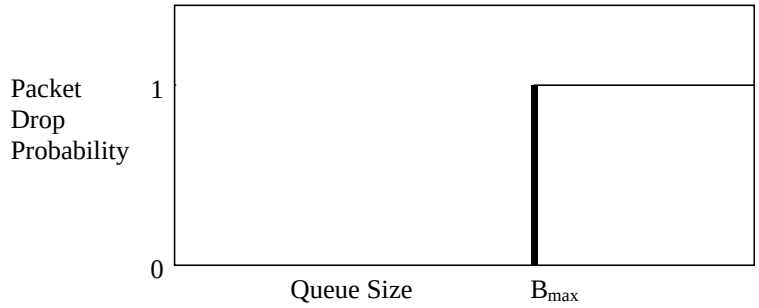
Fig. 2.7. Internet Packet Delays 11/9/2001.

2.4.2 Tail-drop

The tail-drop algorithm was not designed to be an efficient AQM. It is simply a queue which, when filled to its maximum, overflows and drops any subsequently arriving packets. However, we can interpret it as an AQM, which measures the

backlog to determine the congestion level. No congestion is detected by the tail-drop algorithm until the queue is full. As shown in Fig. 2.8, when the queue is full, the maximum congestion signal is generated because all of the arriving packets are dropped. Once sources detect lost packets they slow down and the arrival rate of packets to the queue will be less than the capacity of the link and the packet backlog in the queue decreases. Then, when the buffer is not full, no congestion feedback signal is generated by tail-drop algorithm and the source rates increase until overflow happens again. We can see that the tail-drop AQM results in a cycle of decrease and increase of rates around the point where the buffer is nearly full. The actual mean size of the buffer depends on the load on the link. The tail-drop algorithm is incapable of generating any feedback signal (price) unless the buffer is full. This is why the current Internet suffers long queuing delays and long RTTs.

Fig. 2.8 Tail-Drop Dropping Probability vs Queue Size



Most work on Active Queue Management uses the tail-drop queue as a lower bound for performance comparison. However, in [25] a basic mechanism is used to improve the fairness between flows of the tail-drop queue. We will now introduce AQM proposals that give performance superior to the tail-drop AQM.

2.4.3 RED Random Early Detection AQM

The most well-known AQM algorithm is the Random Early Detection algorithm developed by Sally Floyd and Van Jacobson in 1993 [38]. The design of RED was a response to some of the problems with tail-drop queue. The Internet Engineering Task Force recommends the deployment of RED in RFC 2309 [16], and

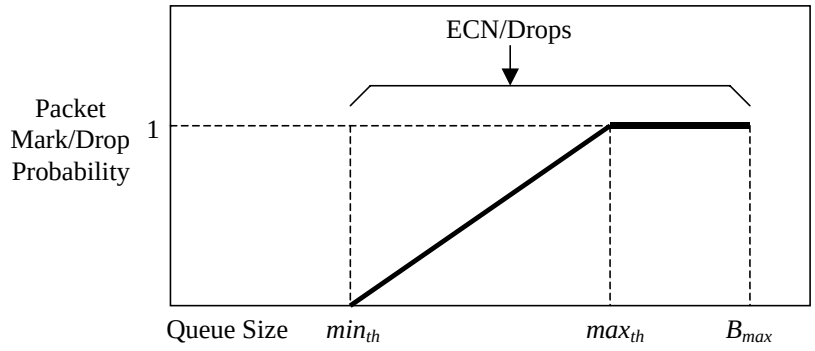
indeed it is widely implemented in routers today. However, by and large it has not been switched on, and doubt has arisen about the degree to which it can improve network performance [85]. In this section, we will describe RED and discuss some of the problems with the RED paradigm.

RED is a descendent of the DECbit congestion avoidance scheme [58]. RED uses queue length as a congestion measure. The rate of congestion notifications generated are directly related to the time averaged queue occupancy. The congestions notifications are either packet drops or ECN marks if the packet is ECN capable. The time-averaged queue occupancy is computed using an exponential averaging scheme, where w_q is an averaging time constant, and $B(t)$ is the instantaneous queue occupancy:

$$Avg(t + 1) = (1 - w_q).Avg(t) + w_q\, B(t)$$

When the averaged queue length exceeds a minimum threshold, congestion notifications are generated with a probability which is proportional to the excess queue length as shown in Fig 2.9. The parameter min_{th} determines the queue length at which congestion notification begins and max_{th} determines the point where congestion notification is performed on all packets. When the queue size exceeds the maximum queue length B_{max}, all packets are dropped.

Fig. 2.9 RED Dropping Probability vs Queue Size



RED addresses a number of problems that tail-drop exhibits. When a tail-drop queue overflows, there is a large number of packets drops all at once. This will result in the *global synchronisation* problem. This problem has been reported by

Floyd [38] and Hashem [50]. Global synchronisation is a cycle of under-utilisation following the burst of drops followed by a period of overload. It occurs because the burst of drops results in a large number of TCP sources reducing their window size at the same time. With RED, the early packet drops are randomised and spread over time. Packet marking/dropping is uniformly random for each packet. This reduces global synchronisation of sources.

Another problem with the tail-drop queue is Lock-Out. Lock-Out describes the effect when new TCP sessions receive significantly less bandwidth than established connections. This phenomenon is reported in [16], and stems from synchronisation phenomena.

A key aim of RED is to reduce the average queue length to accommodate short bursts in the queue. RED achieves a lower average queue size than tail-drop because it generates congestion notifications once the queue is greater than min_{th} unlike tail-drop which drops packets once the queue size is greater than B_{max}. The lower average queue size of RED will be demonstrated by simulation in the next chapter.

Despite some advantages of the RED AQMs over the tail-drop queue, RED exhibits a number of serious problems. In [79] Low *et al.* performs a control-theoretic analysis of TCP/RED and uncovers that this flow control mechanism eventually becomes unstable as RTT delay increases, or when network capacity is increased. In the following chapter, we will review recent work [100] which analyses how stable control can be achieved for any number of flows, network delays, and capacities.

The instability of RED is discussed in numerous papers. In [14] oscillatory backlog behaviour is discussed. In [86] May *et al.* argues that the jitter that RED introduces demands such large jitter buffers of CBR applications (eg. audio streaming) that any delay performance improvements of RED are negated by the latency these jitter buffers introduce. In [21] the difficulty of tuning RED parameters to observe performance increase in web traffic is discussed. The oscillatory behaviour and possible instability of RED is again confirmed in [37] by a more

analytical approach. In “Reasons not to deploy RED” [85] it is questioned whether RED gives any benefit over tail-drop.

The RED algorithm is a basis for many other AQM proposals which seek to enhance the performance of the basic RED structure. For example, in [30] a method to balance bandwidth among flows more equally is introduced. The proposal called “RED+” in [129] adds the ability to identify and discriminate against high-bandwidth, “unfriendly” flows, which do not implement congestion control. The variant “SRED”, Stabilised RED, presented in [95], tunes RED’s parameters by estimating the number of connections present.

In Chapter 3 of this thesis, we will show that measuring the queue occupancy to determine congestion is a fundamental structural constraint of RED and RED-like AQMs. Rather than continuing to enhance the RED AQM structure, we will investigate structurally different paradigms for AQM design. We will show that new AQMs whose structure is based on packet arrival rate measurement, remove the limitations and performance issues of the RED-like AQMs. We will also develop techniques for applying these rate-based AQMs in applications where RED AQMs were previously suggested.

2.4.4 BLUE AQM

In their paper [33], Wu-Chang Feng *et al.* develop a new AQM called BLUE which attempts to address some of the problems of RED. BLUE uses packet loss and link under-utilization events, rather than queue size, to adjust the rate of congestion notification. The congestion notification rate p_m is increased at a set rate if the queue size exceeds a threshold L, and it is decreased if the link is idle. The notification rate increased by d_1 every *freezetime* seconds when the queue is over the L threshold. The notification rate decreases by d2 every *freezetime* seconds when the link is idle. In this way, the congestion notification rate p_m converges to a value which controls the arrival rate so that queue is below the threshold L, and the link is not idle.

Fig. 2.10 BLUE AQM

```
Upon Packet loss or (B(t) > L) event:
      if ((t-last_update) > freeze_time) then
            Pm=Pm+d1
            last_update=t

Upon link idle event:
            if ((t-last_update) > freeze_time) then
                        Pm=Pm-d2
            last_update=t

where t = current time
```

In [33] it is shown that BLUE achieves a more stable marking probability than RED. Although Wu-Chang Feng *et al.* give no analytical results to support their claim that BLUE gives more stable control than RED, the paper presents a number of simulation studies which compare the performance of RED and BLUE and show that BLUE results in a lower packet loss probability for the scenarios presented. It is claimed that BLUE has a more stable queue size than RED and we reproduce the queue length results from [33] for an experiment where the number of TCP connections is increased over time the time axis.

Fig. 2.11 Queue Length Plots of RED and BLUE

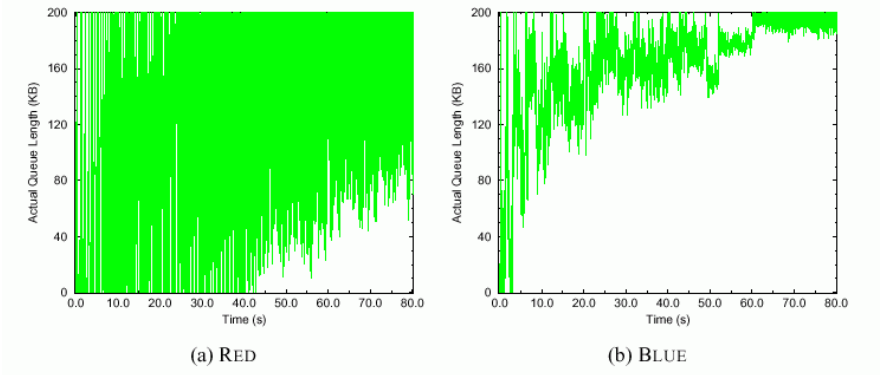
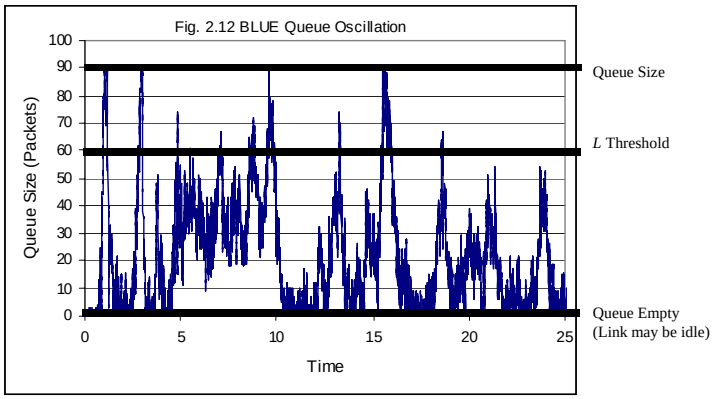


Figure taken from [33].

Despite the improvements claimed in [33] about BLUE a number of shortcomings were uncovered in our performance evaluation of this AQM. Although the queuing in BLUE may appear more stable than RED from Fig 2.11, if we look at

the long term behaviour for a certain load, we can see that BLUE still creates jitter in the network. Fig 2.12 shows the queue length of BLUE for a steady state load of 50 TCP connections. Note that the queue length wanders from 0 to above the threshold L during normal operation.



The design of BLUE necessitates that the queue wanders between 0 and the *L* threshold level in order for the marking probability p_m to be adjusted to an equilibrium level required for a given load of TCP connections. This is because p_m is not adjusted unless the queue is either 0 or *L*. Unless the aggregate arrival rate into the link *Y(t)* is matched perfectly to the link capacity *C(t)*, then the queue will decrease or increase, and cause adjustments to p_m when the link idle or queue threshold events are generated. Such pathological fluctuations in queue size result in delay jitter which is detrimental to the QoS of interactive applications, such as VoIP.

There are a number of other problems with BLUE which are reported in [123] and [69]. Hongwei Kong *et al.* [69] describe experimental results which indicate that BLUE performs poorly when the buffer size is small, because BLUE does not control the average queue level to be low, and when bursts of traffic arrive, significant loss can occur.

Although BLUE makes the first step of uncoupling queue occupancy and congestion notification, we have shown that there are some serious performance problems with the algorithm. For these reasons, we will not include BLUE in further

discussions. For a full comparison of BLUE to all of the AQMs discussed in this thesis, the reader is referred to [123].

2.4.5 REM AQM

Random Exponential Marking (REM) is introduced in [9] and developed in [6, 7, 76, 100]. REM is both a set of AQMs and a novel technique for communicating congestion information. We will first describe this communication scheme and how it differs from the previous AQMs, but we will focus on our main interest in this section, the AQM control law.

REM embodies a mechanism for the precise communication of link congestion prices, so that the link congestion state variable is exactly the congestion price as in the utility optimisation problem discussed in Section 2.3.5. The previously introduced AQMs, RED and BLUE, control the packet dropping/marking rate directly, and the packet dropping/marking rate is not exactly the congestion price in the optimisation problem. This is because the total dropping/marking rate that a source experiences over a path is not additive for links which drop/mark packets randomly and independently. As will be discussed in Chapter 6, the total dropping/marking rate is only approximately additive when each link's dropping/marking rate is small. REM ensures that the end-to-end congestion price signal, communicated by independent packet marking, is exactly additive as assumed by the optimisation problem.

A REM link marks a packet at link l with a probability based on the link price p_l state, and a global encoding constant ϕ ($1 < \phi$):

$$m_l(t) = 1 - \phi^{-p_l} \tag{2.11}$$

Assuming links mark packets independently, the overall probability of a packet being marked has an exponent with the sum of the link prices:

$$1 - \prod_{l \in L} (1 - m_l(t)) = 1 - \phi^{-(p1 + p2 + p3 \ldots)} \tag{2.12}$$

Because sources know the value of ϕ, they can compute the total end-to-end path congestion price. Therefore, in a complete deployment, REM requires a REM link algorithm and a source algorithm capable of decoding REM information.

In [6] it has been shown that inter-operation with the TCP-RENO source algorithm with just the link REM AQM algorithm deployed is possible. In this case, the price $p_l(t)$ state variable can be interpreted as the marking rate, just as the other AQMs discussed. Indeed for $p_l(t) << 1$, we can assume $p_l(t) = m_l(t)$ by (2.11). With this in mind, the three control laws PC1, PC2, PC3 presented in [6] can be interpreted as alternative AQMs:

$$\text{PC1: } p_l(t+1) = \gamma b_l(t) \tag{2.13}$$

$$\text{PC2: } p_l(t+1) = [p_l(t) - \gamma(x^l(t) - c_l)]^+ \tag{2.14}$$

$$\text{PC3: } p_l(t+1) = [p_l(t) - \gamma(\alpha_l \cdot b_l(t) + x^l(t) - c_l)]^+ \tag{2.15}$$

where $p_l(t)$ is the congestion notification rate, c_l is a target capacity just below the actual link capacity, $b_l(t)$ is the backlog, and γ and α are control gain constants which affect speed and stability of control.

It is clear that PC1 control law is very similar to the RED-like AQMs where the congestion notification rate is proportional to backlog. It is the PC2 and PC3 control laws that present a new approach to AQM design. As discussed in [76] PC2 and PC3 uncouple the congestion notification rate from the backlog at the link. PC2 and PC3 measure the arrival rate to the link to compute the congestion notification rate instead of using the backlog. The congestion notification rate is controlled by an integral controller, whose error term is the discrepancy between the aggregate arrival rate to the link and the target link capacity.

Note that PC2 and PC3 differ only in that PC3 adds a backlog penalty term to the control process, which makes the marking rate increase with greater rate if there is a backlog. This was found [6] to improve the transient response of the basic PC2 controller, and reduce the amount of backlog during transient periods when the load changes. The stability properties of PC3 are analysed in [99].

Further work by S. Low and co-authors in this area extends the analysis and improves the REM framework. Several of his papers focus on improving the convergence rate of the basic REM algorithm [7]. With a faster rate of convergence of arrival rate to the target rate, the buffer requirements, and jitter are reduced. In [7] an extension to the control equation based on a Newton-like algorithm is analysed. In [5] an approach using a deadbeat controller is used.

Experimental results in [6] indicate that the control laws PC2-PC3 are able to control the sources such so that the mean backlog at the link is substantially reduced compared to a tail-drop queue or RED. The result in [6] indicate that these AQMs are able to operate with very low backlog, and maintain a high link utilisation. The PC2-PC3 depart significantly both in performance and design from the RED or tail-drop algorithm and we will discuss the differences in detail in the next chapter, where we will develop a new AQM based on a similar principle of rate measurement.

2.5 Fairness Properties

2.5.1 Introduction

In this section, we will outline a number of different philosophies for rate allocation to users. There are different approaches to what a ‘fair’ rate allocation is. In the previous section we have outlined a number of different approaches of AQM design. All of these approaches lead to roughly the same steady state rate allocation to sources, since they are all solving the underlying problem of maximising the total user utility. There are, however, other fairness objectives which have been proposed for the distribution of bandwidth between sources which optimise different criteria. In this section, we will review the most significant fairness criteria which have been proposed in literature and which will be important for Chapter 6 where we will depart from the utility maximisation optimality and design a network with a different fairness property.

2.5.2 Max-Min Fairness

The Max-Min fairness criterion is that the minimum rates through the network should be maximized whilst remaining feasible. A formal optimality

condition for Max-Min is stated in [11] as follows. A feasible rate vector r is an optimal solution to the Max-Min problem iff for every feasible rate vector $\boldsymbol{r}$ with $\boldsymbol{r_i} > r_i$, for some source i, there exists a source k such that $r_k \leq r_i$ and $r_k > \boldsymbol{r_k}$. Put simply, a rate vector is Max-Min if it is feasible and no flow can be increased while maintaining feasibility without decreasing a smaller or equal flow.

To illustrate the Max-Min fairness property, consider the network in Fig 2.13. The Max-Min allocation for this network is: Source 1 = 1.5 Mbps, Source 2 = 0.5 Mbps, and Source 3 = 0.5 Mbps.

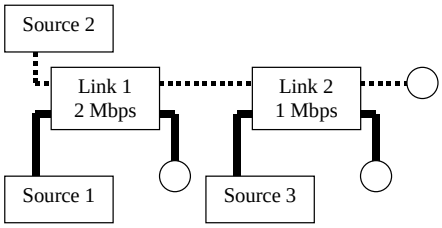


Fig 2.13. Global Max-Min Not Achieved for SumNet

The Max-Min fair criterion arises from fundamental work in social justice such as John Rawls' "A theory of Justice" [106]. The philosophy of the utility optimisation model is described as 'utilitarianism' by Rawls. The idea of utilitarianism is to give the maximum benefit to the whole society, sometimes at the cost of some individuals. On the other hand, the Max-Min philosophy is to attempt to give equal amounts of resource to everyone, and to ensure that the most disadvantaged are disadvantaged as little as possible. A rigorous mathematical discussion of these different approaches to fairness is presented in [43].

In Chapter 6 we will discover that it is really only by coincidence that the Internet bandwidth distribution mechanism is based on the philosophy of utilitarianism. The Internet's adherence to this fairness criteria stems from the legacy of how congestion information is communicated. In Chapter 6, we will introduce a new architecture for an Internet-like network, which instead of optimising utility, is based on the Max-Min fair rate allocation.

2.5.3 Weighted Max-Min Fairness

The weighted Max-Min fairness criterion extends the basic Max-Min fairness criterion to allow flows to have differentiated demand for bandwidth. For each source *i* there is a weight w_i which describes the source's relative bandwidth need. If we construct a rate vector r_i where $r_i = w_i.x_i$, then the rate vector r_i is weighted Max-Min fair, if the rate vector x_i is Max-Min fair. Weighted Max-Min fairness can be described by considering that a single user, who has r_i bandwidth, is actually w_i sources combined as a group in the Max-Min case.

2.5.4 Proportional Fairness

The formal definition of the proportional fairness criterion is that a vector of rates *x* is proportionally fair if it is feasible given the system capacity constraints and if for any other feasible vector *y*, the aggregate of proportional changes is zero or negative for the set of sources *S*:

$$\sum_{s \in S} \frac{y_s - x_s}{x_s} \leq 0$$

If all sources have a utility function of the form $U_s(x_s) = \log x_s$, where *s* is the index of a particular source, then the utility optimising network leads to the maximum utility solution which is proportionally fair [62, 119]. The proportionally fair solution is the Nash bargaining solution [56] [28].

2.5.5 Weighted Proportional Fairness

If the source's utility function is of the form $U_s(x_s) = w_s.\log x_s$, where *s* is the index of the source, and w_s is a unique weight which reflects the relative bandwidth need of the source *s*, then the utility maximum solution for a network is a weighted proportionally fair one [62]. The formal definition of the weighted proportional fairness criterion is that a vector of rates *x* is weighted proportionally fair, for a weight vector *w*, if it is feasible given the system capacity constraints and if for any

other feasible vector *y*, the weighted aggregate of proportional changes is zero or negative for the set of sources *S*:

$$\sum_{s \in S} w_s \frac{y_s - x_s}{x_s} \leq 0$$

Note that the Internet is not proportionally fair, nor is it weighted proportional fair, since there is are heterogenous sources with utility functions other than logarithmic.

2.6 TCP Unfriendly Problem

2.6.1 Malicious Flows

In the previous sections, we have described how the load control model of the best-effort Internet leads to a source rate allocation which optimises the total utility in the network. However, a key issue of the utility optimisation approach is that so long as there is no real incentive for reacting to congestion, *unfriendly* sources may manipulate their benefit functions to gain a greater share of bandwidth, and utility maximisation will not be achieved.

Since, as discussed previously, a source algorithm can in general be described by a local benefit optimisation, $B = U(x) - xc$, the cost c must be tied to a real disincentive for transmitting data, if *unfriendly* users are to be congestion controlled. If the cost c is only tied to the rate of ECN marking, than there is clearly no disincentive to reduce rate, and *unfriendly* users will transmit more to increase their benefit *B*. If the cost c is tied to packet drops, there is some disincentive to reduce rate, since dropped packets may hinder the *unfriendly* user's application, however it may still be in their net interest to transmit more, and suffer losses, to increase the net throughput. So long as there is no congestion based pricing, and the cost c is not tied to positive monetary cost, where c is some *$/bit* in the user's bill for internet usage, with $c = 0$, the benefit function, in monetary terms, is reduced to $B = U(x)$. In this case the *unfriendly* user increases their benefit by increasing their transmission rate.

The difficulty in dealing with *unfriendly* sources in a network which optimises utility is that it is impossible for the network to know whether the

unfriendly transmission rate x_{uf} is due to an unfriendly source that is ignoring the cost c, $B_{uf} = U_{uf}(x_{uf})$, or a friendly source $B_f = U_f(x_f) - x_f c$, whose different utility function $U_f(x_f)$ results in the same transmission rate $x_{uf} = x_f$ given the same congestion signal c. Only when there is a real disincentive for ignoring c, can we be sure that the type of social optimality described by Kelly, Low or Gibbens [45] can be achieved.

To better illustrate the issue of *unfriendly* sources faced on the Internet, consider the case of a user transmitting a constant bit-rate UDP stream at X_{UDP} Mbps over a C Mbps link that is shared with TCP traffic of other users. Since the UDP stream does not react to congestion, it will squeeze the reactive TCPs sessions to share the remaining $C - X_{UDP}$ Mbps of capacity. The share that the TCP sessions receives may be substantially lower than the UDP stream. In the utility optimisation model, one can interpret this situation by deducing that the UDP stream has a higher utility attributed to sending at the high rate than TCP, and the utility optimal solution is to give as much bandwidth to the UDP stream as it needs. However, it is possible that the UDP user is *unfriendly* and has less need for the bandwidth than the TCP user, but has manipulated the source to increase its attained bandwidth. If the UDP user took into account the disincentive that c represents, the transmission rate would be lower.

In the previous sections, we introduced a number of fairness criteria that, rather than basing the rate allocation on a source property (i.e., utility function), assume all sources have an equal right to capacity. Fairness criteria such as Max-Min fairness, seek to equalize the allocated bandwidth to all sources, rather than optimise global utility. Since there is incentive for sources on the Internet to be *unfriendly* in the presence of *friendly* flows, utility optimality is not achievable, and the rates of *friendly* sources suffer. Rate distributions criteria that equalize rates, rather than optimise utility, have been suggested as a solution to the problem of *unfriendly* flows. In this section, we will review a number of proposals which enhance existing AQMs to enforce more loose notions of rate equality, which are local to the link managed by the AQM.

In principle these schemes monitor flow rates and aim to enforce flows to transmit at rates within bounds of what some normative *friendly* source would transmit. A wide selection of literature has adopted the TCP's benefit function as the normative response to congestion eg: [40, 108] . Sources which transmit at a greater rate than TCP would attain in the same circumstances are deemed *TCP unfriendly*.

We will now review proposals which attempt to police flows in the best-effort network in order that *TCP unfriendly* flows are not able to abuse the network resources. Of course, these proposals result in the rate allocations which no longer maximises the utility, and instead result in more equal rate allocation between flows of heterogenous utility functions. Amongst a number of proposals, the more interesting ones which we will describe in detail include CHOKE [101] [120], RED with a penalty box [39], Core stateless router [114], and Stochastic Fair BLUE [35]. Asides from these there many others, including; RED+ [129], SRED [95]. All of these methods are enhancements to the link AQM algorithms that add a degree of per-flow rate control, whilst having minimum per-flow storage and processing of information.

In general, the authors of these proposals do not discuss the relationship between the utility optimisation approach and equalizing rates. These schemes move away from the utilitarian philosophy of utility maximisation, towards the philosophy of rate equality. Although the papers claim to 'improve' fairness between flows, it is important to realize they are in fact changing the paradigm of what fairness is. The advantage of rate equality paradigms is that they are enforceable, since one can measure and control rate, but one cannot readily measure what the real utility of an application is.

In the long term, if congestion based pricing is adopted, then controlling unfriendly flows will not be necessary, as there would be real incentive for the users to only transmit at a rate which reflects the true utility of their application. Transmitting above this rate would not maximise the *benefit* to the user and they would be paying more for their service than their actual need warrants.

2.6.2 CHOKe

The CHOKe [101] algorithm is a stateless active queue management scheme which aims to control source rates so that an equal sharing of bandwidth is achieved at the CHOKe link. CHOKe is short for “CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows”. CHOKe tries to penalize high-bandwidth unresponsive flows by dropping more of their packets.

The CHOKe algorithm is interesting because of its performance as well as its simple and elegant implementation. The algorithm is local to a link and requires no per-flow information. The essence of the algorithm is that, when a packet arrives, a random packet is picked from the queue. If the randomly picked packet comes from the same source as the newly arrived packet, both packets are dropped. In [120], Wang *et al.* show analytically that this simple algorithm controls high-bandwidth unresponsive flows such that as their bandwidth increases, their share of the link capacity decreases to 0. This can be intuitively understood, since the probability of dropping a packet increases for a high-bandwidth flow since (a) more packets arrive and trigger comparison and drop operation (b) the randomly picked packet is likely to be from a high-bandwidth flow. Results in [101] indicate that CHOKe is able to control high-bandwidth unresponsive UDP flows, so that TCP connections can share the link more equitably.

2.6.3 RED extension to encourage Congestion Control

In their paper “Router Mechanisms to Support End-to-End Congestion Control” [39], S. Floyd and K. Fall address the problem of non-congestion controlled best-effort sources. They propose an extension to the RED algorithm which is able to identify sources which are gaining more bandwidth at the link than a TCP session would. Although the proposal is centred around RED, it is conceivable that the proposal, at least in spirit, is applicable to other AQMs. Once the method identifies unresponsive flows, these flows are rate controlled to within the TCP share, or are even punished by being limited to a lower rate so that there is incentive to implement TCP friendly flow control.

The mechanism avoids per-flow processing by observing the RED packet drop history to identify high-bandwidth unresponsive flows. The paper [39] empirically validates this mechanism of being able to identify high bandwidth flows by observing the RED packet drop history. A model of TCP is used to determine what rate a TCP flow would achieve given the same packet drop history, and if the suspect flow transmits more than this, it is identified as unresponsive. Once identified, the flows are put onto a list of misbehaving flows whose rate is limited.

One limitation of the method is its scalability. Although per-flow processing is done only for unresponsive flows, if the number of unresponsive flows is high, the requirements on core routers becomes prohibitive.

2.6.4 Stochastic-Fair Blue

A novel proposal for controlling non-congestion-controlled high-bandwidth sources is the Stochastic Fair Blue (SFB) scheme introduced in [35]. This scheme extends the BLUE AQM to identify and rate limit flows which do not respond to congestion control.

The SFB approach avoids per-flow monitoring in a unique way. SFB assigns flows pseudo-randomly into a number of different groups. Each group has a separate queue and has associated with it the BLUE AQM which computes p_m for that group. Each flow belongs to N groups, and there are NL groups in total. The marking/dropping rate for a flow is the minimum marking rate of all of the groups that the flow is assigned to. If a flow is unresponsive to congestion notification, it will drive the p_m value of all of the groups it is in to 1, since it will cause the queue occupancy in each group to be above the threshold. Since the dropping rate of all of the groups that the flow is associated is driven to 1, the minimum of all of the groups is driven to 1. This will cause the unresponsive rate to be rate limited. Since flows are assigned pseudo-randomly to groups, it is likely that only one flow, the unresponsive flow, is in each of the groups that has dropping rate 1. Therefore only the unresponsive flow is rate controlled.

As described in [35], there is a clear limitation to this method. If the number of unresponsive flows is large, a large number of groups needs to be maintained to

ensure a low probability that responsive flows which happen to be mapped in the same *N* groups as an unresponsive flow are not punished. Some mitigation of this is presented in [35].

2.6.5 Core Stateless Fair Queuing (CSFQ)

In their article "Core-Stateless Fair Queuing: Achieving Approximately Fair Bandwidth Allocation in High Speed Networks." Ion Stoica *et al.* [114] tackle the problem of per-flow rate policing at core routers by introducing a distributed approach for queue management. The paper presents a novel technique which allows per-flow flow control without the computational overhead of keeping flow state at core routers.

Ion Stoica *et al.* organizes the network into islands of core and edge routers where low capacity edge routers connect to high capacity core routers. Their scheme depends on the assumption that edge routers are able to collect per-flow information about the flows, since they operate at lower rates than core routers. The edge routers measure the flow rate of a particular flow and insert this measurement into the packet header which is communicated upstream to the core router. The CSFQ scheme eliminates the problem of per-flow processing in the core for policing flow rates, by distributing the per-flow processing into the low-capacity edge of the network.

With the CSFQ, core routers need only execute a simple fairness algorithm for policing the bandwidth of individual flows. A FIFO queue in the core router drops packets with a probability which depends on how much the rate of a flow exceeds the estimated fair flow rate. Lets assume $\alpha(t)$ is the fair flow rate, and r_i is the flow rate of a flow *i* as marked in the packet header by an edge router. Then the dropping rate at the core router for packets of this flow is:

$$\max\left(0,1-\frac{\alpha(t)}{r_i(t)}\right) \tag{2.16}$$

The core router must estimate the fair share allocation $\alpha(t)$ from the estimate flow rate and its own knowledge of link capacity. Shenker describes a method for approximating the fair share rate. The fair share rate $\alpha(t)$ is approximated in two

parts. If the link is not congested, that is, if the aggregate arrival rate is below the link capacity *C*, then $\alpha(t)$ is set to the maximum flow rate $max(r_i(t))$ of all the active flows. If the link becomes congested, then $\alpha(t)$ is updated according to the algorithm

$$\hat{F} = \sum_i \min(r_i, \alpha(t)) \tag{2.17}$$

$$\alpha(t+1) = \alpha(t) \cdot \frac{C}{\hat{F}} \tag{2.18}$$

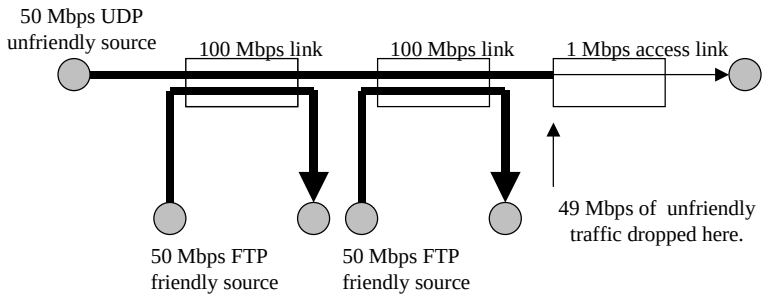
Intuitively, Shenker's algorithm can be seen to control $\alpha(t)$ such that in steady state the portion of the aggregate arrival rate which is not dropped, $\hat{F}$, is matched to the capacity *C*. Packets exiting a core router are relabeled with their new flow rate L_{new}, which is either the fair share rate α, or the same rate as at the entry to the router, L_{old}, if that rate is below the fair share rate:

$$L_{new} = \min\ (L_{old}, \alpha)$$

The new label is used by subsequent CSFQ routers to perform the CSFQ packet dropping and relabelling mechanism.

The approach of Shenker's CSFQ algorithm, CHOKe, SFB and the extended RED algorithm follow the paradigm of enforcing the packet dropping for policing flow rates locally at the link (link-local) where the flow rate exceeds the fair capacity. A key problem of the link-local paradigm is that for high-bandwidth unresponsive flows, a lot of capacity may be wasted inside the network before the congested link drops the packets. This paradigm does not prevent denial-of-service attacks [25, 36], or otherwise high-bandwidth unresponsive flows from wasting resources of the network as illustrated in Fig. 2.14. In Chapter 6, we introduce a distributed method for achieving Max-Min fairness inside a network, which we extend to provide flow policing at the edge of the network. This proposed method avoids the possible capacity wastage of the link-local policing approach.

Fig. 2.14 Link-Local Flow Policing



2.7 Capacity provisioning

2.7.1 Introduction

We began this chapter by discussing that there are two dimensions to the problem of congestion control: 1) Load Control as well as 2) Capacity Provisioning. We have discussed best-effort load control and now we will turn our attention to Capacity Provisioning.

Load Control ensures that hosts do not send packets onto the network at a rate that exceeds the network's capacity. However, Load Control does not in general control the rate at which packets are generated by the applications. In general, the application layer may pass data to the transport layer at a rate which is independent of the source's transmission rate. If this rate is greater than the rate at which the network can serve the packets, backlog will inevitably build inside the source. Load Control techniques have their place in ensuring that this backlog is stored in the sources and not in the network. This allows real-time interactive applications such as VoIP and non-real time applications such as FTP to coexist on the same network, since network delay is not induced for all applications. However, to remove backlog even at the source for applications which have a constant amount to send irrespective of the capacity, enough capacity must be provided inside the network to meet the demand of these applications. In Chapter 5, we will discuss the interaction of different types of applications, and the relationship between capacity provisioning and load control in more detail.

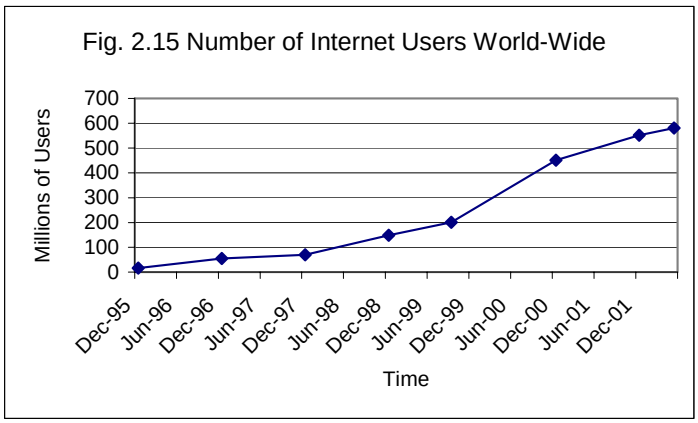
There has been a wide body of work in the very well established area of tele-traffic performance analysis. Starting from fundamental work of M/M/1 queues as

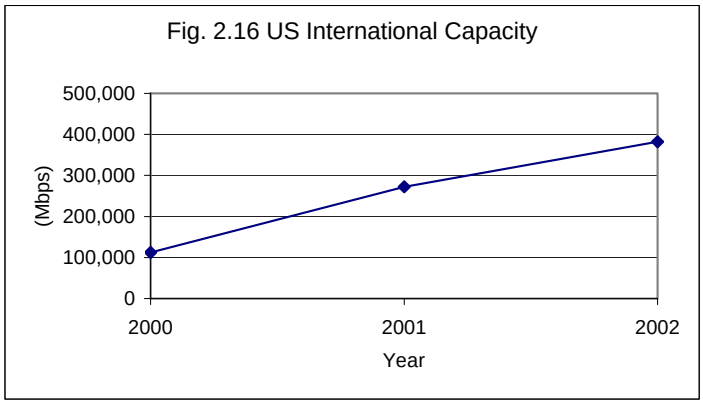
described by Kleinrock in [68], increasingly sophisticated methods of modelling traffic sources and modelling the performance of networks have been proposed. We contribute to this area of research in Chapter 5, where we develop an analytical tool for determining the delay performance across a multi-class access link. This tool is useful for dimensioning of capacity so that delay performance requirements are met.

2.7.2 Performance scaling with Capacity

The need to meet the requirements of increasingly more bandwidth hungry applications and a growing user population puts pressure on the amount of capacity that needs to be provisioned on the Internet. As a final note for our introduction, it is worth considering the future and what impact the continuing increases in capacity and user population have on the nature of congestion, and the QoS experienced. We will discuss the impact of network growth on load control in Chapter 3 and 6, so it is appropriate we address this issue from the capacity provisioning perspective also.

To put our discussion into a context of some real figures for Internet growth, Fig. 2.15 shows the number of world-wide users of the Internet (data from [92]) and Fig. 2.16 plots the growth of the US International Internet capacity (data from [84]). Let us assume that the number of users is roughly in proportional to the number of transmission sources.





It is clear there is steady growth, both in capacity and in the number of users of the Internet. In [63] the question of how such growth affects performance is analysed.

In [63] the effect of scaling the number of arrival processes, the rate of the arrival process and its volume are analysed. A key fundamental finding in [63] is that the packet queuing delay suffered decreases as the number of multiplexed sources increase. The queuing delay also decreases if both the arrival process rate and the capacity are increased by the same factor. To explain this, let us consider an arrival process *X*, where *X*[*s*,*t*] describes the amount of work arriving in the interval between time *s* and *t*. This process feeds a queue with service capacity *C*, and a queuing process *Q(t)*, as described by

$$Q(t) = \sup_{u \geq 0} \{ X[t-u, t] - Cu \ \} \tag{2.19}$$

Suppose that the input to the queue is scaled by the number of multiplexed processes *c*, is sped up by a factor *b*, and is increased in volume by a factor *a*

$$\sum_{i=1}^{c} a \cdot X[b \cdot s, b \cdot t] \tag{2.20}$$

In this case, the mean amount of work arriving at the queue has increased by a factor *abc* and given that the capacity C is also increased by this amount, the queuing delay τ is:

$$\tau(a,b,c) = Q(a,b,c)/(abcC) \tag{2.21}$$

There is no effects in scaling *a*:

$$\tau(a,b,c) = \tau(1,b,c), \tag{2.22}$$

However, increasing the rate of the arrival process and the server, decreases the queuing required:

$$\tau(a,b,c) = b^{-1}\tau(a,1,c) \tag{2.23}$$

Also, Kelly shows that increasing the multiplexing, *c*, results in a decrease in the queuing required for traffic characterised by the Hurst parameter *H*:

$$\tau(a,b,c) = c^{-1/(2-2H)}\tau(a,b,1) \tag{2.24}$$

These two results show that as the capacity of the Internet is increased, and so is the number of traffic sources traversing across the network, the network latency will decrease. This is an important prediction for the future of Internet performance. Put simply, it means that as the network grows, the capacity of the Internet will be increasingly more efficiently utilised, and the delays experiences by users will decrease.

2.8 Conclusions

In this chapter, we have introduced some of the key concepts of congestion control. We separated congestion control into the two dimensions of load control, and capacity provisioning, both of which we will address in this thesis. We described

how the behaviour of load control in best-effort networks is related to economic systems. We have also discussed how in principle bandwidth differentiation can be achieved, through use of source algorithms, which reflect the utility of the application. We have also described how this system can break down when malicious users are present, and some measures which can be taken to provide incentives to use the network resources fairly.

In the following two chapters, we will develop the design of AQMs as well as techniques for the deployment of AQMs which provide very low latency. In Chapter 5, we will change our focus onto capacity provisioning and develop techniques for modelling links which help in determining the capacity requirements. In Chapter 6, we will investigate the nature of congestion signalling, and propose an alternative best-effort Internet-like network, as well as address the problem of malicious users.

The performance problems faced by today's best-effort network are not fundamental ones. The key components of the best-effort load control system, the source and the link AQM algorithm, impact on the performance of the network as we have described. There are some concrete steps, which can be made incrementally to improve the network performance. The deployment of modern AQM algorithms in switches and routers, will improve the delay performance for all applications using the network, and will enable multimedia and interactive applications to coexist on the best-effort network.

CHAPTER

3

Chapter 3: Rate-Based Active Queue Management

3.1 Introduction

The area of active queue management in IP networks has caught the attention of researchers since its significance to network performance was realised when S. Floyd and V. Jackobson first proposed the RED [38] algorithm that modified the behaviour of the basic tail-drop queue. The development of RED inspired a paradigm of research into RED and RED-like algorithms. Various RED-like algorithms were developed to enhance different aspects of performance of the basic RED algorithm. These include SRED [95], DSRED [125, 126], LDA/OPF [127], DRED [128], BLUE [33] and Adaptive-RED [41]. Research into these algorithms has comprised of experiments with real networks, simulation studies and analytical studies.

A common feature of the RED-like algorithms is that they use the amount of backlog at the link as a measure of congestion. We will call these algorithms as *backlog-based*. Recently proposed AQMs, such as REM [9], differ fundamentally from the RED-like algorithms, because these use the packet arrival rate at the link as the congestion measure. In this chapter, we will show that backlog-based AQMs, such as the RED-like algorithms, make backlog in the network inevitable during times of congestion. On the other hand, rate-based AQMs control the sources without necessitating backlog, or increases of backlog, during times of congestions. Rate-based AQMs make it possible to clear the network of backlog keeping the packet delays low for all applications sharing the network. Applications with bulk data to transfer are forced to queue packets at the source, rather than having packets waiting inside the network, choking all other applications.

We will show that rate-based AQMs permit a wider variety of control structures, which permit better controller design than backlog-based controllers. For these reasons, in this and subsequent chapters, we will focus on the development of rate-based AQMs and their applications.

The chapter is divided into three sections. In the first section we will survey and existing AQM proposals. We will analyse rate-based AQM algorithms and compare them to backlog-based AQM. We will review some control theoretic results about AQM control found in literature, to describe how the stability of all types of AQM controllers is affected by the size of the network.

In the second part of the chapter, we will introduce a new rate-based AQM, called GREEN. We will analyse its properties and compare it to other well-known rate-based and backlog-based AQMs. We will then review a control theoretic study of GREEN to understand its performance and stability characteristics. A simulation study will be used to compare a number of well-known AQM algorithms to GREEN. We will compare GREEN with tail-drop, RED, and REM. The simulations will demonstrate how rate-based algorithms can provides high link utilization whilst maintaining low delay and packet loss.

In the final part of this chapter, we discuss the deployment of rate-based AQMs in real networks, and how to overcome some performance issues that arise when rate-based AQMs are deployed. We will discuss our implementation of rate-based AQMs in a LINUX-based router and uncover some previously unexpected phenomena that affect the performance of rate-based AQMs in networks. We show that when traffic includes non-ECN capable TCP, modifications to the basic rate-based AQM controller are required. We analyse this phenomenon and provide a framework for deploying rate-based AQMs in real networks with heterogenous sources which are ECN and non-ECN capable.

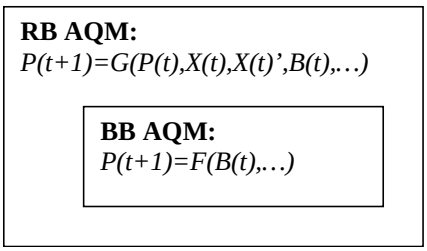
The analysis and development of rate-based AQM algorithms in this chapter is a basis for the development of further techniques in subsequent chapters. In Chapter 4, we extend rate-based AQM techniques to multi-queue systems, and in Chapter 6 we develop an entirely different signalling infrastructure for congestion control.

3.1.1 Overview of AQM structures

As evident from the previous chapter, there is a large variety of different AQM algorithms in literature. Some were developed based on intuitive reasoning and others by a modelling and analytical approach. In this section we will attempt to give some structure to the variety of AQMs available, and describe the structural limitations of some approaches.

As mentioned in the introduction, broadly, there are two paradigms of congestion control algorithms, characterised by the way they observe congestion. Backlog-based (BB) control, like tail-drop, RED and WRED, measures only the number of packets in the buffer to determine the severity of congestion. Arrival rate-based (RB) control schemes, such as REM or GREEN [123], measure the packet arrival rate, as well as possibly the backlog, to estimate congestion and determine *P(t)*. The division of AQMs into BB and RB reflects the historical development of AQMs, from being enhancements to the packet dropping behaviour of the tail-drop queue, to the current control systems approach for controlling the aggregate arrival rate *X(t)*. Fig. 3.1 emphasises this relationship between BB and RB AQMs.

Fig 3.1 RB and BB AQM



The backlog process is effectively an integrator of the excess packet arrival rate *X(t)* destined for a link of capacity *C(t)*:

$$B(t+1)=[B(t) + X(t) - C(t)]^{+} \qquad (3.1)$$

The RED-like and tail-drop algorithms can be approximated by *P(t)=F(B(t));* where *F(x)* is a positive and increasing function for $x > 0$ and *F(0)=0*. For RED *F(x)*

is a piece-wise linear function, and for tail-drop it is a threshold function. The structure of the RED and tail-drop BB AQMs results in a coupling between the congestion signal $P(t)$ and the backlog:

$$P(t)=F(B(t)) \tag{3.2}$$

By (3.1), BB AQMS are making an indirect measurement of the arrival process $X(t)$. On, the other hand, RB AQMs make a direct measurement of $X(t)$. The difference is central, and limits the performance of BB AQMs.

The backlog process $B(t+1)=[B(t) + X(t) - C(t)]^+$ cannot observe long term arrival rates $X(t) < C(t)$ as if $X(t) < C(t)$ for a sufficient period of time, then $B(t)$ reaches zero. Once $B(t)=0$, and if $X(t)$ continues to be less than $C(t)$ and $B(t)$ remains zero, we can say nothing about how close $X(t)$ is to $C(t)$ by observing the state of $B(t)$. Therefore, by observing only $B(t)$, the BB AQM cannot make any measurement of $X(t)$ when $X(t) < C(t)$ and $B(t)=0$. Since in this case, the BB AQM cannot measure $X(t)$, it cannot control $X(t)$. This means that BB AQMs are unable to control the source rates, such that in equilibrium $X(t)$ is below the capacity $C(t)$, without inducing queuing in the network. BB AQMs induce queuing in the network, since they do not react to congestion unless $B(t)$ is already positive, because they cannot measure $X(t)$ and detect congestion until $B(t)$ becomes positive. Experimental results later will confirm this conclusion.

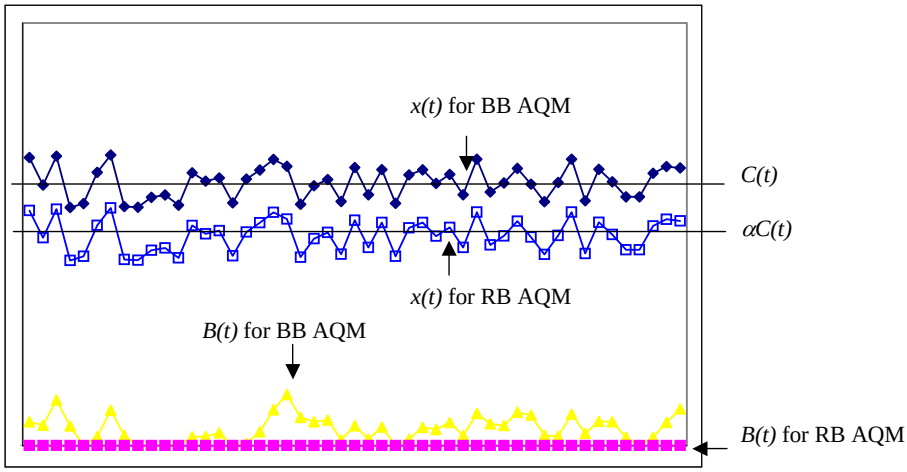
RB AQMs measure $X(t)$ directly. The basic structure of a RB AQM controller is like (3.1), however, the direct measurement of $X(t)$ makes it possible to control the gain of the feedback α, as well as the target utilisation $0 < u < 1$:

$$P(t+1) = P(t) + \alpha\times(X(t) - u\times C(t)) \tag{3.3}$$

Note that in steady state, when $P(t+1) = P(t)$, $X(t) = u\times C(t)$. Therefore RB AQM can achieve $X(t) < C(t)$ in equilibrium. Fig. 3.2 illustrates an example of the equilibrium properties of BB and RB AQM with an arrival process which varies around an equilibrium point. Note that since RB AQM controls the aggregate arrival

rate *X(t)* about the equilibrium *u*×*C(t)*, this results in no backlog. With BB AQM, *X(t)* is controlled about the equilibrium point *C(t)*, which results in queuing, as *X(t)* necessarily exceeds *C(t)* to generate positive feedback.

Fig. 3.2 Queuing and Target Capacity of BB AQM & RB AQM



In practice, RB schemes exhibit some queuing, because the equilibrium point is made close to the actual capacity and the aggregate traffic is bursty.

Another fundamental limitation of BB AQMs is that the structure of (3.2) couples congestion to delay. Note that as the load[1], in other words, the number of TCP connections present, increases, the feedback signal *P(t)* must also be increased so that the aggregate arrival rate *X(t)* remains constant. However, because *P(t)=F(B(t))*, in order to increase *P(t)*, *B(t)* must also increase. This means backlog and delay in the network for both the RED and tail-drop algorithms increase during congestion periods. As noted in [76] this coupling means that congestion is linked with bad-performance, since the delay in the network increases. This coupling is not necessary, and RB AQMs do not exhibit this problem.

---

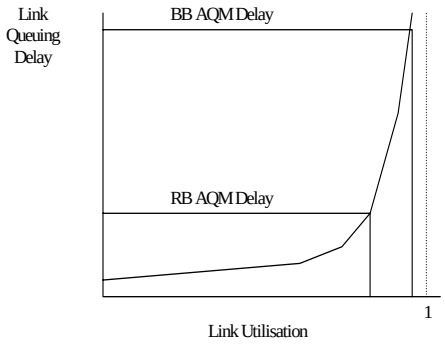
[1] When referring to Load in context of TCP connections we mean the number of TCP connections present. In this thesis, TCP connection are assumed saturated, that is, during the connection time, there is always data available for transmission. When referring to Load, in the context of offered traffic, we mean the amount of data generated for transmission.

In Summary, we see the RB AQM control structure to be the model for future AQM development because (a) it is the most general AQM model as the measurement of both *B(t)* and *X(t)*,*X*(t)' etc. does not restrict the design of the AQM (b) it is capable of eliminating queuing delays in the network since $X(t) < C(t)$ in equilibrium (c) they do not produce a coupling between network load and delay (d) they are able to produce a controlled utilisation of the link, as $X(t) = u \times C(t)$. Therefore the work in this chapter and the following chapter will focus on developing RB AQM design and application.

3.1.2 RB AQM Multi-Service Implications

The ability of the RB AQM to target an arbitrary level of utilisation allows the packet delays inside the network to be greatly reduced at a minimum sacrifice of utilisation. From (3.1), we can see that BB AQMs inherently target a utilisation of 1, since in equilibrium when $B(t) = B(t+1)$, $X(t) = C(t)$. If we assume the queuing process at the link to be M/M/1, basic queuing theory [68] tells us that at high utilisation, as the utilisation of the link is increased, the delay increases dramatically, as depicted in Fig. 3.3. By driving the link at a slightly lower utilisation (say, 98% instead of 99%), RB AQM can dramatically decrease the queuing at the link, as depicted in Fig 3.3.

Fig. 3.3 Delay vs Utilisation for Random Process



Of course, the total delay for a packet also includes the delay at the source before the packet is transmitted. RB AQMs forces sources to buffer their data for

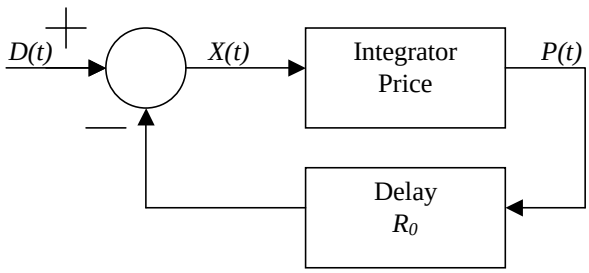
transmission at the source, and therefore shifts the bulk of the delay from being inside the network, to inside the source. The delay for bulk data transfer applications remains largely unchanged. However, the key advantage of RB AQM control is that all packets passing through the network experience low queuing delay once inside the network. Therefore interactive applications can receive low packet delay. This allows multi-service applications to coexist on the same best-effort network.

3.2 RB AQM Control Stability & Scalability

In this section, we will provide an overview of the key stability properties of a RB AQM flow control and how they are affected by the size of the network. This section reviews the results of [100], which suggest how the parameters of the RB AQM and source algorithm should be scaled, so that the congestion control loop remains stable in face of different RTTs, different number of sources and a variable number of links on the end-to-end path.

For the purposes of highlighting stability issues of flow control, we consider only the essential characteristic of the flow control loop; the integrator action of the RB AQM controller. As such the model in this section and in [100] simplifies the congestion control feedback to the loop presented in Fig. 3.4. The AQM produces a congestion signal $P(t)$ that regulates the source transmission rate. In this model, source control is simplified to a linear control. Such that a source with demand $D(t)$ bps, transmits at a controlled rate $X(t) = D(t) - P(t)$ bps.

Fig. 3.4 Flow Control Closed-Loop System



We will expand the analysis in [100] which uses a small signal approach to model the system. The aggregate arrival rate at the link is $X(t) = X_0 + \delta X(t)$, where X_0 is the equilibrium arrival rate and $\delta X(t)$ is the small signal deviation. Likewise, the marking rate is $P(t) = P_0 + \delta P(t)$, where P_0 is the equilibrium marking rate and $\delta P(t)$ is the small signal deviation.

In [100], it is argued that the simplest control law which can meet the objective of controlling the arrival rate $X(t)$ to match the target capacity C, is an integrator that sums the excess flow into the congestion control price (3.3). We can rewrite the basic RB AQM integrator control law of (3.3), using the small-signal fluid-flow model as:

$$\dot{P} = K \cdot \delta X(t) \tag{3.4}$$

We can now analyse closed loop stability of the system by working out the poles of the system. From (3.4), we can find the s-transform of the controller:

$$P = \int_{-\infty}^{t} K \cdot \delta X(t) \cdot dt \tag{3.5}$$

$$P(s) = \frac{1}{s} \cdot K \cdot X(s) \tag{3.6}$$

We can express the feedback signal, delayed by one RTT t_d, $P(t - t_d)$ as:

$$e^{-s \cdot t_d} \cdot P(s) \tag{3.7}$$

Therefore the closed-loop system transfer function becomes:

$$T(s) = \frac{\frac{1}{s} \cdot K}{1 + \frac{e^{-s \cdot t_d}}{s} \cdot K} \tag{3.8}$$

The characteristic equation of this system is $s + e^{-s \cdot t_d} K = 0$. Following the analysis of [81] on p. 20, we write $s = \sigma + i\omega$, so that:

$$\sigma + Ke^{-\sigma t_d} \cos(\omega t_d) = 0 \tag{3.9}$$

$$\omega - Ke^{-\sigma t_d} \sin(\omega t_d) = 0 \tag{3.10}$$

This system has an infinite number of roots. When the system is stable all of these poles are in the left hand plane. As the parameters K and t_d are increased, at some point, the right-most pole will travel from the left hand plane into the right-hand plane, making the system unstable. At the point at which this pole is on the imaginary axis, when $\sigma = 0$, $K\cos(\omega t_d) = 0$ and $K\sin(\omega t_d) = \omega$, so $\omega = \pm K$ and $t_d = \frac{\pi}{2K}$. Thus the system is stable when:

$$K \cdot t_d < \frac{\pi}{2} \tag{3.11}$$

Note that as the delay or gain is increased, the system can become unstable. For stable congestion-control, it is important for the stability of this feedback system to be invariant to delay, and the gains that can feasibly occur in the loop, due to different network topologies.

To make the stability of the system invariant to different propagation delays t_d, [100] introduces a gain $1/t_d$ in the loop. This gain is placed in the source, where the RTT of the connection can be measured. When K incorporates this scaling, we can see that the dependence of t_d on pole positions is removed in (3.11).

The loop gain K also is affected by two factors, 1) the number of links on the end-to-end path of the communication 2) the number of sources transmitting over the link. The stability of the system also needs to be made invariant to these parameters of the network.

Fig. 3.5 shows the feedback loop when there are M bottleneck links on the end-to-end path of the transmission. Each link adds to the total congestion signal communicate to the source. Therefore, an increase in the source transmission rate δX, will elicit an increase in the congestion signal $M \times \delta P$. This translates into a loop gain

M with *M* bottleneck links on the end-to-end path. To make the stability of the system invariant to the number of bottleneck links, [100] introduces a gain 1/*M* at the source. Measuring *M* is not trivial, and in [100] it is suggested that either an upper-bound is used, or additional bits in the packet and router infrastructure is introduced to measure this information. In Chapter 6, we introduce a new congestion control architecture which does not require the measurement of *M*.

Fig. 3.5 Multiple Bottleneck Link Path

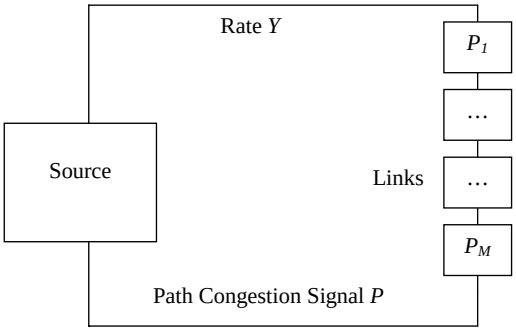
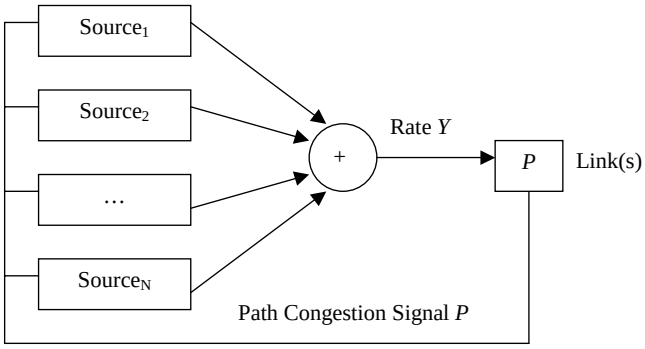


Fig. 3.6 shows the feedback loop when there are *N* sources transmitting over the same link. Each source adds to the aggregate arrival rate *Y* at the link. Therefore, a decrease in the congestion signal δP, will elicit an increase in the aggregate transmission rate $N \times \delta Y$. This translates into a loop gain *N* when there are *N* sources transmitting over the link. To make the stability of the system invariant to the number of sources, [100] introduces a loop gain *1/N*. Unfortunately, on a stateless Internet network, neither the link, nor the sources have the information required to detect the number of sources sharing a link. However, [100] notes that if the link has a *1/C* gain and each source has a gain X_0 (its transmission rate), the 1/*N* gain is achieved in a distributed manner. (Since $X_0/C = 1/N$ if we assume each source shares link capacity equally).

Fig. 3.6 Multiple Sources Transmitting on Path



Let us split the complete loop gain K into a gain at the source K_S and a gain at the link K_L, $K=K_L.K_S$. The complete source gain, K_S, required to make the system invariant to RTT, the number of bottleneck links, as well as the number of sources is:

$$K_S = \frac{\alpha_s \cdot x_0}{M_i \cdot t_d} \tag{3.12}$$

where $0 < \alpha_s < 1$ is a source gain constant. In this chapter, we are particularly interested in the design of AQM algorithms, and the only scaling parameter required at the link is:

$$K_L = \frac{\alpha_l}{c_l} \tag{3.13}$$

where $0 < \alpha_l < 1$ is a link gain constant.

As an aside, the link between loop delay and stability explains why the mark-front strategy introduced in [75], results in better performance. In [75], the strategy where packets are ECN marked at the front of the queue instead of the back is shown empirically to decrease backlog and improve control. This strategy effectively decreases the control loop delay t_d by removing the delay due to queuing at that link from the marking feedback path.

3.3 GREEN AQM

In this section, we introduce a new RB AQM algorithm called GREEN. GREEN is an extension of the basic integrator AQM control law (3.3). In later

sections, we will show by simulation that GREEN is able to outperform tail-drop, RED and performs at least as well as REM PC3 despite requiring less tuning parameters.

3.3.1 GREEN Architecture

The GREEN algorithm is described by the following control law:

$$P(t+\Delta T) = P(t) + \varphi(t) \cdot U(\delta(t))$$

where

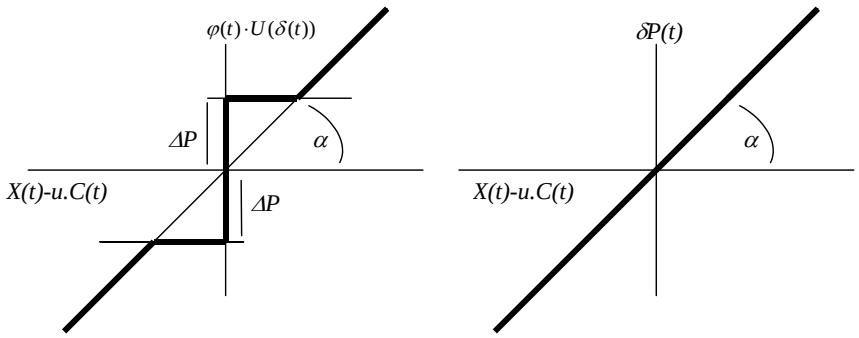
$$\delta(t) = \alpha \cdot (X(t) - u \cdot C(t))$$

$$U(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

$$\varphi(t) = \max(abs(\delta(t)), \Delta P)$$

where $X(t)$ is the link's estimated arrival rate (bps), $C(t)$ is the link capacity, u is the target utilisation, α is the control gain, P(t) is the marking/dropping rate, ΔP determines the minimum adjustment, and $1/\Delta T$ is the update rate. To visualise the GREEN algorithm, the amount of adjustment to $P(t)$, $\delta P(t)$ for every ΔT seconds, as a function of $(X(t) - u \cdot C(t))$ is shown for GREEN and a linear integrator control law in Fig. 3.7.

Fig. 3.7 (left) GREEN *P(t)* Adjustment (right) Linear Integrator *P(t)* Adjustment

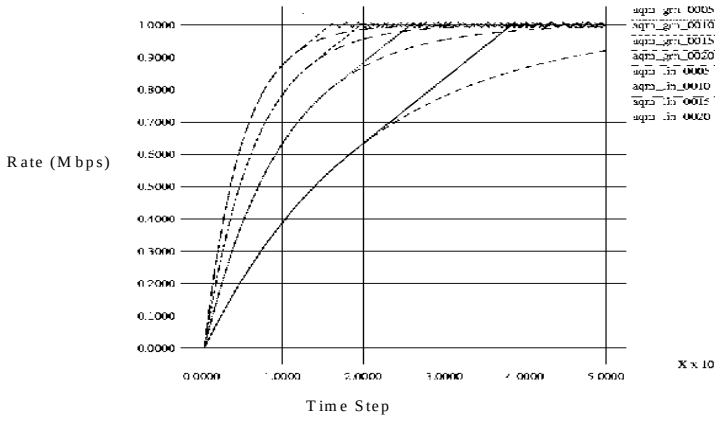


The target link utilisation u is assigned a value less than 1 to clear the link buffer in equilibrium. The control gain α and the minimum adjustment ΔP should be

scaled in proportion to 1/*C*, the expected link capacity, to make stability invariant to link capacity as discussed in Section 3.2. The update interval ΔT should be an order less than the shortest RTT in the network.

It is known that the convergence rate of the integrator controller (3.3) is not optimal [7]. It is important for the arrival rate to a link to be controlled towards the target capacity quickly, since long transient periods, where arrival rate is far below capacity, create under-utilisation and periods where arrival rate is above capacity create queuing and delay. GREEN improves the integral control law by a first-order method of limiting the minimum adjustment of the marking probability *P(t)* to ΔP per update interval. The proposal in [7] suggest another method of accelerating the integrator. However, the method in [7] requires the computation of a scaling matrix which involves division, an operation not readily amenable to FPGA or ASIC implementation with limited space and time resources.

Fig. 3.8. GREEN Transient Response.



To determine how GREEN's design affects control, Fig. 3.8 shows a step response of the GREEN AQM as well as the linear integrator. The step response is taken for the system in Fig 3.4. Fig 3.8 shows the aggregate arrival rate as it is controlled from the initial condition of 0 towards the link capacity *C(t)*=1 Mbps. The parameters for the experiment are delay = 40 ΔT, ΔP=*0.001* and α=(0.0005, 0.0010, 0.0015, 0.0020, 0.0025) for both GREEN and the linear law. Note that Fig. 3.8 shows that GREEN is able to control the arrival rate towards the target capacity

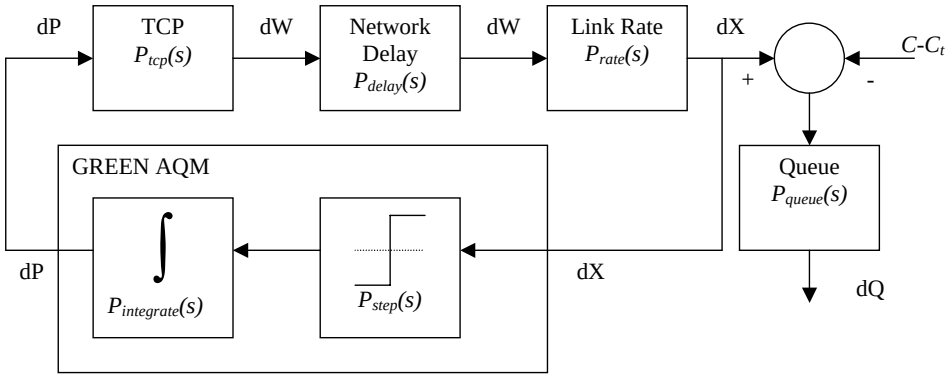
quicker than the linear control law, for the same 'large-signal' control gain α. Note that as $\Delta P \rightarrow 0$, GREEN has the same stability properties as the linear control law. As ΔP grows, the speedup achieved is at the cost of creating a small limit cycle about the target capacity. This limit-cycle effect will be analysed in detail in the following section.

3.3.2 GREEN Control Analysis

In this section, we give an overview of the control theoretic analysis of the Rate-Based AQM GREEN. We are interested in two important several results 1) whether the system is stable 2) affect of parameter settings. The work presented in this sub-section is a summary of the results from the manuscript [69] Hongwei Kong *et al.*, who perform a control theoretic analysis of our GREEN AQM. We give an overview of their analysis to better understand how RB AQM, and in particular GREEN, work.

The analysis is of the system model shown in Fig. 3.9. There are *N* TCP sessions feeding a single GREEN bottleneck link. Each session has the same RTT R_0. The system consist of the congestion feedback loop where GREEN AQM algorithm generates a marking signal *P(t)* which controls the *N* TCP sessions that transmit at an aggregate rate *X(t)*.

Fig. 3.9 GREEN System Model



The dynamics of GREEN are analysed using a small-signal model around an equilibrium point. In this small signal model, it is assumed $\alpha = 0$ and $\Delta P > 0$, which

makes the GREEN AQM only have a step response, with no linear component. The window size of the TCP sessions is $W(t) = W_0 + \delta W(t)$, where W_0 is the equilibrium windows size and $\delta W(t)$ is the small signal deviation. The aggregate arrival rate at the link is $X(t) = X_0 + \delta X(t)$, where X_0 is the equilibrium arrival rate and $\delta X(t)$ is the small signal deviation. Likewise, the marking rate is $P(t) = P_0 + \delta P(t)$, where P_0 is the equilibrium marking rate and $\delta P(t)$ is the small signal deviation.

In [70] TCP window control is modelled using the formulation developed by [88]. Kong *et al.* uses the same linearization as made by [53]. This linearized TCP window adjustment can be represented by the following transfer function:

$$P_{tcp}(s) = \frac{\frac{R_0 C^2}{2N^2}}{S + \frac{2N}{R_0^2 C}}.$$

The function $P_{tcp}(s)$ relates how an adjustment of the marking rate $\delta P(t)$, on a link with N sessions, of capacity C and with RTT R_0, produces a change $\delta W(t)$ in the window of a single TCP session.

The changes to the windows size of each TCP session produce a change in the aggregate arrival rate at the link. The aggregate rate of arrival at the link is the number of TCP sessions N times the transmission rate of each session. The transmission rate of a TCP session is its window size over the RTT. Then it follows that the small signal deviation of transmission rate $\delta X(t)$ is related to the small signal windows size deviation $\delta W(t)$ as follows:

$$\delta X(t) = \frac{N}{R_0} \cdot \delta W(t).$$

Therefore the transfer function which describes the aggregating of individual TCP window changes to a single aggregate rate change at the link is:

$$P_{rate}(s) = \frac{N}{R_0}.$$

Note that in the RTT delay is modelled as a singled lumped delay of R_0. The delay transfer function is:

$$P_{delay}(s) = e^{-s \cdot R_0}.$$

The GREEN AQM can be represented as a transfer function with the small-signal aggregate packet arrival rate $\delta X(t)$ as the input and a small-signal marking rate $\delta P(t)$ as the output. The essential elements of GREEN: 1) the marking rate integrator $\frac{1}{s}$, and 2) ΔP step increase or decrease at rate $\frac{1}{\Delta T}$ can be represented as a non-linear transfer function:

$$\delta P_{GREEN}(s) = \frac{\Delta P}{\Delta T}\frac{1}{s}U(\delta X(s)).$$

The complete loop function $L(s)$ of the system is:

$$L(s) = P_{GREEN}(s) \cdot P_{Rate}(s) \cdot P_{TCP}(s) \cdot P_{Delay}(s). \quad (3.14)$$

3.3.3 Analytical Results

Given the above system description, Kong *et al.* apply the Nyquist stability criterion to the system to determine stability properties.

Since the GREEN AQM has a minium adjustment ΔP at every time step ΔT, intuitively it is expected that the system never settles to a stable point. One would intuitively expect a small limit cycle around a stable point. Indeed, the analysis performed in [70] of the above model yields the important result that there exists a unique stable oscillation point, which means that the system has a stable limit cycle of operation. The limit cycle can be made arbitrarily small by changing ΔP and ΔT at

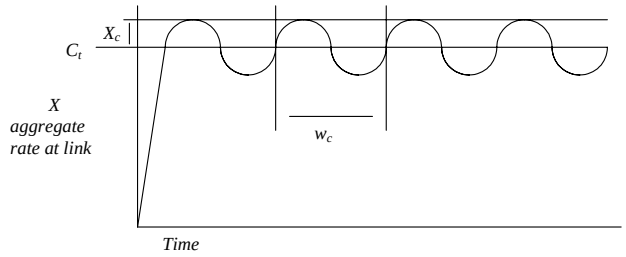
the cost of convergence rate. Kong *et al.* formulate the frequency w_c and amplitude X_c of this limit cycle oscillation:

$$R_0 w_c + \tan^{-1}\left(\frac{w_c}{2N / R_0^2 C}\right) + \frac{\pi}{2} = \pi, w_c \in (0, \frac{\pi}{2R_0}).$$

$$X_c = -\frac{4}{\pi} G(jw_c) = \frac{\frac{R_0^2 C^3}{N^2}\frac{\Delta P}{\Delta T}}{\pi w_c \sqrt{(\frac{w_c R_0^2 C}{2N})^2 + 1}}. \qquad (3.15)$$

This limit cycle is illustrated in Fig. 3.10.

Fig.3.10 GREEN AQM Limit Cycle



Kong *et al.* make the further observation that the efficiency of GREEN is limited by the amplitude of this limit cycle. This is because, to eliminate queuing, the arrival rate *X(t)* must always be below the link capacity. Therefore the target capacity C_t must be set such that $C_t + X_c < C$. This limits the maximum target capacity for a particular RTT R_0, link capacity and ΔP, ΔT settings. Indeed, from (3.15), the faster the system adjusts the marking rate with a higher ΔP or smaller ΔT, the lower is the possible utilisation C_t/C.

The analysis by Kong *et al.* describes the upper bound on the amplitude of the limit cycle, because all *N* sources are assumed to be at the same RTT. In a real network, sources have different and random RTT, which desynchronises the cycle, and reduce its magnitude. Indeed, in the simulations below, there is a diversity of RTTs, and the limit cycle amplitude was not significant.

The key result of the analysis of Kong *et al.* is that GREEN AQM is able to operate in a stable regime. We will now simulate the GREEN AQM to compare its performance to other AQMs.

3.4 Simulation AQM comparison

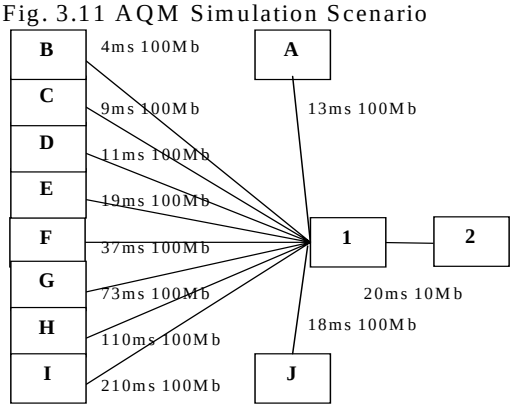
3.4.1 Simulation Introduction

In this section, we will compare the performance of a number of well-known BB and RB AQMs with each other and GREEN. We will perform a number of simulations of different scenarios with the GREEN, REM, RED and tail-drop AQMs using the NS-2 simulator [94]. The simulations will serve to confirm some of the earlier analysis of the steady state properties of BB and RB AQMs. We are particularly interested in the equilibrium backlog properties, utilisation and packet loss rates for BB and RB AQMs under different traffic loads, propagation delays and buffer sizes.

Fig. 3.11 shows the common network topology used for all trials described here. The network has source nodes A-J and a single destination, Node 2. The AQM algorithm is placed in Node 1, which represents the ingress of a router and governs the Node 1 to Node 2 link. Note that multiple TCP connections can originate from each source A to J.

Each trial lasted 120 seconds with measurements being taken after a 20 second warm up period. The active TCP connections were evenly distributed among the A-J sources. Connections started at a uniformly distributed time over the test period with a uniformly distributed duration between 3 and 23 seconds. The packet size was 1000 bytes.

Fig. 3.11 AQM Simulation Scenario



The configuration parameters for each of the AQMs followed recommended settings: RED : min_{th} = 20% and max_{th} = 80% of the queue size. REM: (Parameters adapted from 'Interworking with Reno' [6]) $\gamma = 0.01$ $\phi = 1.003$ $\alpha = 0.1$ updatetime = 0.01 bo = 55. (Public domain *rem-mark.cc* [8] was used). GREEN: ΔT = 10ms, ΔP = 0.001, C_l = 0.97% of Capacity, K = 0.1. The parameter α was set to zero so that the control law is reduced to a simple step function. All algorithms were configured to use ECN, so that all packet loss measured was only due to overflow.

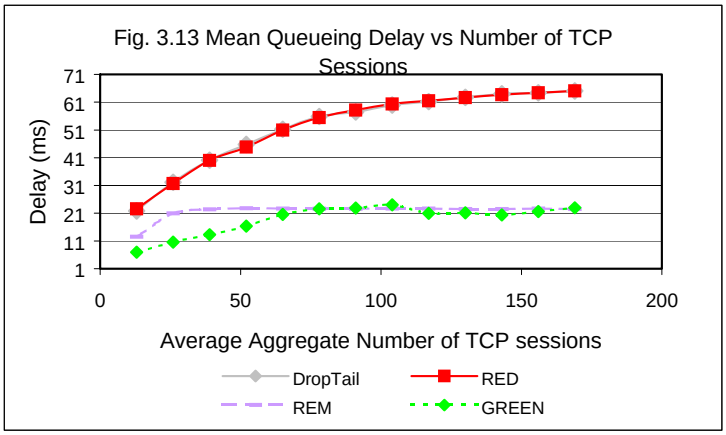
3.4.2 Trial 1: AQMs under different Load

The AQMs were tested under different traffic loads, by varying the number of active saturated TCP sessions. The total number of TCP sessions activated in each 120s simulation period ranged from 100 to 1300 (mean of 11 to 141 concurrent sessions). The buffer size was 91 packets (72.8ms), so that it is lower than the bandwidth delay product of some connections. The link utilisation, (i.e., percentage of time the 1-2 link is not idle), packet loss rate (at 1-2 link due to buffer overflow), and queuing delay at the link were measured. Fig. 3.12 summarizes the mean of utilisation, loss and delay over the 13 simulation periods for each AQM. Fig 3.13 shows the packet delay performance and Fig. 3.14 shows the packet loss performance across the range of loads for all AQMs.

Fig 3.12 Trial 1 Result Summary – Mean Utilisation, Loss, Delay for AQMs

	Tail-drop	RED	REM	GREEN

Util %	0.98	0.98	0.96	0.97
Loss %	6.85	5.43	0.23	0.05
Delay(ms)	52.45	52.41	21.68	18.79



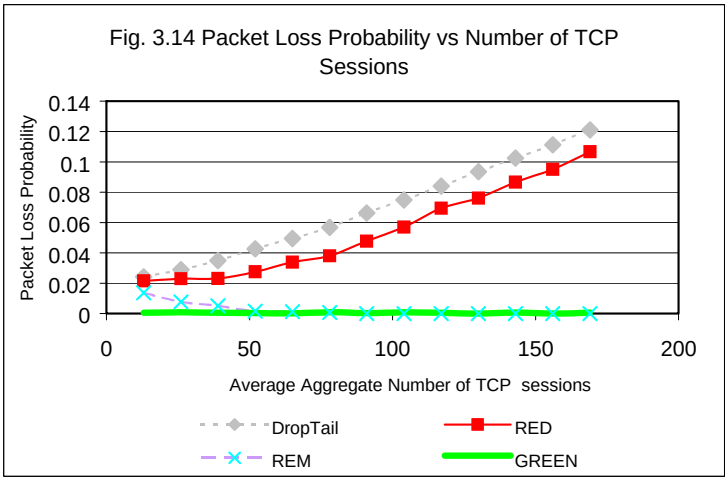


Fig. 3.13 shows that the RB AQMs REM and GREEN both exhibited low queuing delay across all loads. This resulted in almost no packet loss as shown in Fig. 3.14. Notice from Fig. 3.12, that the 2 to 3 times lower queuing delay of RB AQMs as compared to the BB AQM occurs with only a 1-2% decrease in utilisation. This is consistent with basic queuing theory results for queuing with random arrival processes [68], and our discussion in Section 3.1.2.

For the BB AQMs, RED and tail-drop, the delay (backlog) increased with the load in agreement with our discussion in Section 3.1.1. The delay necessarily increases with the load because of the coupled relationship between backlog *B(t)* and congestion notification rate *P(t)*, *P(t)=f(B(t))*. This also explains the increased loss rate with load, since an increasing mean queue size increases the probability of loss. Because of the coupling between *P(t)* and *B(t)*, the BB AQMs RED and tail-drop cannot target an arbitrary utilisation level, since *B(t)* must be positive and increasing with load. As the load increases, these AQMs inevitably drive the system at higher utilisation and result in higher backlog than RB AQMs as shown in Fig. 3.13.

RED has a slightly lower loss rate than tail-drop, because it is able to mark some packets using ECN, rather than having the entire congestion notification signal being packet drops.

3.4.3 Trial 2: AQMs with different buffer sizes

In this experiment the maximum buffer size was varied to determine the relationship between the maximum buffer size and the equilibrium backlog performance of BB and RB AQM algorithms. 1000 TCP sessions over 120 seconds were initiated for each queue size. Utilisation, packet loss and delay were measured. For brevity, only the delay and loss performance is shown in Fig. 3.15 and Fig. 3.16. The maximum buffer size is measured in ms, which is the number of bits backlog over the link capacity.

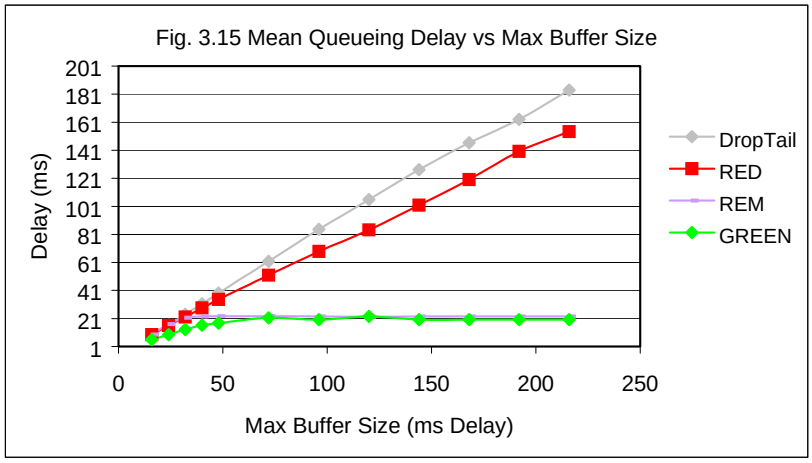


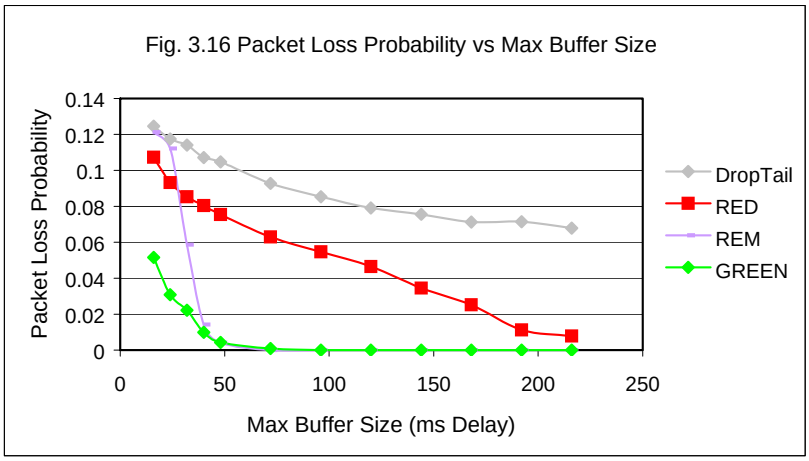
Fig. 3.15 shows that the mean backlog for RED and tail-drop increases with the maximum queue size. This is because for both the tail-drop and RED AQMs, the congestion notification rate is related to the proportion of the queue which is filled with backlog as discussed in 3.11.

For the tail-drop AQM, there is no congestion notification until the queue is already full. Hence, regardless of the maximum queue size, TCP needs to overflow the queue before the sources are controlled by sensing dropped packets. So the tail-drop queue will always operate around the point of being full. This explains the long RTT times experienced on the Internet, where most queues are tail-drop, and TCP traffic fills them with backlog before any congestion control occurs.

For RED AQM notice that the mean delay increases with the maximum queue size. As discussed in 3.1.1, with RED the congestion notification rate is roughly proportional to the percentage of the backlog in the queue. Therefore as the maximum queue size increases, the equilibrium level of backlog which generates the same congestion notification rate also increases.

For the RB AQMs, REM and GREEN, the mean backlog/delay remained level around 20 ms across the range of maximum queue sizes. For RB AQMs where the congestion notification rate is driven by an integrator process, the notification rate is adjusted only with the objective of controlling the arrival rate towards the target capacity. Therefore, there is no direct coupling between the congestion notification rate generated, or the backlog present in the queue. The backlog which does occur, is only what is necessary to achieve the target utilisation u. Since the sources were not changed across the range of maximum queue sizes, the arrival process remains similar, and therefore the backlog that is necessary for the same target utilisation is also fixed.

The uncoupling of congestion notification rate and backlog size in RB AQMs means it is possible to have a large buffer, to reduce packet loss during traffic bursts, without the drawback of high mean delay which results when using the RED-like or tail-drop BB AQMs with large buffers.



We will now describe the packet loss results for the RB AQMs. Normally, the buffer size should be provisioned to be able to absorb the bursty nature of TCP traffic without significant packet loss. In this experiment, the buffer sizes was varied from one large enough to exhibited little loss, to a small buffer that exhibited high loss.

As indicated by Fig. 3.16, there is a high packet loss rate for all AQMs when the buffer has less than 50 ms of burst capacity. Fig. 3.16 shows that both RED and tail-drop exhibit higher loss than the RB AQM algorithms. This is due to the higher mean backlog size for both of these algorithms.

As shown in Fig. 3.16 GREEN has a lower packet loss rate than REM with low buffer sizes. In this trial $\alpha = 0$ for GREEN, and the price is only adjusted by ΔP at each interval. The linear term $(X(t)-C(t))$ in the price adjustment makes REM more aggressive to changes in traffic. At low buffer sizes, when significant overflow drops occur, the arrival rate at the link will drop sharply, as TCP sessions react to loss. With REM, $P(t)$ is decreased sharply when X(t) drops after an overflow event, because of the $(X(t)-C(t))$ term. However, a drop in $P(t)$ invites sources to transmit more and can perpetuate further overflows. Because GREEN can be configured with $\alpha = 0$, eliminating the linear $(X(t)-C(t))$ term, GREEN can ignore the magnitude of $(X(t)-C(t))$ and filter out large bursts in traffic, without decreasing the small signal gain ΔP. AQMs such as REM do not have a separate large signal and small signal gain. Other experiments in our study [123] also confirm that being able to filter out

large bursts, without sacrificing the ΔP gain, improves performance for cases with low flow aggregation, where the traffic is also quite bursty.

3.4.4 Conclusion

In this section we introduced the AQM algorithm GREEN. We characterised its performance by analytical and simulation techniques. In subsequent sections we will develop techniques for deploying and extending the applications of RB AQMs. We will use GREEN in demonstrating and simulating these techniques, however it is important to stress that they are applicable to any other RB AQM with a structure based on (3.3).

3.5 RB AQM Deployment with non-ECN sources

3.5.1 Introduction

In the previous part of this chapter, we have introduced different AQM algorithms and shown the advantages of RB AQM control. These controllers are able to control the source rates such that the mean level of backlog is very low and reduce the network's latency closer to the propagation delay. However, in the analysis and experiments, we have assumed that congestion notifications are ECN marks. This assumption has also been made in other works proposing RB AQM algorithms [6] [76] and ignores a significant deployment issue.

In this section, we uncovers a significant issue that a non-ECN low latency TCP/IP network with RB AQM faces and propose a solution. The analysis is applicable to any RB AQMs derived from the basic integrator structure (3.3), such as REM or GREEN. We call this problem the "low latency efficiency collapse".

With ECN still not widely deployed, the main congestion notification remains packet dropping. By reducing the RTT to near the propagation delay, TCP sessions become very aggressive and the packet dropping rate required for congestion notification becomes prohibitively high. In this section, a solution to this problem is introduced. We show that ECN and non-ECN IP packets should be queued in separate queues with independent AQMs for optimal delay and loss performance.

In the following sections, we explain the problem of low latency efficiency collapse analytically and provide experimental results on a scenario involving small latency to demonstrate the efficiency collapse. Finally, we will present a solution to this issue.

3.5.2 Analysis of Low Latency Efficiency Collapse

In this section, we will present an analysis of the low latency efficiency collapse problem. We will develop a formula for the efficiency of a TCP transmission over a RB AQM network.

To analyse the behaviour of RB AQMs with TCP when packet drop congestion notification is used, we require a model of TCP. The first-order model proposed by S. Floyd [40] will be used. Firstly, we will expand on this model's derivation presented in [40] so that the origins and assumptions of the model are clarified. More sophisticated models for TCP have been proposed which include time-out and other infrequent events [96] [47], however, the first order model is sufficient to capture the efficiency collapse problem.

Consider a TCP connection with RTT R with packet size B bytes. Assume that the system is operating in steady-state conditions such that the RTT R is constant, and the transmission rate as measured over several RTTs is also constant. Let us assume that in this steady-state regime, packet drops are equally spaced, then the maximum window size of TCP, at the time of the drop is W. TCP decreases the window size by half upon packet drop, and increases the window size by 1 every RTT. Then as shown in [40], the number of packets transmitted between each drop, G, is:

$$G = \frac{W}{2} + \left(\frac{W}{2} + 1\right) + \left(\frac{W}{2} + 2\right) + \left(\frac{W}{2} + 3\right) + \ldots + W$$

If we define S_n as the sum of an arithmetic sequence:

$$S_n = \frac{n(n+1)}{2} = 1 + 2 + 3 + \ldots + n$$

Then we can evaluate *G*:

$$G = S_w - S_{\frac{W}{2}} = \frac{3W^2}{8} + \frac{3W}{4} \approx \frac{3W^2}{8}$$

Given *G*, Floyd *et al.* obtains a bound on the number of drops per packet *p* transmitted, and the TCP window size given this loss rate:

$$p \le \frac{8}{3W^2} \qquad W \le \sqrt{\frac{8}{3p}}$$

Because the peak windows size before a packet drop is *W*, the mean window size W_{mean} is at most *0.75 W*. The throughput of a TCP transfer is therefore:

$$T \le \frac{0.75 \cdot W \cdot B}{R} \quad (bps)$$

The full model of TCP as developed in [40] is:

$$T \le \frac{1.5 \cdot \sqrt{\frac{2}{3}} \cdot B}{R\sqrt{p}} \tag{3.16}$$

Let us assume that the actual TCP throughput is some constant *k* times this upper bound. Then, the approximate TCP throughput is:

$$T = \frac{k \cdot B}{R\sqrt{p}} \tag{3.17}$$

The congestion notification rate *p* as a function of the *R* (RTT) and the target throughput *T* is:

$$p = \left(\frac{k \cdot B}{T \cdot R}\right)^2 \quad 1 \ge p \ge 0 \tag{3.18}$$

By (3.18), as R is reduced, the amount of packet dropping needed to control sources increases at rapidly increasing rate. RB AQMs create a low R because they clear the network of backlog since $T = u$ x C, and $u < 1$. By (3.18), with low R, the dropping rate generated by the RB AQMs for non-ECN packets is very high.

An intuitive explanation for the low latency efficiency collapse is as follows. TCP increases its sending rate upon every positive acknowledgement. As the RTT reduces, these positive acknowledgements return at a faster rate causing TCP to increase its sending rate at a faster rate. As a result, in order for the AQM to maintain the same flow rate, with a smaller RTT, more packet drops are necessary. In other words, decreased RTT permits a faster rate of positive acknowledgements. In order to control the rate, these must be compensated by a faster rate of negative acknowledgements, which must be signalled by dropping packets for non-ECN sources. As TCP retransmits the lost data, this data loss affects efficiency. The expected number of packet transmissions X needed for a successful transmission is given by:

$$X = \frac{1}{1-p} \ . \tag{3.19}$$

By (3.19) X increases at an increasing rate as packet loss increases. The efficiency E is given by $1/X$ which, by (3.19), is

$$E = 1 - p \ . \tag{3.20}$$

By combining (3.18) and (3.20), the efficiency E as a function of the RTT R is given by

$$E = 1 - \left(\frac{k \cdot B}{T \cdot R} \right)^2 . \tag{3.21}$$

Note that the efficiency is highly sensitive to the RTT R. The fundamental result is that efficiency cannot be maintained as the latency is reduced for non-ECN

sources (congestion notification by packet dropping). This is a fundamental problem in operating AQMs that aim to reduce queuing in an environment that includes non-ECN sources.

This result gives a new reason for ECN. ECN will enable the operation of an efficient low latency TCP/IP network. Where RB AQM are deployed with traffic which is not ECN capable, special measures, which we will introduce in following sections, need to be taken.

3.5.3 Experimental Low Latency Efficiency Collapse

To study low latency efficiency collapse, an experiment was performed using a network of two computers running the LINUX operating system with TCP/SACK and no ECN as shown in Fig. 3.19 and in Fig. 3.20 in more detail. The AQM used was GREEN [123], which is representative of other RB AQMs because it has the property that the equilibrium packet dropping rate occurs when $X(t) = u \times C(t)$.

Fig. 3.19 LINUX AQM Test-Bed Experiment Setup

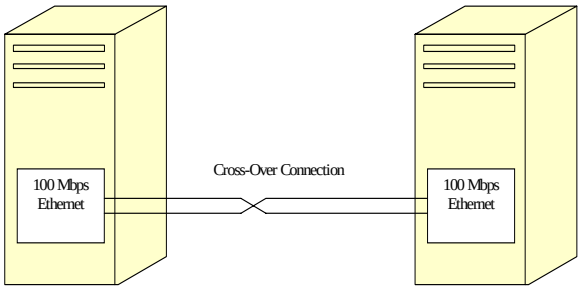
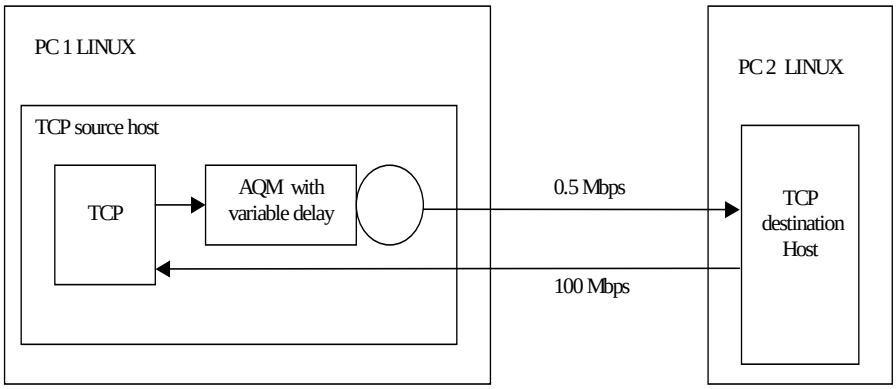


Fig. 3.20 AQM and Delay Test Network



On the source host the Kernel (version 2.4.3-20) clock was increased from the default value of 100Hz to 1000Hz to improve measurement accuracy. To emulate a range of RTT values, packets entering the AQM queue are time stamped, and not released until the delay time *D* has expired. Hence,

$$\text{RTT} = D + (\text{propagation} + \text{queuing} + \text{processing delays}) \qquad (3.22)$$

The actual RTT was measured using the PING program executed at one-second intervals with over a hundred samples for each value of *D*. Fig. 3.21 shows the measured relationship between *D* and RTT. Because the AQM queue service rate is the bottleneck in the system, *D* is a good estimator of the actual RTT as evidenced by Fig 3.21. However, at small values of *D*, RTT does not go below 10 ms due to the scheduling latencies of both computers.

Fig. 3.22 shows how the measured loss rate increases drastically as RTT is decreased to 0. Operating with no enforced delay creates a congestion notification drop rate of 0.175 which results in about 79% throughput. This is a significant 21% loss of capacity. In reality, the situation is even more grave, as a drop rate of 0.175 results in frequent TCP timeouts which pause transmission altogether. Indeed, with such a high drop rate, it was found that additional TCP sessions would sometimes not

even start, as the handshaking packets were lost. Therefore, widespread deployment of low latency AQMs results in efficiency collapse for non-ECN sources.

Fig. 3.21 Round Trip Time (RTT) Vs AQM Induced Delay D

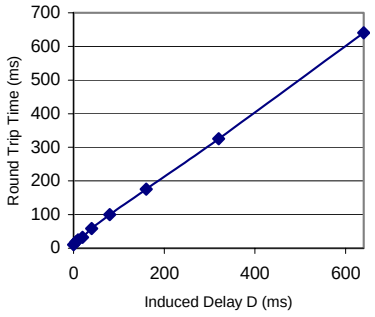
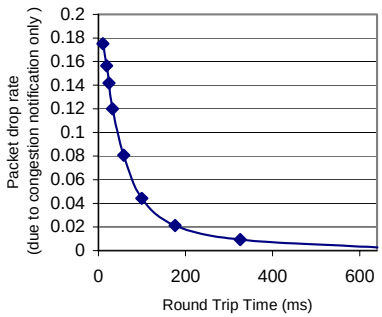


Fig. 3.22 Congestion Notification Packet Dropping Rate Vs RTT



In literature [76, 77, 100], congestion control analysis has been performed on the basis that the RTT is a constant. This assumption has been made as for future networks queuing delay will decrease due to statistical multiplexing and higher capacities, as described in Section 2.7.2. Unfortunately this ignores the low latency efficiency collapse. We have demonstrated that it is not enough to only considered the packet-marking rate or dropping rate as a signal of congestion to TCP. In fact, the complete congestion signal that affects TCP also includes the RTT *R*. That is, the complete congestion feedback signal from the network that determines the transmission rate *T* in (3.17) includes RTT *R* as well as the congestion notification rate *p:*

$$f(R, p) = R\sqrt{p} \tag{3.23}$$

3.5.4 Solution to Efficiency Collapse Problem

In previous sections, we discussed the relationship between RTT and the required packet dropping for non-ECN sources and showed that an increased RTT results in a lower packet-dropping rate. Therefore, one solution to reducing the packet dropping is to induce a fixed packet delay to increase the RTT. Since by (3.22), the minimum RTT time is bounded by the induced delay *D*, by (3.18) the loss

is bounded to some desired maximum loss $l_{d.}$. In this case, the D value required to satisfy the maximum bound of l_d, for a TCP flow with throughput T bps can be obtained from (3.17) by:

$$D = \frac{k \cdot B}{T\sqrt{l_d}} \quad . \tag{3.24}$$

Since the source and destination are not known in advance, it is not possible to distribute this enforced delay across the links of the network so that each source and destination receives the minimum possible delay. In this case, a conservative policy is to enforce the minimum delay at each router. As the minimum delay necessary for reasonable loss figures is in the order of 30 ms for the tested case, this is a good compromise.

Ironically, the tail-drop and RED do not have to introduce the induced latency D because of the large backlog they operate with, (as demonstrated in the experiments). This backlog ensures that the minimal required latency is satisfied. The RB AQMs can deliver lower latency then a path with tail-drop or RED as only the minimum delay needs to be induced. The enforced delay D at each queue can be implemented by setting a time-stamp (of value *stamp*) on the arriving packet with *stamp* = current time, and not dequeuing the packet until the current time passes D + *stamp*, as shown in Fig 3.23.

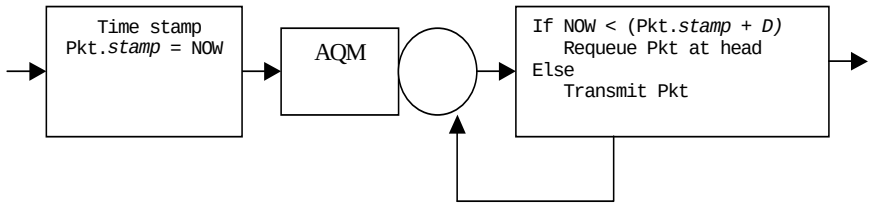


Fig. 3.23 AQM with Controlled Delay

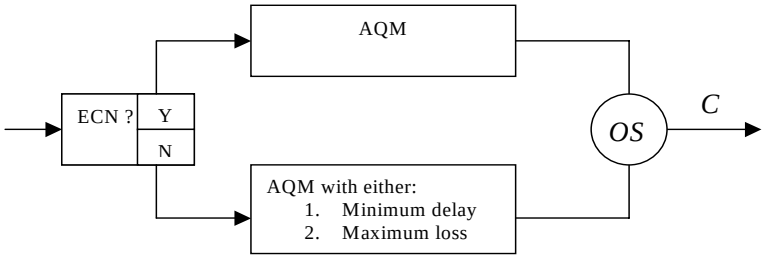
Another approach to the efficiency collapse problem is to limit the maximum dropping rate of the RB AQM to the loss rate upper bound l_d. Therefore the actual dropping rate $D(t)$ of non-ECN packets is derived from $P(t)$ such that

$D(t)=min(l_d,P(t))$. Note that as the load increases, there will be periods when $P(t) > l_d$, and the dropping rate which is limited to l_d will not be enough to control the packet arrival rate to the target capacity. Backlog will build in the queue once $X(t) > C(t)$. However, by (3.23), the increase in RTT due to the backlog will increase the effective congestion control signal $f(RTT,P(t))$. With enough backlog, the sources can be controlled at the target rate even with the limited dropping rate l_d. The amount of backlog (delay) which is necessary can be calculated by (3.24).

We can see that both approaches of dealing with the efficiency collapse problem results in an induced delay D. The approach of inducing a constant minimum delay results in predictable delay, and the approach of inducing a limit on the dropping rate induces this delay when the dropping rate l_d is not adequate to control sources.

The unfortunate consequence of delay D is that ECN enabled flows, which can successfully operate with low latencies, must also suffer this delay. The ideal case, would be to have all sources ECN enabled, and not require the enforced delay D at all. However, the Internet will continue to have a mixture of ECN and non-ECN TCP sources. If RB AQMs were deployed without the enforced delay or maximum loss rate, the low latencies and resulting high congestion notification rate would cause a grossly unfair allocation of bandwidth to ECN enabled flows, as non-ECN flows would suffer many timeouts and retransmission inefficiencies.

Fig. 3.24 Congestion Notification Signal Splitter



To solve this problem a new component is introduced, the Notification Signal Splitter, Fig 3.24. This element maintains a separate queue for ECN enabled packets and non-ECN enabled packets. When a packet arrives, the packet is sent to the

appropriate queue based on the *ECN Capable Transport* (ECT) bit in the IP header. Each queue is controlled by a separate instance of the AQM algorithm. An output scheduler (*OS*) determines capacity sharing between the two queues[2].

The non-ECN queue either induces the minimum delay *D,* or the maximum dropping rate l_d on the packets routed through it. The ECN queue executes the unmodified AQM function, with no minimum delay or marking rate limitation. This allows maximum amount of packets (all ECN packets) to receive the minimum latency (for given target utilisation).

3.5.5 Conclusion

In this section, we have uncovered a significant performance issue in migrating to a low latency RB AQM Internet with non-ECN TCP. The techniques presented prevent the efficiency collapse that would occur if latencies were made very small by deployment of RB AQMs in the presence of non-ECN TCP. A framework for the transition from packet dropping to ECN has been presented which makes possible the benefits of low latency AQMs without the efficiency collapse problem.

3.6 Conclusion

In this chapter, we have surveyed a number of different proposals for AQM structure. We have described general limitations of BB AQMs, and discussed particular limitations of the widely deploy tail-drop queue, and widely published RED AQM paradigm. We have shown that the RB AQM paradigm has clear advantages over the BB AQM paradigm, as it is able to control the utilisation of links, and clear the network of backlog. Nevertheless, as the Internet slowly moves from packet-dropping to ECN marking as the congestion notification signal, we have uncovered some considerations in deployment of RB AQMs in Today's network.

---

[2] In Chapter 4 we detail techniques for interfacing scheduling algorithms to RB AQM control, so we omit this discussion here.

In this chapter, we have concentrated on a single queue link application for RB AQM. In the next chapter, we will further develop the application of RB AQMs to systems with multiple queues.

CHAPTER

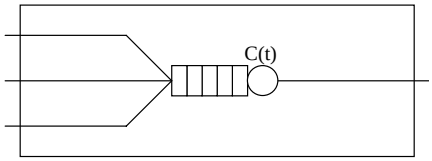
4

Chapter 4: Rate-Based AQM in Multi-Queue Systems; DiffServ and Switch.

4.1 Introduction

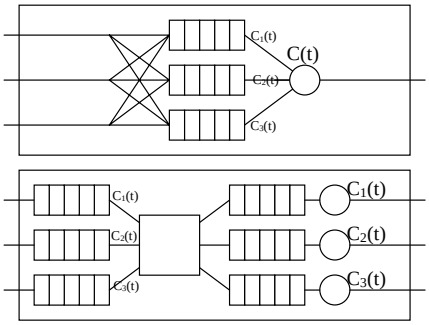
The development of AQM algorithms in the literature has almost exclusively focused on a model of Internet nodes (switches, routers) that consists of a single queue feeding a single output link of a well-defined capacity, such as shown below:

Fig. 4.1 Single Queue Network Node.



However, many real-world systems may comprise of multiple queues per output link, or multiple output links with multiple input links, and the capacity available for each output link can be dynamic because it depends on the cross-traffic inside the system:

Fig. 4.2. Multi-Queue Systems.



Examples of such systems include any link with a scheduling algorithm, like a DiffServ link, and Switches or Routers that have a switching fabric that is shared by several output and input links. Research based on the fundamental single queue model has yielded a good understanding of how congestion control works. The study of the single queue AQM has given landmark results on the stability and convergence of congestion control systems, and the development of better AQM structures. In this chapter, we extend this research by applying existing AQMs developed for the single-queue case to the multi-queue systems. In particular, we will apply the set of RB AQMs to applications with multiple-queues and multiple output links. We will explore two such widespread applications in detail a) the DiffServ link and b) a combined input and output queued switch.

4.2 Rate-Based AQM for DiffServ

4.2.1 Introduction

As discussed in Chapter 2, there are a number of load control mechanisms, characterised by the type of bandwidth guarantees they offer to connections and the amount of connection state information stored in the network. The range goes from connection admission control schemes, such as RSVP, through to stateless congestion control such as TCP/IP. DiffServ occupies a middle ground, where individual connections are controlled on a connectionless/stateless basis from the perspective of the network, but aggregates of flows, i.e. classes, receive pre-configured treatment at links, which requires per-class information at the link. DiffServ is a good compromise between system complexity and performance assurance.

DiffServ classifies packets into different service classes by setting a 6-bit pattern in the IP header, called the Differentiated Services Code Point DSCP. Each of the 64 possible classes is associated with a particular forwarding mechanism at a node, called a Per-hop Behaviour PHB. The PHB affects the relative QoS of each class [103]. The classes are called Behaviour Aggregates (BA). The IETF DiffServ standards describe qualitatively the behaviour expected for each BA but do not regulate the specific forwarding mechanisms required at the routers and switches to

achieve these behaviours. This keeps the door open for different proposals of DiffServ mechanisms. We will describe some of these proposals in this section, and introduce our own proposal which is based on RB AQM control.

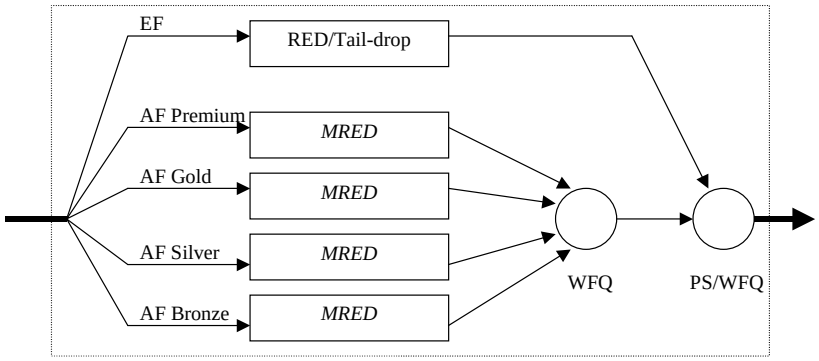
There a number of DiffServ classes currently defined by the IETF. The Expedited Forwarding (EF) class is the highest priority class and is for applications requiring low-loss, low-latency and low-jitter. Typical applications for the EF class include voice over IP VoIP, interactive games, and online trading programs. The Assured Forwarding AF class has a lower priority than EF. AF is comprised of a number of subclasses with different grades of service priority. The AF class is intended for best-effort applications such as FTP, and WWW. Each subclass within AF is identified by the notation AFxy, where x specifies the service class and y the packet drop precedence. The AF class is subdivided into 4 service classes, coined Premium AF1y, Gold AF2y Silver AF3y and Bronze AF4y. The IETF specifies [121] that each of these subclasses should receive a guaranteed minimum share of link capacity. Typically, the bandwidth available B(AFxy) to a class x in a bottleneck link is such that B(AF1y) > B(AF2y) > B(AF3y) > B(AF4y). Within each subclass, one of three levels of the packet drop precedence (y) is specified. The packet drop probability dP(AFxy) of each level should be dP(AFx1) <= dP(Afx2) <= dP(Afx3).

The DiffServ system is composed of a number of different components. *Boundary nodes* monitor and control traffic input into the DiffServ system at the edge of the network. Boundary nodes include *packet classifiers* which map IP packets into particular classes and *traffic conditioning* which performs metering, shaping and policing of traffic entering classes. In the core network, DiffServ is comprised of *interior nodes*, which perform the forwarding with different per-hop behaviours. For a detailed description of these elements of DiffServ, the reader is referred to [10] [15] and [29]. In this thesis, we will focus on the DiffServ congestion control mechanism which exists in a DiffServ link to implement the PHB.

The congestion control mechanism by which DiffServ achieves the prioritisation and differentiation of service can be separated into two independent and concurrent mechanisms: (1) open-loop packet scheduling algorithms which enforce statically pre-configured prioritisation of packets based on class (2) closed-loop

AQM which controls the transmission of packets of each service class onto the end-to-end source to destination path and controls packet drop precedence. Fig 4.3 shows these two mechanisms in a typical DiffServ implementation.

Fig. 4.3 Typical DiffServ Implementation



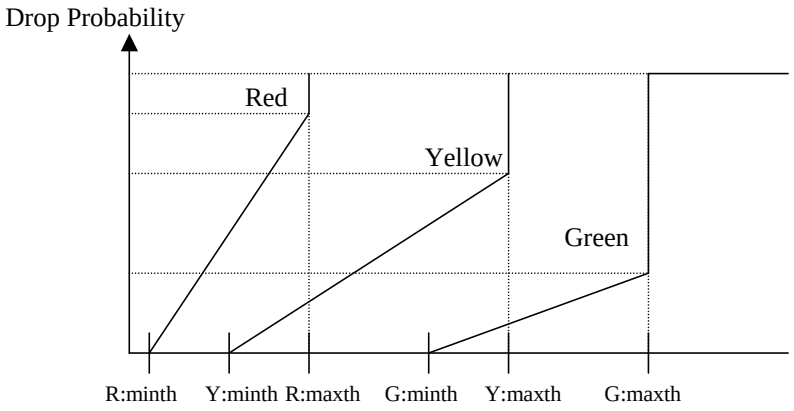
The ‘open-loop’ scheduling algorithms determine the allocation of bandwidth to each class, as well as packet transmission order. Examples of scheduling disciplines include: First In First Out (FIFO) [49], Round Robin (RR) [87], Priority Scheduling (PS) [87] and Weighted Fair Queuing (WFQ) [87, 102]. A typical DiffServ implementation uses a priority scheduler to give the EF class packets absolute priority over AF class packets. The bandwidth available to the AF class is typically shared between each AF subclassed by using a WFQ scheduler. The WFQ scheduler ensures that each service class received a pre-configured portion of bandwidth during overload.

The DiffServ link has a separate queue for each of the classes (EF, AF Gold, AF Silver etc). Each of these queues is managed by a separate AQM algorithm. The AQM also determines the drop precedence within each AF class. Current proposals in literature and implementations of AQM in DiffServ are based on extensions to the RED AQM to provide the different packet drop precedencies. These extensions to RED are collectively known as multi-level RED (MRED) [80], and specific RED variants in this group include: WRED [23] [27], RIO-C [80, 110], and RIO-DC [80]. Not only has MRED received significant research interest, but commercial routers

have been deployed using the MRED approach, such as the CISCO IOS routers [24, 27] which use WRED.

The WRED algorithm is representative of the other MRED algorithms, so we will describe it here in more detail. WRED computes a single average queue size which includes packets from all drop precedencies. By convention, packets of the three drop precedence within an AF class are delimitated by the colors red, green and yellow, such that the packet drop probability of each color is; Pr(red) > Pr(yellow) > Pr(green). WRED gives each color a separate packet dropping probability as shown in Fig. 4.4 based on the aggregate queue size for all colors.

Fig 4.4 WRED Marking Function



Although today's premier DiffServ architecture consists of MRED, literature has shown that performance of RB AQM control, such as REM [76] or GREEN [123], has significant advantages over BB AQM techniques. In this section, we form a basis for a RB AQM DiffServ architecture by applying RB AQM control to packet scheduling algorithms. We will show that a RB AQM DiffServ link can achieve very low packet latency and loss compared to the current MRED techniques. Our discussion begins by describing how DiffServ fits in with the classless network paradigm discussed in Chapter 2.

4.2.2 Need for DiffServ

The work by F. Kelly's [62, 64] suggests that service differentiation is possible in a classless network paradigm. A number of papers [66, 76, 98] have proposed a completely 'economic' driven architecture for differentiated services without schedulers and packet classes. In this subsection, we will discuss why a class-based network such as DiffServ is still fundamentally necessary, and therefore why we are interested in applying RB AQM to DiffServ. We will give an overview of how DiffServ fits into the broader load control area and why combining RB AQM control with packet scheduling is desirable.

In the classless paradigm, sources differentiate their demand for bandwidth by having different utility functions, $x = U(p)$, which determine the source's transmission rate x based on the current network congestion 'price' p. Sources which require more bandwidth than others, simply send more, suffering a higher 'price' p. In such a system, scheduling algorithms are not used because the allocation of bandwidth is determined solely by the economic process. As discussed in [62] and described in Chapter 2, it has been shown that such a system maximises the aggregate of the utilities of all the sources.

If maximising the aggregate utility of the system is the only criterion, this system is sufficient. However, no guarantees can be made about the amount of bandwidth actually allocated to each source, because the current 'market' of all sources on the network determines this. To guarantee a minimum on the bandwidth a source will receive requires knowledge of all of the utility functions of all of the sources sharing the link. If this information is available, it is then possible to calculate how much bandwidth each source will receive. However, in practice, a network will consist of sources with known demand, or utility, functions $U_1(p) \ldots U_N(p)$, and other sources with unknown utility functions $X_1(p) \ldots X_M(p)$. To fully characterise the bandwidth demands of a network and determine all of the utility functions of the sources acting at a given time, full knowledge of all applications that the users may execute and the time they are executed is required. In practice such information is not accessible and therefore it is in not possible to predict the

bandwidth allocation to each source in a network. Furthermore, the presence of malicious sources which are not responsive to congestion makes it even more difficult to predicting bandwidth usage. Therefore it is not possible to guarantee any minimum rate to a source.

However, a real network will consist of a subset of sources which require a minimum rate guarantee, and a subset which are satisfied by their 'market-share'. Many real applications require minimum rate guarantees. For example, an office with a set of voice-over IP telephones, an interactive online game, or video-conferencing, all require a guaranteed amount of bandwidth from the network at any time, regardless of the background traffic. If a subset of sources needs a guaranteed minimum rate from a link, the economic system is not sufficient. A flow isolation mechanism, which removes the known flows needing guarantees out of the competitive economic environment that contains other flows with unknown utility functions, is essential to guarantee minimum rates. DiffServ, with packet marking and link class-based scheduling, does this by guaranteeing minimum capacity to subsets of known flows. In practice, IP router manufactures have recognised this need for scheduling algorithms by incorporating class-based QoS mechanisms inside their products [24].

Now that we have presented the need for (1) class-based scheduling algorithms and (2) RB AQM control in Chapter 3, the algorithm which combines the two is presented next, followed by an empirical performance evaluation.

4.2.3 Algorithm Background

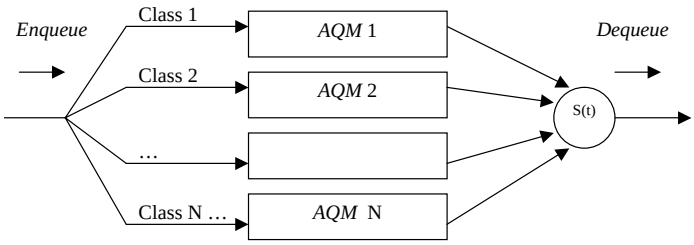
The RB AQM DiffServ algorithm replaces MRED queue management. Our proposed design of a multi-class RB AQM applies to any work conserving scheduler; however we will limit our discussion to the WFQ scheduler.

The structure of the multi-class RB AQM combined with a WFQ scheduler is shown in Fig. 4.5. Each queue in the scheduler is managed by a separate instance of a RB AQM algorithm, of the basic form:

$$P(t+1) = P(t) + \alpha \times (X(t) - u \times C(t)) \qquad (4.1)$$

The WFQ algorithm is parameterised by a weight vector *W*, where the element W_i is the proportion of capacity class *i* receives when there are packets available for transmission for all classes. For a detailed description of the WFQ scheduler, the reader is referred to [87, 102].

Fig. 4.5 RB AQM Architecture in a Class-Based Scheduler



The distinctive issue, faced by RB AQM in a class-bases scheduler, is that the capacity available to each class *i*, denoted c_i, and the packet arrival rate for that class, denoted x_i, need to be known. In work conserving scheduler, such as WFQ, where unused capacity in one class is redistributed to other classes, the capacity available to each class is time-varying and depends on, and affects, the traffic in other classes. In the following sections, we will present a technique for calculating and controlling c_i, the capacity allocated to each class, to make multi-class RB AQM possible.

A basic algorithm is introduced in the following subsection which results in a functional work conserving RB AQM system, where each class is guaranteed its minimum share, M_i. However, the capacity above the minimum is not distributed with any notion of fairness. Instead, the classes with the most aggressive traffic win the slack capacity. Later, we present a notion of weighted fairness, and an extension to the basic mechanism to enforce it.

4.2.4 Basic Algorithm

Consider a stream of packets scheduled by a work-conserving WFQ scheduler, of *N* classes. Let *B* be the vector representing the sizes (bits) of the *H* packets that have been served most recently. The order of the elements of vector *B*

are in reverse order to their service completion times. In other words, B_0 is the size of the most recently served packet, B_1 is the size of the previous packet and so on. Finally, B_H is the size of the oldest packet packet in *B*. Similarly, we define the vector *C*, of *H* elements, such that C_j is the class ($Cj \in \{1, 2, 3, \ldots N\}$) of the packet represented by B_j, $j = 1, 2, 3, \ldots H$.

Let *S(t)* be the physical capacity of the link at time *t*. When *S(t)* is time varying, such as with Ethernet, DSL, or radio, it can be estimated from the last packet's transmission time. The scheduling algorithm, such as WFQ, guarantees minimum rates to each class. The weight W_i corresponds to the share of capacity that each class *i* is guaranteed.

In a work conserving scheduler, the actual capacity available to a class depends on the traffic in other classes as well as on the minimum rate allocation W_i. Whilst W_i controls the lower-bound, a class may receiving more capacity than the W_i portion when there is no traffic in other classes. Without apriori knowledge of the traffic, the future capacity available to a class can only be estimated from the previous capacity. To estimate the class capacity, let the identity function *I(j,i)* be:

$$I(j,i) = \begin{cases} 1 & \text{if } C_j = i \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

The estimate class capacity, $S_i(t)$, is calculated from the portion of server time allocated to class *i* by the scheduling mechanism in the past *H* packets:

$$S_i(t) = \frac{\sum_{j=0}^{H} B_j(t) \cdot I(j,i)}{\sum_{j=0}^{H} B_j} S(t) \qquad \text{where } i < N. \tag{4.3}$$

Note reduced complexity techniques such as exponential averaging could be employed to compute (4.3). The lower bound on the capacity is obtained by considering the minimum service rate guaranteed by the WFQ scheduling mechanism, given by:

$$M_i(t) = \frac{W_i}{\sum_{j=1}^{N} W_j} S(t). \quad (4.4)$$

The capacity allocated $c_i(t)$ to each class i is a projection of previous capacity attained $S_i(t)$, bounded by the minimum rate guarantee $M_i(t)$:

$$c_i(t) = Max(M_i(t), S_i(t)) \quad (4.5)$$

Since a process for computing $c_i(t)$ is obtained, all of the inputs for AQM algorithm are available. Note (4.5) is evaluated at each update of the AQM which at the maximum rate, is at every packet arrival.

Notice that $c_i(t)$ is the capacity allocated to class i, not the capacity actually consumed by class i. The capacity not consumed by the class to which it is allocated, may be used by other classes. If for example, no class i packets arrive, $s_i(t)$ will be 0, and $c_i(t)=M_i(t)$. Although in this case no capacity is consumed by class i, if a burst of class i packets were to arrive, $M_i(t)$ capacity is guaranteed.

4.2.5 Extended Fair Share Algorithm

The algorithm in 4.2.4 is extended here to enforce proportional fairness. The fair allocation enforcement applies only to bottlenecked classes, where $x_i(t) \geq c_i(t)$. Classes which are not bottlenecked at the link, $x_i(t) < c_i(t)$, need no enforcement of fairness, since their rate is below their fair capacity and their bandwidth demand is satisfied. We define a weighted fair allocation of capacity to a bottlenecked class i, $F_i(t)$, as:

$$F_i(t) = \frac{W_i}{\sum_{j=all\ \ bottlenecked\ classes} W_j} (S(t) - \sum_{j=all\ non-bottlenecked\ classes} x_j). \quad (4.6)$$

This proportionally fair allocation regime ensures that the slack capacity from non-bottlenecked classes is allocated to bottlenecked classes in proportion to the

weight W_i. In this extended algorithm, the capacity of non-bottlenecked classes is given by (4.5), and for bottlenecked classes, the capacity is given be (4.6).

Notice that the total capacity allocated for both non-bottlenecked classes and bottlenecked classed, may be more than *S(t)*. However, the non-bottlenecked classes do not utilise their allocated capacity $c_i(t)$, and only the amount of capacity which is not utilised by the non-bottlenecked classes, is re-assigned to bottlenecked classes by (4.6).

4.2.6 RB AQM Packet Drop Precedence

Differentiated drop precedence within each AF class is required by IETF DiffServ standards. Previously, we described how BB AQMs achieve this by using a multi-level MRED architecture. Here we will extend the RB AQM to achieve multi-level drop precedence within a given AF class.

We will assume that the RB AQM is able to keep a low mean queuing delay and the buffer size is large so that all packet drops are due to congestion notification events generated by the AQM. Let us define the precedence of packet drops for each of the three traffic 'colors' within an AF class (red,yellow,green) by the weights $\alpha_{red}, \alpha_{yellow}, \alpha_{green}$ where $\alpha_{red} > \alpha_{yellow} > \alpha_{green}$, $0 > \alpha_{red}, \alpha_{yellow}, \alpha_{green} > 1$ and $\alpha_{red} + \alpha_{yellow} + \alpha_{green} = 1$.

The RB AQM computes the packet drop probability *P(t)* based on the aggregate traffic for all *colors*. Then the packet drop probability for a given *color*, which is defined as the number of dropped packets of the *color* compared to the number of packet arrivals at the link of that *color* is:

$$P_{color}(t) = \left[P(t) \cdot \alpha_{color}\right]_0^1$$

then we satisfy the drop precedence requirement, since $\alpha_{red} > \alpha_{yellow} > \alpha_{green}$:

$$P_{red}(t) > P_{yellow}(t) > P_{green}(t)$$

Because the congestion notification rate is now rescaled, the bounds of *P(t)* also need to be rescaled. The bounds of *P(t)* are now $0 \geq P(t) \geq \frac{1}{\alpha_{green}}$, since this allows each *color* to have a drop probability range of 0 to 1.

Note that in the following performance evaluation section, we ignore this relatively minor issue of drop precedence and apply the single *P(t)* to all packets within a class, focussing on the dynamics of the aggregate behaviour of the class.

4.2.7 Implementation and Transient Performance Evaluation

In this section, we describe the implementation of the multi-class RB AQM algorithm and later demonstrate its performance with simulations. The simulations results will demonstrate that the multi-queue RB AQM system achieves proportional allocation between classes, under a variety of traffic scenarios, and that it also outperforms the multi-queue BB AQM system, as it can maintain a consistent low queuing delay for a range of loads.

4.2.7.1 Multi-class RB AQM Implementation

In the multi-class RB AQM, each class *i* has a separate instance of an AQM algorithm. For class *i*, the RB AQM we implemented in the simulations was the GREEN AQM, as described in Section (3.3.1).

The WFQ scheduling algorithm used in the simulations of the multi-class RB AQM system is described by the pseudo-code in Fig 4.6. The *wfq.deque* function is invoked when the link is ready to transmit the next packet and the *wfq.enque* function is invoked when a packet is received for transmission onto the link. This particular WFQ variant gives immediate service to packets of higher priority class. A packet queued in a higher priority class will always be served next, so long as the class's work quantum, W[I], has not been exceeded. This allows the RB AQM & WFQ system to be used for serving the EF traffic as well as AF traffic. EF packets can simply be assigned to the highest priority class, with AF traffic occupying lower classes.

Fig. 4.6. WFQ Scheduler Pseudocode

```
pkt* wfq.deque()
{
while(TRUE) {
        for I = 1 to N {
                if (class[I].nextpkt.size < S[I])
                {
                        S[I] = S[I] - class[I].nextpkt.size();
                        return (class[I].dequeue);
                }
        }
        for I = 1 to N {
                if (S[I] < MaxS );
                        S[I] = S[I]  + W[I];
        }
}
}

wfq.enque(pkt *packet)
{
        class[packet.class].enque(packet);
}
```

Where

N = number of classes.

W[I] = service quantum (bits) per round given to class I, where I = 0 for highest priority class, I = N for lowest priority class.

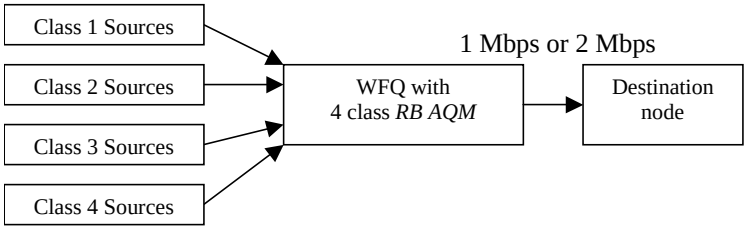
S[I] = remaining service quantum for class I in current round.

class[I].nextpkt.size = function returns the size (bits) of the next packet in class

4.2.7.2 Performance Evaluation

The RB AQM DiffServ was simulated using Network Simulator 2 [94] to: (a) validate its design and performance and (b) compare its performance to the existing BB AQM DiffServ. Three scenarios simulated are presented in the following sections, one with TCP traffic only, another with both TCP and UDP traffic and a third with real-time traffic. All scenarios used the same network topology, as depicted in Fig. 4.7. For Scenarios 1 and 2, the Diffserv managed link has a 1 Mbps capacity and it is 2 Mbps in Scenario 3. Multiple TCP or UDP sessions are aggregated to form the traffic of each of the four classes transported over the link. TCP sessions are ECN capable. All data packets are 1000 bytes. We will now describe each simulation scenario and the results.

Fig. 4.7 Overview of Simulation Topology



Scenario 1A and 1B: TCP Traffic Simulation Set-up

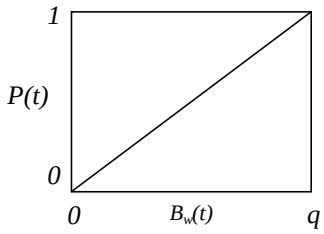
In Scenario 1A and 1B, we compare the delay and utilisation performance of a RB AQM DiffServ link with a BB AQM DiffServ link, when the traffic in all classes is generated by TCP sources. Scenario 1A simulates the multi-class RB AQM with the fairness enhancement (4.2.5) and Scenario 1B simulates a multi-class BB AQM DiffServ, using WRED with WFQ. The flow rates of traffic in each class and the total number of packets backlogged for all classes are measured. The parameters for this scenario are listed in Fig. 4.8.

Fig. 4.8 Simulation Parameters for Scenario 1A & 1B

Class	u_i *Utilisation*	W_i	Sources	Start (sec)	Stop (sec)
1	0.93	8001	8 TCP 40ms RTT	0	100
2	0.93	4001	8 TCP 40ms RTT	20	140
3	0.93	2001	16 TCP 40ms RTT	40	180
4	0.93	1001	16 TCP 40ms RTT	60	220

The WRED implementation used for the BB AQM scenario 1B is a linearised version of WRED. The marking function takes the form shown in Fig. 4.9.

Fig. 4.9 WRED Marking Function



The WRED implementation uses a weighted average of backlog, denoted $B_w(t)$, to determine the packet marking probability. The marking probability is related linearly to $B_w(t)$, by $P(t) = \alpha B_w(t)$, where α is the reciprocal of the maximum queue size for each class q. In Scenario 1B the queue size q is 10.

Scenario 1A and 1B: TCP Traffic Simulation Results

Results shown in Fig. 4.10 confirm that a weighted fair allocation of capacity is achieved with the RB AQM DiffServ, as the magnitude of the flow rate from each class is proportional to its minimum rate W when the traffic from different classes is switched on and off.

Results in Figs. 4.11 and 4.13 show the total backlog in all classes for the RB AQM and BB AQM (WRED) system respectively. The thick black line in both graphs shows the average backlog measured over 300 packets. The figures illustrate the systemic problems of WRED and WFQ. For BB AQM, in the interval 50s to 100s, when all classes are active, note that backlog has significantly increased because of the increased traffic load. This illustrates the previous analysis, that with BB control where $P(t)=f(b(t))$, backlog is coupled to the load by the control system. On the other hand, the results for the RB AQM in Fig. 4.11, indicate that the backlog is not coupled to load, and queuing delay does not increase with load.

Fig. 4.10 Scenario 1A: RB Packet flow rate

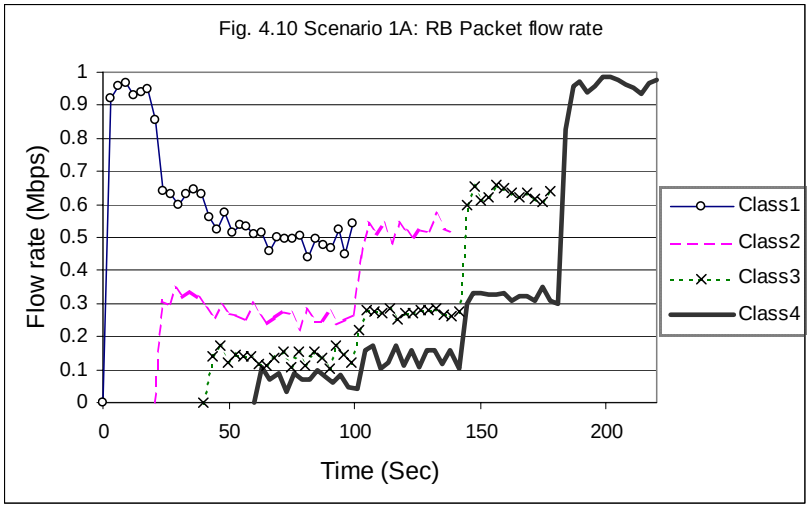


Fig. 4.11 Scenario 1A: RB Aggregate Backlog

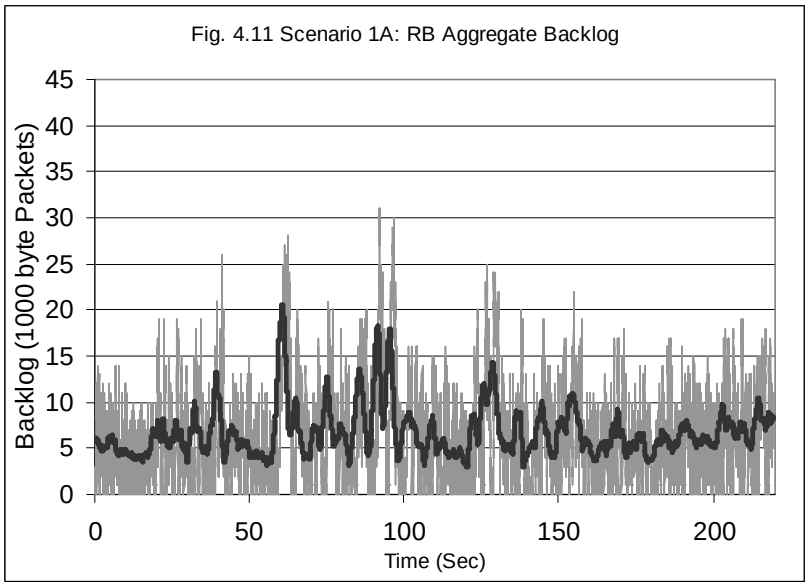


Fig. 4.12 Scenario 1B: WRED Packet flow rate

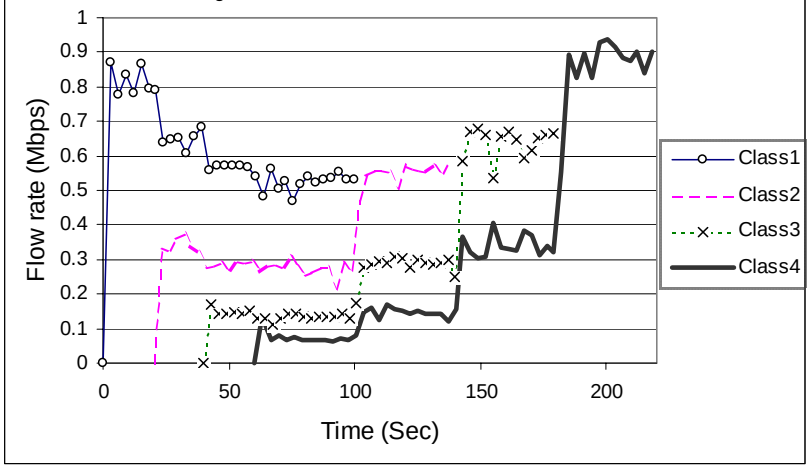
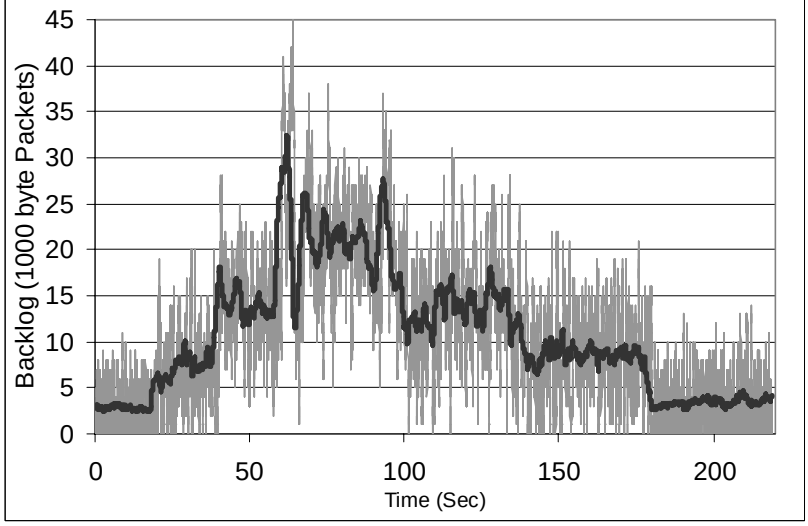


Fig. 4.13 Scenario 1B:WRED Aggregate Backlog



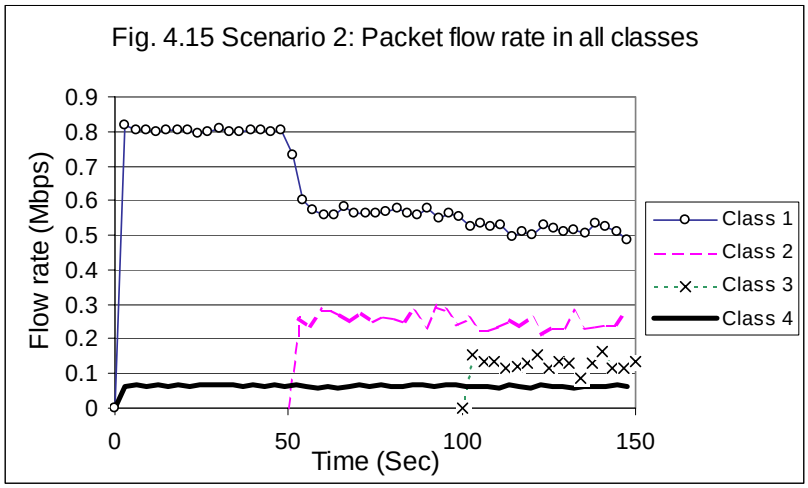
Scenario 2: TCP and UDP traffic

In this scenario, we will demonstrate that RB AQM DiffServ can maintain proportional fairness between classes even when the traffic contains malicious sources, which are not responsive to congestion control. In this scenario, the traffic in Classes 1 and 4 is from UDP constant bit rate sources transmitting at 0.8 Mbps and 0.05 Mbps respectively. The traffic in Classes 2 and 3 is from congestion responsive TCP sources. The CBR Class 1 UDP traffic rate is above its proportional share when other classes are active. For the complete parameters refer to Fig 4.14.

Fig. 4.14 Simulation Parameters for Scenario 2

Class	u_i *Utilisation*	W_i	Sources	Start (sec)	Stop (sec)
1	0.93	8001	1 UDP 20ms RTT 0.8 Mbps	0	150
2	0.93	4001	16 TCP 20ms RTT	50	150
3	0.93	2001	16 TCP 20ms RTT	100	150
4	0.93	1001	1 UDP 20ms RTT 0.05 Mbps	0	150

The results in Fig. 4.15 show that RB AQM DiffServ is able to allocate bandwidth with proportional fairness despite the presence of an unfriendly, non-congestion-controlled UDP sources. Notice that at 50sec, when Class 2 traffic is switched on, the UDP traffic in Class 1 is throttled down to its fair share by an increased packet dropping rate. At this point Class 1 becomes a bottlenecked class. In this way, the TCP sources can attain their configured share despite the aggressive UDP source.



Scenario 3: Real-Time Traffic

In this scenario, we simulate real-time traffic and demonstrate how the latency performance of the RB AQM DiffServ architecture is better than the BB AQM architecture. Two classes are used to simulate the interaction of data traffic

and real-time traffic. Class 2 contains TCP/FTP data traffic, and is insensitive to delay. Class 1 is the real-time traffic, with a hard maximum queuing delay requirement of < 50 ms. The real-time traffic in Class 1 is generated by saturated TCP transfers. As discussed later, this simulates a rate-adaptive video traffic. The load (number of sessions) of Class 1 sessions is varied in the experiment from 1 to 450. A number of trails were simulated, using WRED with queue size value q set to 5 (WRED5),10 (WRED10) and 20 (WRED20) packets, and using RB control with parameter u_1 set to 0.8 (RB80) and 0.85 (RB85). The Diffserv link capacity is 2 Mbps, with 1 Mbps assigned to Class 1 and 1 Mbps assigned to Class 2.

Fig 4.16 Simulation Parameters for Scenario 3

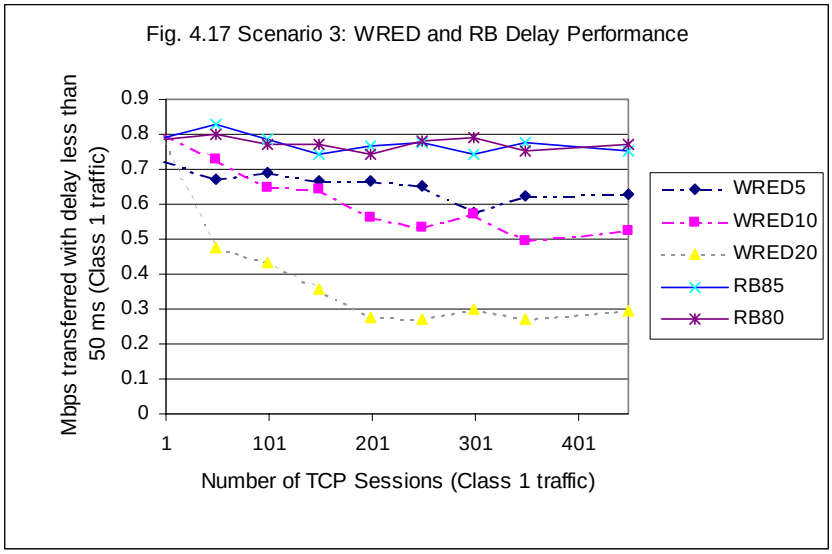
Class	u_i *Utilisation*	W_i	Sources	Start (sec)	Stop (sec)
1	0.8, 0.85	2001	50-450 TCP 40ms RTT	0	450
2	0.95	2001	8 TCP 40ms RTT	0	450

TCP is used to approximate a real-time adaptive multi-rate source. Audio and video protocols are typically based on UDP, RTP and RTCP. However, recent real-time multimedia protocols respond to loss by adjusting their rate, and are thus in principle similar to TCP [32] [107] [19] [97]. Although their transient behaviour, and amount of response to loss is different than TCP, any real-time protocol that seeks to take advantage of available capacity on a best effort network, must in principle be congestion controlled. Unless the real-time source increases its rate when there is available capacity, and decreases it when capacity decreases, the quality of transmission is suboptimal. Many existing CODECS are designed for varying channel conditions, such as a best effort network. For instance, the G.723.1 Audio speech codec adjusts its output rate, and adapts to the available bandwidth. Similarly, MPEG-4 includes extensive support for multi-layered, multi-rate video. The RTP communicates the amount of packets lost, which allows the sender to adapt its rate to the channel. We model adaptive multimedia sources as saturated sources, such as an FTP transfer with infinite data, as the source always has more video or audio information that it could possibly send to improve quality.

In the simulation, we measure the amount of packets, in Mbps, which are delivered with less than 50ms queuing delay in the Diffserv queue. Packets delivered late, >50ms, no longer contain useful information to a real-time application and do not contribute to the Mbps. Since real-time sources do not retransmit packets, the TCP packet retransmissions are considered as new packets in the simulation.

The results, in Fig. 4.17, show how for a variety of settings, and traffic loads, RB AQM DiffServ effectively delivers more data within the delay requirement. As the load of real-time traffic increases, RB AQM delay performance does not deteriorate. On the other hand, the BB AQM DiffServ effectively delivers less real-time data as the load increases. This is due to the load and delay coupling of BB AQM control.

Fig. 4.17 Scenario 3: WRED and RB Delay Performance



As discussed previously, the problem with BB schemes such as WRED, is that the backlog must be positive for source rate to be controlled. In this trial, the maximum queue size for WRED was reduced from 20 to 10 and then to 5. Reducing the maximum queue size gave diminishing returns since the utilisation was significantly lowered. On the other hand, increasing the queue size resulted in a higher average backlog, which delayed more traffic beyond the 50ms requirement.

Also, as evident in Fig. 4.17, unlike RB control, the optimal setting of parameters for WRED varied widely with the traffic load.

4.2.8 Comment on Performance with Unresponsive Flows

An important issue is the performance of non-congestion controlled UDP traffic. In the previous section, we dealt with rate adaptive real-time sessions, however non-rate-adaptive UDP still commonly used to carry real-time services which have an upper bound delay requirement. If such traffic receives enough capacity, both BB and RB schemes function identically. However, when the amount of non-congestion controlled UDP traffic exceeds the capacity, BB schemes, such as WRED will increase backlog and delay, whereas RB control will prevent excessive delay by increasing the dropping rate without the backlog increasing. This means that is instead of being excessively delayed, packets are discarded. Therefore in a congestion situation, the portion of packets which are transmitted, still meet the delay requirements. The portion which are discarded would likely not have been able to be served within the delay requirement. With WRED, in a congestion situation, the delay performance of all packets suffers.

4.2.9 RB AQM DiffServ Conclusion

In the previous section, we presented a technique for applying rate-based active queue management to a class-based scheduling algorithm. The method presented is scalable since it is low in computational complexity. It forms a solid architecture for DiffServ implementation in routers and switches and has been shown to outperform the current WRED with WFQ architecture. Furthermore, this work will enable the wide body of research into rate-based congestion control schemes to be applied to improving the performance of DiffServ.

4.3 RB AQM CIOQ Switch

In this section, we investigate the implementation RB AQM algorithms in another multi-queue system, the combined input and output queued switch. We find that a structure with an AQM only at each output port is possible and we analyse the

constraints on the switch fabric speed that this structure imposes. A simulation is performed to validate our design and its performance is compared with the current tail-drop queue switch.

4.3.1 Introduction

Devices such as combined input and output queued (CIOQ) switches have multiple queues whose capacities depend on cross traffic between multiple input and output interfaces. There are numerous examples of commercially available CIOQ switches, which include the Hewlett-Packard Procurve Series [55] and Cisco switches [1]. Until now, CIOQ switches have been studied and built using tail-drop and RED queues. However, the promising AQM architectures, such as REM or GREEN which involve the measurement of packet arrival rate as a means of measuring the congestion, have not been applied to the CIOQ switches. This section proposes a structure for implementing these RB AQM algorithms in a CIOQ switch.

4.3.2 CIOQ Switch background

A switch consists of three major components, the input interfaces, the output interfaces and the switching fabric. Higher end switches consist of separate I/O modules and a separate switch fabric in the form of a back plane for interconnecting the modules. The I/O modules may have a number of different physical interfaces. In this section, we will use this modular topology of a switch for our proposal and analysis.

Fig. 4.18 CIOQ Switch

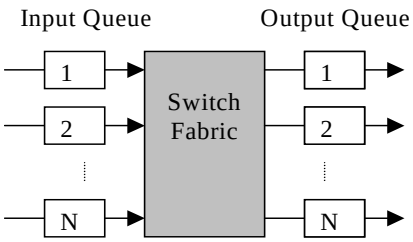
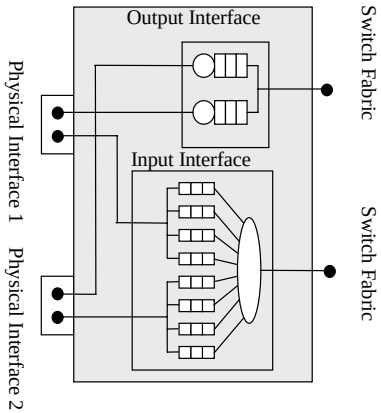


Fig. 4.19 CIOQ Switch I/O Module



The switching fabric is the core element of the switch. It is a resource shared by all of the I/O modules to forward packets from the input interface to the output interface. Different switching fabrics are discussed in [48]. This paper will focus on a space division switch fabric which has a dedicated channel for each fabric output port. Such switches are typically implemented using a cross-bar architecture [111]. The switch fabric normally operates at a rate *S* times faster than the I/O modules [74], where *S* is called the speedup.

As discussed in detail in [73], there are several strategies for the structure of switch queues. These include Input queued (IQ), output queued (OQ) and combined input and output queued (CIOQ) switches. An OQ switch has a FIFO queues at each output interface. However, this requires a speedup of *N*, the number of output ports, for both the switch fabric and buffer memory bandwidth to be non-blocking. This makes OQ switches impractical for high-capacity, high port count switches. An IQ switch has a single FIFO buffer between each input interface and each switch fabric input port. This design has its caveats too. When several packets arrive at an input interfaces destined for the same fabric output port, the switch fabric can only forward one packet at a time and the remaining packets are kept in the FIFO buffers until the port is free. This can result in Head-of-Line (HOL) blocking, where packets behind the blocked packets are also blocked, even though their destination port is free. To avoid HOL blocking, IQ switches separate each input FIFO buffer into *M* separate FIFO buffers, with a separate FIFO buffer for each switch fabric output port from each input interface. However, the scheduling of packets from each input queue

through the switch matrix has a significant computational cost and requires centralized coordination and communication with each input module. The optimal scheduling solution is too computationally intensive for practical implementation, and sub optimal approximations are used.

To improve the performance of these sub optimal scheduling, the switch fabric operates with a speed-up. This switch fabric speedup necessitates an output queue, since the sped-up switch fabric can now forward packets to the output interface at a rate above the capacity of the interface. The result of adding the output queue to the IQ strategy is the CIOQ strategy. It has been shown [22] that a CIOQ switch with a speed-up in the order of 4 can achieve 99% throughput. In [74] it is also discussed that for a certain kind of schedulers, even a speedup of 2 can be almost non-blocking. The advantage of the CIOQ strategy is that it allows switches to operate with near OQ performance with only a moderate speedup, which is far less demanding on the components than the *N* speedup required by the OQ structure. The CIOQ strategy is used in many commercial high-performance switches with multiple interfaces and we will limit our proposal and analysis to this type of switch.

To avoid head-of-line HOL [74] blocking, each input interface has a separate queue for packets destined for different output interfaces. An example CIOQ I/O module is shown in Fig. 4.19, where there are four separate input queues for each of the two input interface, which allow the switch to have four different output interfaces. The module has two output interfaces, and therefore has two output queues.

The remainder of this section is organized as follows: in Section 4.3.3, we describe the proposed architecture and in Section 4.3.4, we state the capacity constraints that this architecture imposes on the switch design. In Section 4.3.5, we describe a method for modelling the multiple-queue switch which we simulate in Section 4.3.7.

4.3.3 Proposed RB AQM CIOQ Switch Architecture

Fig. 4.20 illustrates an example of the proposed architecture of a RB AQM switch. We conceptually separate the input and output functionality of an I/O

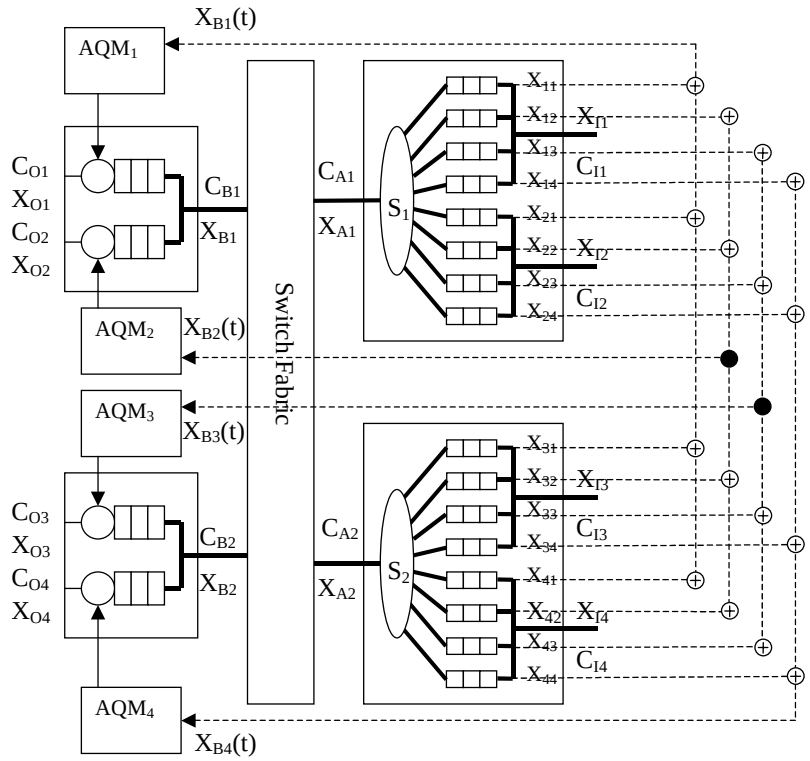
module, into separate virtual input and output modules interconnected by a switch fabric. For each physical output port m, there is one instance of the control algorithm AQM_m.

In our analysis, the switch fabric is abstracted from its physical implementation. The parameters which completely define the switch fabric are the input port capacity C_{Ai} for each input port i, and output port capacity C_{Bj} for each output port j. We assume that given these capacity constraints, the switch fabric is able to forward any input to output port traffic.

There are I input modules in the switch. Each input module d has a set of input interfaces IC_d. Each input interface n is an element in the set IC_d and has capacity C_{In}. The aggregate traffic arriving at input interface n is X_{In}. The traffic arriving at input interface n destined for output interface m is X_{nm}. Each input module has a separate queue for each output interface destination. Each input module d has a work conserving scheduler S_d, for example a round robin scheduler, which determines the next packet to forward to the switch fabric input port. The aggregate traffic forwarded from input module d to a switch input port is X_{Ad}.

There are O output modules in the switch. Each output module e has a set of output interfaces OC_e. Each output interface m is an element in the set OC_e and has capacity C_{Om}. The traffic output by an output interface m is X_{Om}. The traffic arriving at the input of the output module e from the switch fabric output port is X_{Be}.

Fig. 4.20 RB AQM Switch Architecture



Since each AQM_m has the structure given by (4.1), the total packet arrival rate $X_m(t)$ and packet backlog $B_m(t)$ destined for output interface m as well as the output interface capacity $C_{Om}(t)$ must be measured. The capacity $C_{Om}(t)$ can be estimated by measuring the packet size and transmission time, and the derivative term $X'_m(t)$ can be calculated from a measurement of $X_m(t)$.

To measure $X_m(t)$ each input module must measure and store $X_{nm}(t)$ and $B_{nm}(t)$, the backlog of packets in input interface n destined for output interface m. The maximum rate at which these $B_{nm}(t)$ and $X_{nm}(t)$ measurements need to be communicated from the input module to the *AQM* algorithm is at each packet arrival. The total packet arrival rate X_m(t) and backlog B_m(t) is calculated as inputs into the AQM algorithm:

$$X_m(t) = \sum_{n=1}^{I} X_{nm}(t) \tag{4.7}$$

$$B_k(t) = \sum_{n=1}^{I} B_{nm}(t) + B_{Am}(t) \tag{4.8}$$

where $B_{Am}(t)$ is the packet backlog in the output interface m. Now that each input of the AQM has been described, the AQM (4.1) can be evaluated to update the marking/dropping rate $P_m(t)$. The marking and dropping strategy is different. For ECN capable packets, the mark-front strategy investigated in [75] reduces the feedback time of the congestion signal. Therefore ECN capable packets are randomly marked in the output module at the head of the output queue. Non-ECN capable packets, are randomly dropped and this should be done at the earliest point in the switch. This prevents the packet which will ultimately be dropped from being forwarded through the switch and avoids wasting switch fabric and buffer capacity.

4.3.4 RB AQM CIOQ Switch Fabric Constraints

In this section, we formally specify the switch fabric capacity requirements for controlling congestion with a single AQM controller per output port.

Let us assume end-to-end flow control is stable and source rates converge to a steady state at or below the network capacity. Then we will show that as long as the output interface capacity is the only bottleneck in the steady state, a single AQM per output port design is sufficient. The two conditions for this are:

1. For any input module d, the switch fabric input port capacity C_{Ad} connecting to it must be able to carry all of the incoming traffic $X_{Ad}(t)$ from module d:

$$C_{Ad} \geq \sum_{n \in IC_d} C_{In} \geq X_{Ad}(t) \quad d = 1,2...I \quad . \tag{4.9}$$

2. For any output module e, the switch fabric output port capacity C_{Be} driving the module must have enough capacity to fully load the physical capacity of all of the output interfaces of the module:

$$C_{Be} > \sum_{m \in OCe} C_{Om} \quad e = 1,2...O \quad . \tag{4.10}$$

In steady state $\hat{X}_{nm} = u_m.C_{Om}(t)$, therefore:

$$C_{Om} \geq \sum_{m=1}^{I} \hat{X}_{nm} \quad m = 1,2..O \; . \tag{4.11}$$

Then given (4.9) and (4.10),

$$C_{Be} > \sum_{m \in OC_e} \sum_{n=1}^{I} \hat{X}_{nm} \; . \tag{4.12}$$

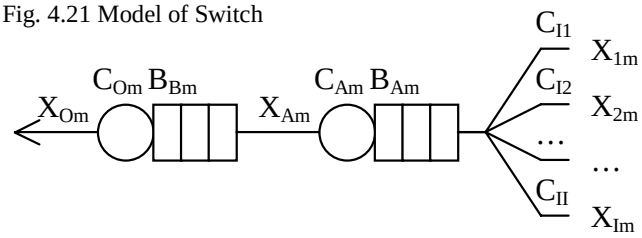
Therefore we have shown that given the minimum capacity conditions of the switch fabric (4.9) and (4.10), a single AQM at the output interface is capable of controlling the traffic to match it to the single bottleneck.

When the traffic is not in steady state (4.11) and (4.12) may be violated. Note that (4.12) cannot be violated without violating (4.11). The AQM always controls the congestion signal to steer the system towards $X_m(t) = u_m.C_m(t)$, and restore (4.11). By restoring (4.11), (4.12) is also restored. So, although the switch fabric output port capacity may become a second bottleneck in transient conditions, it is only necessary to performing congestion control on the one bottleneck.

4.3.5 Two Queue Model of CIOQ Switch

To simulate the switch shown in Fig. 4.20 in transient conditions, the model is simplified. In this section, we show that, for the purpose of analysing the amount of backlog in the input and output queues, the multi-queue switch is equivalent to a two queue network, as shown in Fig. 4.21.

Fig. 4.21 Model of Switch



The growth of an input interface queue that buffers traffic from input interface *n* destined for output interface *m* is the arrival rate into this queue $X_{nm}(t)$ minus the service rate of the queue. Since by (4.9) the only possible bottleneck in serving the *n* input queue is the switch fabric output port, the available capacity of this port is the service rate available for the *n* input interface queue. We will assume worst case conditions in our analysis. The least amount of capacity available at the switch fabric output port for an output interface *m* is when all other output interfaces on the same module are fully loaded:

$$C_{Min\,Bm} = C_{Bm} - \sum_{y \in OC_e, y \neq m} C_{Oy}$$

where *e* is the output module of interface *m*. Then the growth of the input *n* interface buffer with packets for the *m* output interface is:

$$\frac{dB_{nm}}{dt} = X_{nm} - (C_{Min\,Bm} - \sum_{k=1, k \neq m}^{I} X_{Akm}(t)) \;\; B_{nm} \geq 0$$

where $X_{Akm}(t)$ is the traffic rate from input queue *k* to output queue *m*. Let us sum the growth of all of the input queues with traffic destined for output interface *m* in all of the input modules:

$$\frac{dB_{Am}}{dt} = \sum_{n=1}^{I} \frac{dB_{nm}}{dt} \quad B_{Am}, B_{nm} \geq 0 \qquad (4.13)$$

$$= \sum_{n=1}^{I} X_{nm}(t) - (I \cdot C_{Min\,Bm} - (I-1) \cdot \sum_{k=1}^{I} X_{Akm}(t))$$

where $\frac{dB_{Am}}{dt}$ represents the growth rate of a virtual aggregate input queue for output interface *m*. Note that the total traffic from the input interfaces *n* destined for output interface *m*, $\sum_{k=1}^{I} X_{Akm}$, is at most the capacity of the switch-fabric output port available, $C_{Min\,Bm}$, assuming all other output interfaces on the output module are fully loaded:

$$\sum_{k=1}^{I} X_{Akm} = \min(C_{Min\,Bm}, \sum_{n=1}^{I} X_{nm})$$

then the growth rate of the virtual input queue for output interface *m*, and the growth rate of the m^{th} output queue is:

$$\frac{dB_{Am}}{dt} = \sum_{n=1}^{I} X_{nm} - C_{Min\,Bm} \qquad B_{Am} \geq 0$$

$$\frac{dB_{Bm}}{dt} = \sum_{k=1}^{I} X_{Akm} - C_{Om} \qquad B_{Bm} \geq 0$$

The dynamics of the total backlog in the switch for output interface *m* can now be expressed as the sum of the dynamics of two separate queues:

$$\frac{dB_m}{dt} = \frac{dB_{Am}}{dt} + \frac{dB_{Bm}}{dt} \qquad B_m \geq 0 \tag{4.14}$$

Note (4.14) only describes the amount of backlog in the switch, and not the ordering of packets, since the behaviour of the schedulers in the switch was not modelled.

4.3.6 CIOQ Switch Transient Behaviour

Given the two queue model of the switch established in the previous section, we will now study the dynamics of packet backlog in the input and output queues for the RB AQM switch. In steady state there will be no backlog in the switch, since RB AQM ensures that $X(t) < C(t)$. However, when sources turn on and off and due to the

ramping nature of TCP traffic, the traffic rate will inevitably have transients that may be above the capacity of the switch. These transients will be absorbed into the input and output queues. How big these queues need to be to keep packet loss below some value will depend on the arrival rate distribution that describes the transients. Generally, the bigger the queues are, the lower the packet loss probability. However a practical switch design is constrained to a maximum amount of memory *M* bits for the input and output buffers. In this section, we establish what fraction of *M* should be in the input queue and output queues to minimize the packet loss for a worst case transient condition and for realistic traffic.

When (4.11) is violated by a traffic surge, then the output interface is unable to forward all of the traffic arriving from the switch fabric and backlog builds in the output buffers. When (4.12) is violated by a traffic surge, then (4.11) is also violated, and the switch fabric is unable to forward all of the traffic arriving at the input interfaces of the switch, and backlog builds in both the input and output buffers.

Assuming all of the output interfaces of the module the j^{th} output interface is on are fully loaded, then the maximum possible arrival rate for the j^{th} output interface is:

$$X_{\max j} = \sum_{j=1}^{I} C_{Ij} - C_{\min Bj}$$

and the ratio of input to output backlog growth r_{max} in the worst case transient is:

$$r_{\max} = \frac{\frac{dB_{Aj}}{dt}}{\frac{dB_{Bj}}{dt}} = \frac{X_{\max j} - C_{\min Bj}}{C_{\min Bj} - C_{Oj}}$$

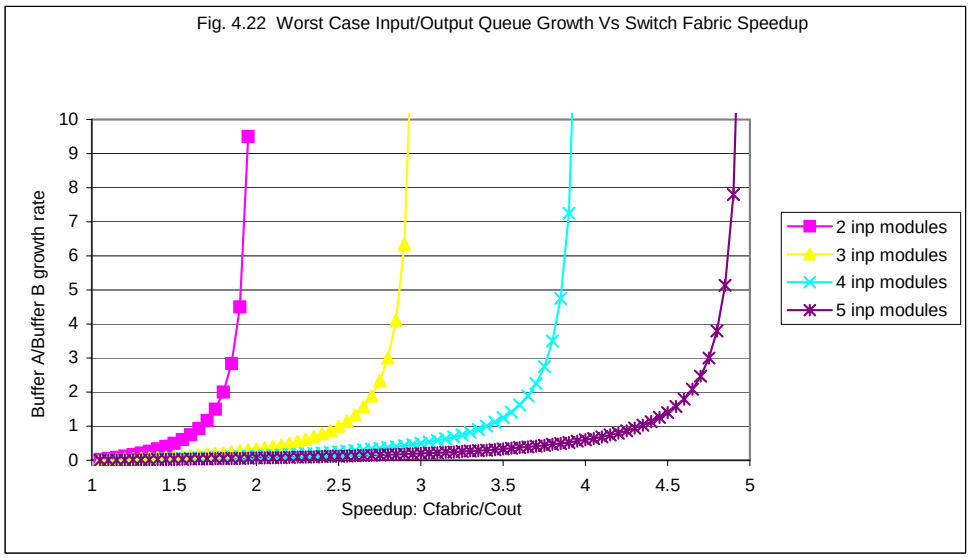
The metric r_{max} gives an upper bound on the rate of growth of the aggregate input queue compared to the output queue. If we have *M* bits of memory, the worst case transient can only be buffered for a limited time. To maximize this time, the portion of *M* allocated to the input and output queues should be in proportion to their growth

rate. So the worst case analysis tells us that we do not need to give all input queues more than $M \cdot (\frac{r_{\max}}{1+r_{\max}})$ bits of the total memory.

The measure $\frac{r_{\max}}{I}$ is the worst case growth rate of an individual input queue compared to the output queue when all input queues have the same capacity. Fig. 4.22 shows $\frac{r_{\max}}{I}$ for a switch with one output interface and from 2 to 5 input interfaces (I = 2,3,4,5) with a switch-fabric speedup factor from 1 to 5. For this example, the output capacity is C_{O1}=1 and the input capacity of each interface is C_{Ii}=1. Note that $\frac{r_{\max}}{I}$ is 1 or less for the full range of input interface numbers, once the speedup is greater than 2. This means that with a moderate speedup of 2, a reasonable memory allocation for this switch is to give an equal amount of memory, $\frac{M}{I+1}$, to each of the input and output queues.

Note that with real traffic there is a very low probability that this worst case transient occurs, so the growth of the input queue compared to the output queue is not so severe and not as much of the memory needs to be devoted to the input queues. In the next subsection, we simulate the RB AQM switch with realistic TCP traffic to investigate more realistic input and output backlog behaviour.

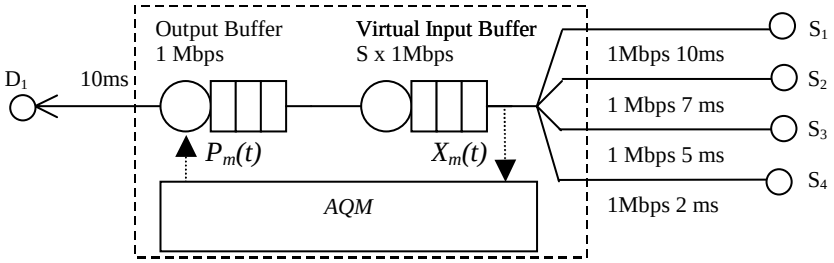
Fig. 4.22 Worst Case Input/Output Queue Growth Vs Switch Fabric Speedup



4.3.7 RB AQM CIOQ Switch Simulation

RB AQM and tail-drop switches were both simulated in two experiments with TCP traffic. The network topology for both experiments is defined in Fig. 4.23. The switch fabric speedup is *S*. The switch has four input interfaces connected to links with different propagation delays. Each source node S1 to S4 is capable of hosting an arbitrary number of TCP sessions. Each TCP session originates from a random source, S_1 to S_4 and terminates at D_1. When on, TCP sessions are saturated source. TCP sessions have uniform random duration fro 4 to 20 sec. The input and output queue sizes are 20 packets each.

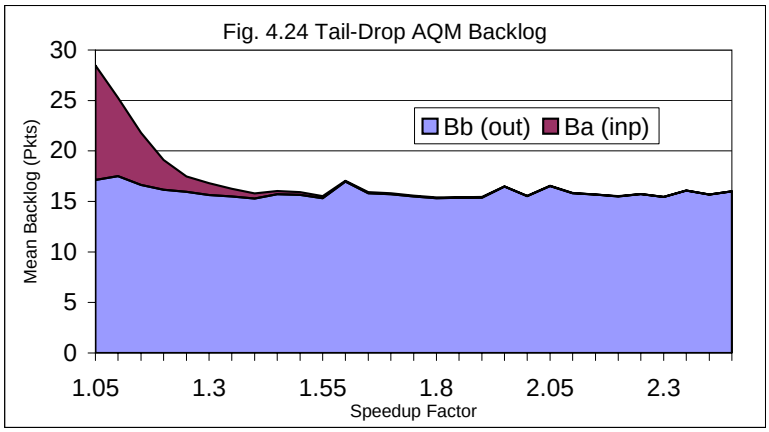
Fig. 4.23 Model of RB AQM Switch

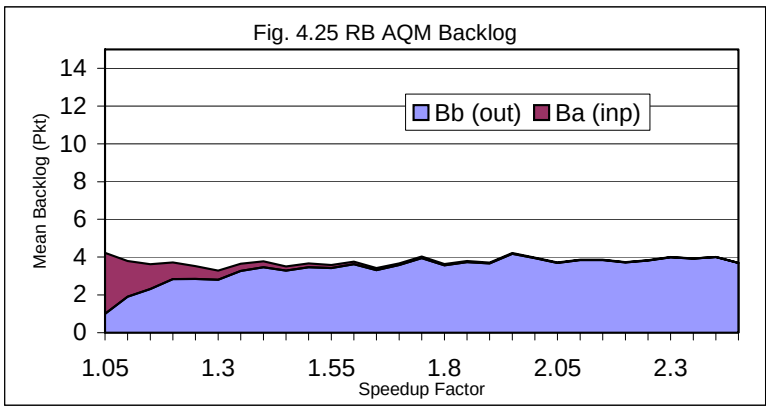


The particular RB AQM used in the switch is the GREEN AQM, as detailed in Section 3.3.1. For both experiments, $\alpha = 3\text{x}10^{-6}$, $\Delta P = 1\text{x}10^{-5}$, $u = 0.98$. All sources were ECN enabled.

4.3.7.1 Experiment 1: Speedup

RB AQM and tail-drop switches were simulated. The mean backlog for a range of switch fabric speedups *S* was measured, shown in Fig. 4.24. and 4.25. For each speedup, the mean backlog was taken over a 100s simulation period. During the simulation period 200 TCP sessions were start and stopped, with uniform random start times over the simulation period.



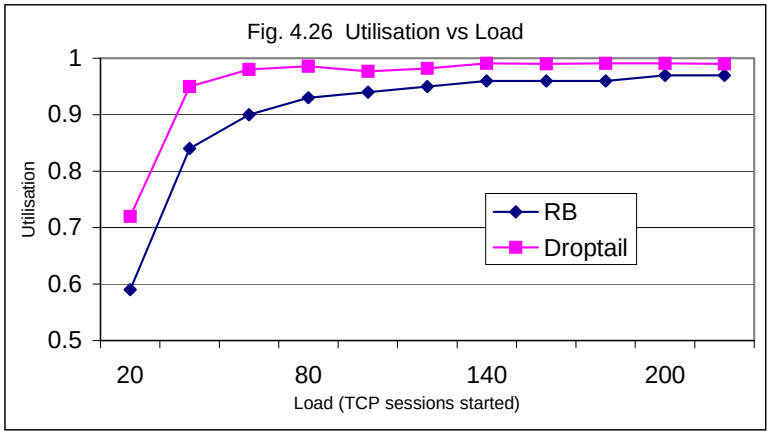


Notice that for the full range of speedup *S*, the RB AQM switch has substantially lower mean total backlog than the tail-drop switch. As discussed, the tail-drop switch maintains full buffers because it is unable to signal congestion until the buffer overflows. For both the tail-drop and RB AQM switches, a speedup of 2 results in almost all of the backlog being in the output queue. This means that for such speedups the output buffers should be larger than the input buffers and that any service differentiation for scheduling of packets needs only to be performed at the output queue, since this is where almost all queuing delay occurs.

4.3.7.2 Experiment 2: Load

RB AQM and tail-drop switches were simulated with speedup S=2. The mean backlog and output link utilization were measured for a range of traffic loads, shown

here in Fig. 4.26, 4.27 and 4.28. A total of *N* TCP sessions, where *N* = 20, 40 … 220, were start and stopped with uniform random start times over the simulation period.



The results show that the utilization of the RB AQM trial was slightly lower than that of the tail-drop and the backlog of the RB AQM trial was significantly lower. This agrees with the fundamental result of queuing theory that for a given random arrival process, reducing the mean backlog reduces utilization. The tail-drop queue maintains a very high utilization, which the RB AQM reduces slightly for a significant reduction in packet backlog. Unlike the tail-drop queue, the trade-off between utilization and backlog can be controlled by RB AQM, here with the u_m parameter. This allows the switch to maintain lower mean queuing delay, whilst having large buffer to absorb bursts.

Note that as the mean number of TCP connections is reduced, utilisation drops for both types of switch because there is not enough offered traffic.

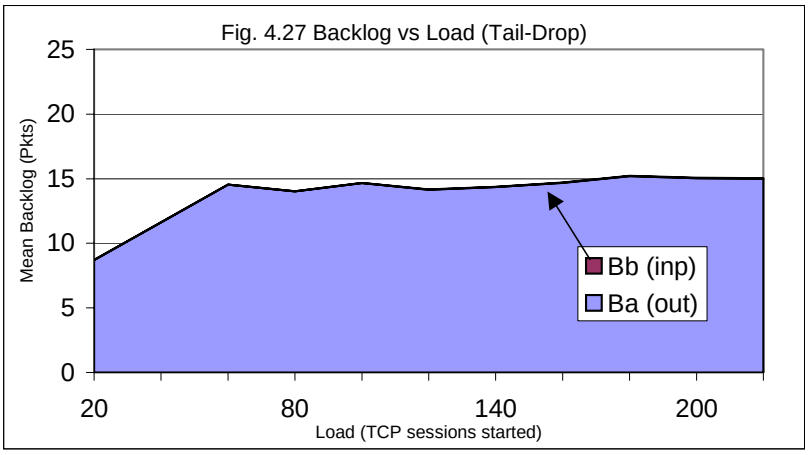


Fig. 4.27 Backlog vs Load (Tail-Drop)

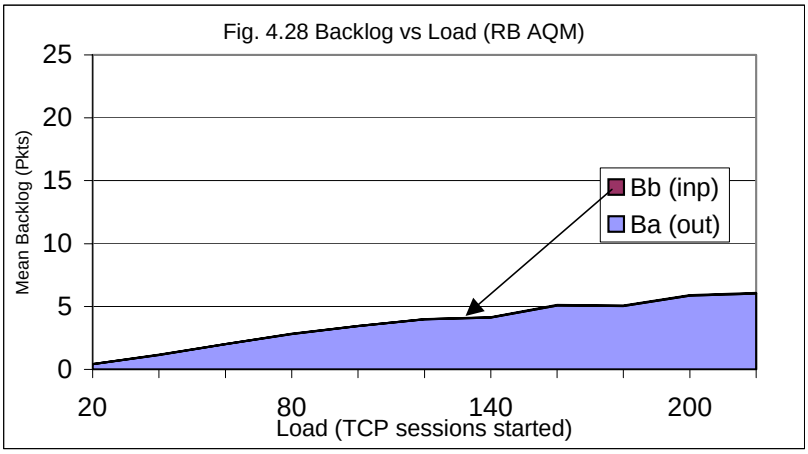


Fig. 4.28 Backlog vs Load (RB AQM)

4.3.8 Conclusion

In section 4.3, we presented an architecture for implementing RB AQM in a CIOQ switch. Only a single AQM controller per output port was shown to be required. The switch fabric speed-up required was also calculated. Simulations were performed to verify the proposed architecture and compare its performance to the existing tail-drop architecture. It was shown that RB AQM can achieve a low mean backlog whilst maintaining a high utilization.

4.4 Conclusion

The application of RB AQM controllers to multi-queue systems is necessitated by the need to bridge the advances in active-queue-management and the

practice of router and switch design. In this chapter, we presented and tested a framework for applying RB AQM control to DiffServ links and to CIOQ switches. There remains work to be done in this area by developing frameworks for applying RB AQM control to other multi-queue systems, for example; switches based on bus-architectures, such as PC based routers, or non-work-conserving schedulers.

CHAPTER

5

Chapter 5: Multi-Service Network Provisioning

5.1 Introduction

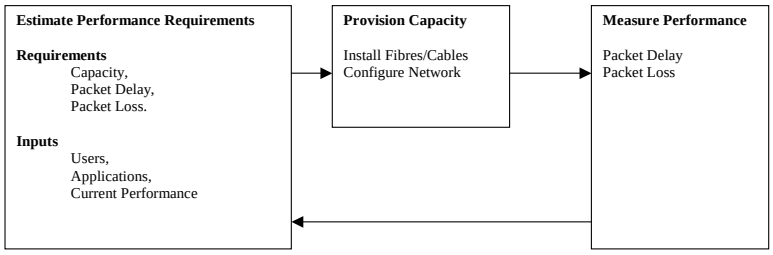
In previous chapters, we have developed load control techniques, which keep the backlog and delay inside the network low. As discussed in Chapter 2, enough capacity must still be provisioned so that the total source-application-layer to destination-application-layer delay is kept low. The work in this chapter extends Chapter 2, as we develop analytical tools for provisioning capacity on the same kind of multi-service links for which we previously developed congestion control techniques. The analytical tools developed here are motivated by the convergence of voice, video and data onto a unified network. By using the tools, links that carry data, voice and video can be provisioned with enough capacity to meet performance specifications.

5.1.1 Capacity Provisioning

Just as load control is a closed loop system which controls the source rates, then Capacity Provisioning can be thought of as a closed loop system which controls the network capacity. However, as the amount of capacity is controlled by laying fibre, installing cables, and configuring routers, it is not in general an autonomous system. The loop begins with establishing the performance requirements. These are based on the amount of users and the type of applications they required. The current performance level of the network is also used as a performance benchmark. Performance requirements include metrics such as Packet Delay, Packet Loss and Bandwidth. The network engineer then uses this data to determine the required network capacity. Once the network is operating, the actual performance is

measured. If the performance is not adequate, because not enough capacity was installed, or the number of users, or applications changed over time, then the capacity provisioning process begins again. Accurate analytical tools for estimating the amount of capacity are critical in this cycle, since they allow the network engineer to determine the amount of capacity to install, so that the correct amount of capacity can be installed in just one iteration of the process. The key to accurately estimating the capacity requirement, is being able to capture the packet generation process, which is driven by the application and user. Stochastic models are widely used for this. In this chapter, we will develop stochastic models for voice, video and data applications, and use them as a foundation for an analytical tool which determines packet delay performance from the traffic and capacity specifications of the link.

Fig. 5.1 Capacity Provisioning Process

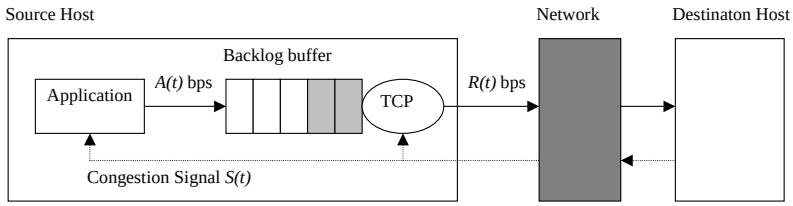


5.1.2 Relationship between Capacity Provisioning and Congestion Control

The way capacity provisioning and congestion control interact depends on the type of source which is generating the traffic. The origin of data for transmission is the application layer. Fig 5.2 illustrates the relationship between the application layer data generator and the transport control protocol packet transmitter. The application generates data at a rate *A(t)* bps, and the transport control protocol sends packets onto the network at *R(t)* bps. As *R(t)* is controlled by the congestion control system, it is limited by the available capacity on the network, however, *A(t)* is controlled by the application and may not be controlled by the network congestion level. As far as congestion control, the application layer can be classified into two categories 1) elastic and 2) inelastic. Elastic applications control the amount of data they generate for transmission based on the available capacity of the network. Inelastic applications

have a fixed amount of data to send regardless of the congestion level of the network. Examples of elastic applications include multi-layer video transmission, where the quality of the video is reduced if not enough network capacity is available. FTP is an example of an inelastic application, since FTP must transfer an entire file, regardless of the congestion level, and regardless of the time it takes to transmit.

Fig. 5.2 Application and Transport Control Protocol Structure



When the application data rate *A(t)* momentarily exceeds the network capacity between the source and destination hosts, congestion control determines how much of the backlog is stored at the source and how much is allowed to backlog inside the network. The backlog in the network is stored inside the buffers of switches and routers. RB AQMs can control the source rates far enough below the network capacity, so that no significant backlog can build inside the network. The backlog is kept at the source until capacity is available in the network. Tail-drop queues, on the other hand, allow the backlog to be stored in the network as well as the source, since they do not generate a congestion signal until the network queues are already full. As discussed previously, the problem with backlog being inside the network, is that it creates delay for all sources sharing the network, not just the ones generating packets above the capacity of the network. Even low bandwidth applications must suffer the queuing delay inside the network if network queues are allowed to fill.

As discussed, the total delay a packet experiences, from the time it is generated, and received by the destination application, includes both the delay inside the network and the source:

$$D_{total} = D_{source} + D_{network}$$

The network delay, $D_{network}$, consists of the propagation delay plus queuing delay inside the network and the source delay D_{source} is the queuing delay inside the source due to the backlog of packets generated by the application but not yet sent onto the network. Using RB AQM control, the total delay $D_{network}$ for both elastic and inelastic sources can be brought close to the propagation delay. However, congestion control can only keep D_{total} low for elastic applications. This is because elastic sources respond to network congestion and do not generate more data than can be carried by the network. However, for inelastic sources, congestion control cannot control D_{source} since the application generates data independently of the congestion level. Only provisioning enough capacity in the network to serve the application demand can ensure that the total delay for packets belonging to inelastic sources is also low.

In this chapter, we contribute to the tools that help carriers provision enough capacity in the network to keep packet delay low for a load of inelastic applications. We will focus on multi-service networks to extend the work in the previous chapter where we developed a congestion control architecture for a multi-class link. We will develop analytical tools for the performance evaluation of a DiffServ for a particular, but increasingly more common application of multi-class networks; to carry voice, video and data on a convergent network. Firstly, we will discuss some previous work in the field.

5.1.3 Previous Work

There are a number of previous studies of DiffServ and more generally, multi-service links. Literature in the field falls into a number of broad categories; Simulation studies of Diffserv performance, Analytical studies of DiffServ and fundamental analytical work in hybrid switching systems.

A large body of simulation and analytical studies investigate the performance of TCP in a DiffServ network. Our work is fundamentally different from this, because as discussed in Section 5.1.2, we look at the application performance for

inelastic applications. Studying TCP only gives information about the performance received by the transport layer, not the application layer. However, total packet delay experienced by the application layer is ultimately what the user perceives. The study by J.Rezende in [109] is an example of a simulation study of TCP performance. In [109], TCP is transmitted as Assured Forwarding class traffic, and its performance is investigated in the presence of other background traffic.

In [71], the differentiation of service quality for TCP traffic in different sub-classes of the Assured Forward class is studied. The work analytically shows that Diffserv can achieve differentiated bandwidth allocation to different classes of TCP traffic.

Other DiffServ simulation studies investigate the performance of higher priority traffic in the presence of lower priority TCP traffic. In [130] the authors perform a simulation study of the performance of voice traffic in the presence of best-effort traffic. Two models of voice traffic are explored, CBR and ON-OFF. While [130] focuses on the performance of the voice traffic in the presence of applications generating lower priority data traffic, we are actually concerned with the performance of these data applications, in the presence of voice or video CBR traffic.

Unlike any work known to us in modelling DiffServ, we model the link as a single server queue in a Markovian environment and analyse the queuing performance of data packets by matrix geometric methods. The field of matrix geometric methods has been largely developed by the work of Marcel Neuts. We use iterative numerical methods to solve the underlying system.

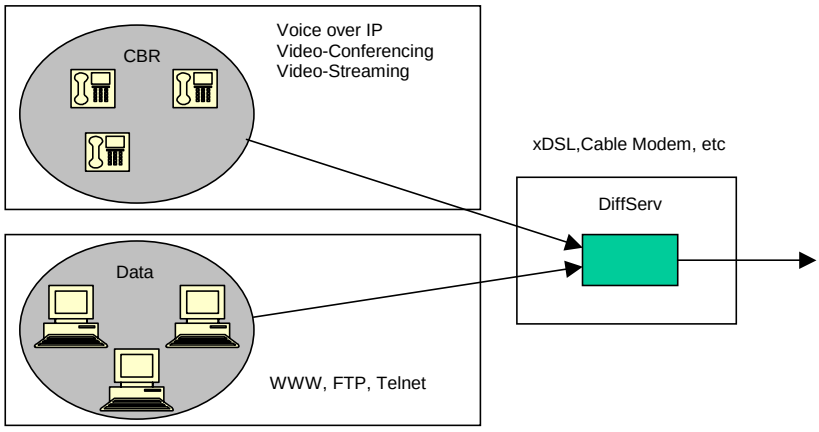
The mathematical formulation of the problem which we use is most similar to that in [131] by M.Zukerman and P. Kirton (also refer to [104] by Potter and Zukerman). The application studied there is a B-ISDN switching system, which, like DiffServ, is a hybrid switching system, with two types of traffic, burst data and CBR connection traffic. In this work, the data packet performance in the presence of higher priority CBR streams is studied. Like the work in this chapter, it is also formulated as a queue in a Markovian Environment. However, the paper does not provide a method for readily obtaining numerical solutions to the system. The work presented in this chapter, developed in collaboration with Beatrice Meini of the

University of Pisa Italy, provides a method for evaluating a similar system with quadratic convergence. This allows a larger system, with more possible connections, and more possible connection types to be evaluated than previously.

5.1.4 Provisioning for Voice, Video and Data Convergence.

The convergence of data networks with voice and video networks is an important application and driver of the development of multi-class links. We will analyse a particularly common scenario as shown in Fig 5.3, where DiffServ is used to provide both voice/video and data services over one access link into a home or office. DiffServ allows voice and video applications to receive predicable performance in the presence of TCP data traffic. Voice is typically assigned to a higher priority class than TCP and receives capacity in preference to TCP traffic. We will analyse the interaction that the presence of higher priority voice or video traffic has on the performance of TCP data traffic generated by inelastic applications in a DiffServ link. To do this, we must model the applications that generate voice, video and data traffic.

Fig. 5.3 DiffServ Application



We assume that voice connections have a constant bit rate CBR. Although multi-rate codecs exist for voice, sending voice as an uncompressed 64kbps stream is favourable, as compression codecs have a high latency [44].

Video can be encoded as either a Variable Bit Rate (VBR) or Constant Bit Rate (CBR) stream. Given that a video is encoded with the same amount of bits for both methods, VBR encoding can maintain a constant quality, whereas CBR can maintain a constant bit rate with variable quality. VBR video exhibits significant burstiness on multiple time scales, due to the frame and scene structure of the footage [34]. CBR encoding significantly simplifies the allocation of “disk, memory and network resources”[34] [34]. CBR encoding for video is therefore typically used for streaming and conferencing applications, as it makes the capacity requirements more predictable and allows applications to reserve capacity efficiently. Our model uses CBR for video as well.

We will develop an analytical model of a two class DiffServ link, where one class consists of CBR connections that carry high priority voice or video sessions, and a second class exists for lower priority data. We will analyse the performance of data traffic in the presence of connection oriented CBR traffic sharing the same link.

5.2 Two-Class System Description

In this section, we give a description of the system that we will be modelling. We will remind the reader of the essential features of the DiffServ architecture relevant to our performance analysis, and we describe the type of traffic we are modelling. For a detailed description of DiffServ, the reader is referred to [91].

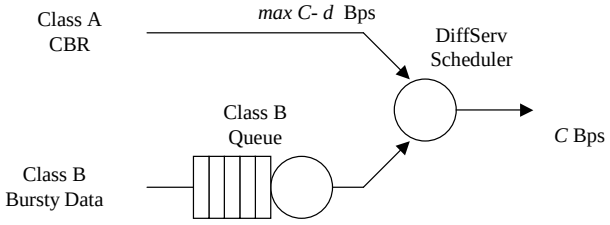
As discussed previously, DiffServ classifies packets into classes by a 6-bit pattern in the IP header, called DSCP. Applications requiring low-loss, low-latency, low-jitter and assured bandwidth service generally send data as Expedited Forwarding EF class packets. The EF class is typically used for delay and loss sensitive applications such as voice over IP (VoIP) and Video sessions. The Assured forwarding (AF) class offers a lower priority service, and itself is subdivided into a number of subclasses (Premium, Gold, Silver, Bronze). Generally AF carries best effort TCP data, such as HTTP and FTP traffic.

Although DiffServ has a large number of classes defined, we are only modelling the link in the presence of two classes of traffic: 1) voice and video traffic and 2) best effort data traffic (e.g. TCP). Let us denote the two classes as A and B.

Class A traffic uses the EF class on the DiffServ link, and class B traffic uses any of the AF classes.

We assume that the DiffServ link uses a priority scheduler, which transmits any available Class A packets ahead of Class B packets, so long as there is a minimum capacity available for Class B packets. Fig 5.4 depicts our model, where the access link has capacity C *Bps*, and a priority scheduler gives priority to Class A packets ahead of Class B packets. Class B packets which cannot be transmitted immediately are queued. We are concerned specifically with the statistics of this class B queue for data traffic.

Fig. 5.4 Two Class DiffServ Scheduler



Let the capacity required by a single class A voice or video CBR connection be *s* Mbps. We will assume that the total link capacity *C* is divided into *c* units of capacity *s*, *s=C/c*. Of the *c* units of capacity, *d* units are reserved exclusively for Class B data packets. Therefore only *c-d* capacity units are available for Class A CBR connections.

Class A packets are generated by hosts which establish CBR connections. The reserved capacity for Class B packets is representative of connection-admission control. When connection admission control is used, such as RSVP, the number of Class A connections is limited to *c-d*, with additional connections being blocked.

We will model the class A and B traffic as stochastic processes. These models are supposed to capture the behaviour of the applications which generate the packets. The class B data application is modelled as a two state Markov modulated Poisson process (MMPP). As illustrated in Fig. 5.5, the model of class B traffic captures the aggregate behaviour of all of the hosts transmitting class B packets.

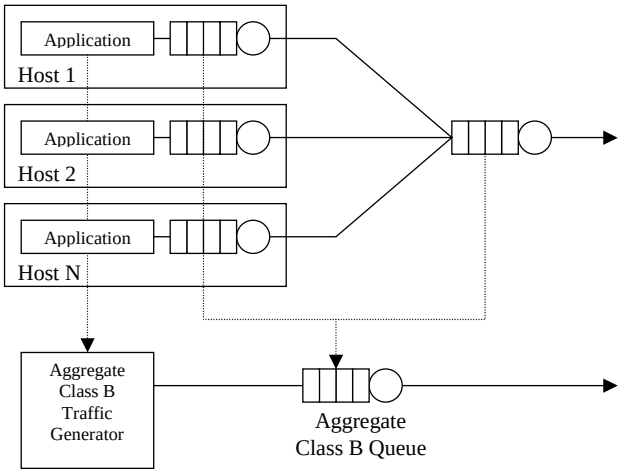
Class A packets are CBR connections, whose establishment and hold times are an exponential random process.

The queuing delay which we are concerned with in this chapter is the delay in all the queues experienced by a Class B packet from the time the packet is generated to the time it is serviced by the DiffServ link. To obtain the packet delay distribution of Class B packets, we model the system as a single server queue. As shown in Fig. 5.5, in reality, the queuing occurs in two places 1) at each host's TCP transmission queue, 2) at the aggregate class B queue in the DiffServ link. However, since (1) TCP guarantees delivery (2) TCP utilizes available bandwidth[3] at the bottleneck link, once a packet is generated, it is effectively awaiting service until capacity is available at the DiffServ link. Since all the hosts compete for the same capacity, the wait for service can be represented as a single server queue (SSQ) process that all Class B packets from all hosts are in. The bandwidth available to Class B packets in this SSQ is dependent upon the number of Class A CBR sessions active on the DiffServ link.

The SSQ is modelled as a queue in a Markovian environment [42, 90] to reflect stochastic nature of the capacity available to the Class B packets given the effect of Class A CBR connection loading. We use matrix analytic methods [90] to obtain the numerical results of the delay of Class B packets.

[3] Our assumption that TCP maintains a high utilisation of the DiffServ link is mild. TCP is known to maintain a high utilisation of network capacity, as evidenced by work in previous Chapters where utilisation of >97% in most scenarios is experienced.

Fig. 5.5 Aggregate Class B Model



The aim of this work is to provide an analytical solution about this mean queuing performance of class B packets, given the parameters of Class A and Class B traffic. In the next section, we will describe the analytic model, and in the section following, we will present the performance evaluation of the two class link as well as a numerical example.

5.3 Traffic Model

5.3.1 Class B Data Packet generation model

The generation of Class B packets by an aggregate of applications for transfer across the DiffServ Link is modelled using a two state Markov Modulated Poisson Process (MMPP) [52]. We will now discuss how the MMPP process can capture the bursty nature of data traffic.

A two state MMPP is an alternating Markovian process with two packet generation states, where the generation process in generation state m is a Poisson process with rate: λ_m, m=0,1. The sojourn time in each generation state is exponentially distributed with the mean sojourn time in generation state 0 and 1 being r_0^{-1} and r_1^{-1} respectively. The values of the mean and the variance of MMPP traffic are given in [52]. Let N_t be the amount of work generated during time interval $(0, t)$. For MMPP, the mean of N_t, denoted $E[N_t]$, is:

$$E[N_t] = \frac{\lambda_0 r_1 + \lambda_1 r_0}{r_0 + r_1} \cdot t$$

and the variance of N_t, denoted V_t, is:

$$V_t = E[N_t] \cdot \left(1 + \frac{2(\lambda_0 - \lambda_1)^2 r_0 r_1}{(r_0 + r_1)^2 (\lambda_0 r_1 + \lambda_1 r_0)} - \frac{2(\lambda_0 - \lambda_1)^2 r_0 r_1}{(r_0 + r_1)^3 (\lambda_0 r_1 + \lambda_1 r_0) \cdot t} \left(1 - e^{-(r_0 + r_1)t}\right) \right)$$

It is well-known that the autocorrelation of the arrival process has significant effect on queuing performance. It is also well-known that real traffic exhibits long range dependence (LRD) [3]. However, since buffer size is limited so is the time period over which autocorrelation has effect. The larger the buffer size, the longer is the time period over which autocorrelation affects queuing performance. If the buffer size is equal to zero, autocorrelations has no effect on performance.

If we accept the view that for a given buffer size, the shape of autocorrelation curve, from a certain point onwards, does not affect queuing performance, we can use the MMPP, which is a short range dependent (SRD) process, to model LRD real traffic for the purpose of queuing performance evaluation.

For that purpose, we will consider the variance time curve of MMPP. In Fig. 5.7, we plot four variance time curves with different parameter sets of MMPP traffic. By comparing the four presented curves, the critical time interval of the traffic and the slopes of the curve within the critical interval can be controlled by the parameters of MMPP. For curves (a), (b) and (c), we obtained SRD traffic with different critical time intervals. This means that it is possible to use a simple model such as MMPP model, and fit its parameters, such that over a given finite simulation period, the traffic is indistinguishable from LRD traffic.

Fig. 5.6. Parameters of the MMPP traffic in Fig. 5.7

	(a)	(b)	(c)	(d)
λ_0	0.01	0.01	0.01	0.01
λ_1	1	0.1	0.05	0.05
r_0	10^{-8}	10^{-8}	10^{-5}	0.5
r_1	$2{\cdot}10^{-9}$	$2{\cdot}10^{-9}$	10^{-6}	0.005

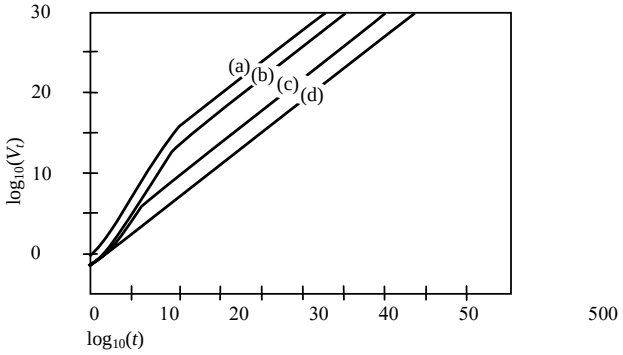


Fig. 5.7. The Variance Time Curve of MMPP

5.3.2 Class A Voice/Video CBR packet generation model

In this section, we describe how the voice or video applications which generate class A connections are modelled. A Class A CBR stream is allocated capacity for the duration of the connection. The CBR stream sends packets at a constant rate for the duration of the connection. Admission of a CBR stream decreases the available capacity for the Class B data packets and the completion of a CBR connection increases the capacity available for the data packets.

The admission and completion of the CBR connections is modelled as an *M/M/k/k* process, where *k* is the maximum number of connections supported by the system. The CBR connection arrival process is Poisson with parameter λ_v and the connection holding time is assumed exponential with mean $1/\mu_v$. This is consistent with well accepted models for connection oriented traffic in telephony traffic engineering [131].

5.4 Queuing Analysis

We have modelled DiffServ as a queue in a Markovian environment, and we follow Neuts' analysis of such a queuing model as described in [90] pages 254-264.

The infinitesimal generator is first introduced here to describe the system, then, we review Neuts' solution as applied to our queuing problem, and finally we show how the rate matrix *R*, an intermediate variable required for Neuts' solution, is computed.

5.4.1 Infinitesimal Generator

The state of the system under consideration is denoted by the three dimensional vector (*i,j,m*) where *i* is the number of Class B data packets in the queue (including the one in service), *j* is the number of capacity units available for data packets, *m* is the arrival state (m takes the values 0 and 1). The Class B service rate is always equal to $j\mu$, where μ is the service rate provided by one capacity unit, and $j = d, d+1, d+2, \ldots, c$. The packet data arrival rate is λ_m, $m = 0, 1$. All the possible state transitions are presented in Fig. 5.8. Hence, Fig. 5.8 defines the infinitesimal generator matrix *G*. The sum of the entries in each row of *G* is 0. Let $\hat{h}_{i,j,m}$ be the probability of being in state (*i, j, m*), then the vector $\hat{h}$ is:

$$(\hat{h}_{0,0,0}, \hat{h}_{0,0,1}, \hat{h}_{0,1,0}, \hat{h}_{0,1,1}, \ldots,$$
$$\hat{h}_{0,c,1}, \hat{h}_{1,0,0}, \hat{h}_{1,0,1}, \hat{h}_{1,1,0}, \hat{h}_{1,1,1}, \ldots,$$
$$\hat{h}_{1,c,1}, \hat{h}_{2,0,0}, \hat{h}_{2,0,1}, \hat{h}_{2,1,0}, \hat{h}_{2,1,1}, \ldots).$$

The state transition balance equation is then $\hat{h}G = 0$. The steady state queue size distribution x_i is related to $\hat{h}$ as such $x_i = \sum_m \sum_j \hat{h}_{i,j,m}$. We now obtain x_i by another approach, using Neuts' analysis.

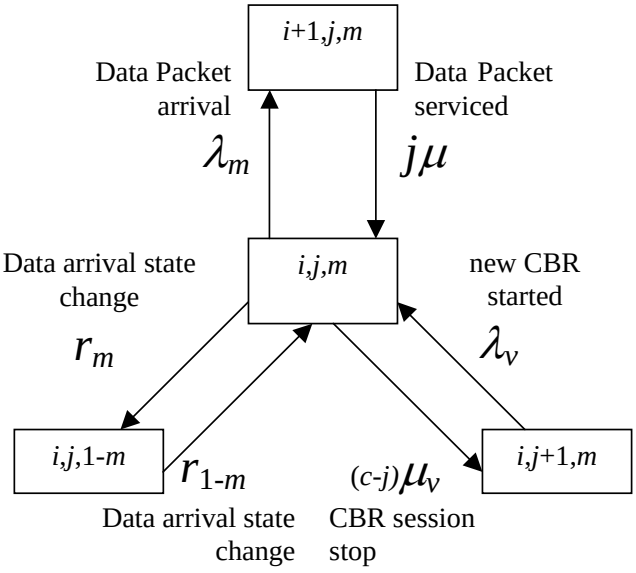


Fig. 5.8 State Transition Diagram

5.5 Neuts' solution

5.5.1 Reformulating problem

The steady state queue size distribution vector *x* can be solved analytically by considering the Class B data packet SSQ's parameters being driven by a Markovian environment. The Markovian environment is comprised of two processes, the CBR connection arrivals and departures as well as the transition between Class B data arrival states 0 and 1. These processes are independent, and determine the state of the environment *j* and *m*. The transition probability matrix of the Markovian environment is denoted *Q* and it is shown in Fig. 5.9.

The Markovian random environment is the stochastic process that determines the number of CBR connections in the system and the packet data arrival state. As discussed, the packet data service rate fluctuates based on the CBR connections in the system. For each state of the environment, there is an appropriate data service rate in vector μ and packet data arrival rate in vector λ. The packet data service rates in each state of the environment (*j*, *m*) are:

$$\mu_{j,m} = \mu_j = j\mu, \qquad m=0,1 \text{ and } j=d, d+1, d+2, ..., c.$$

Let vector μ be defined by

$$\mu = (d\mu, (d+1)\mu, (d+2)\mu, \ldots, c\mu).$$

The packet data arrival rates in each state of the environment (j, m) are:

$$\lambda_{j,m} = \lambda_m, \qquad m=0,1 \text{ and } j=d, d+1, d+2, ..., c.$$

Let vector λ be defined by

$$\lambda = (\lambda_0, \lambda_1, \lambda_0, \lambda_1, \lambda_0, \lambda_1, \ldots. \lambda_1).$$

	d,0	d,1	d+1,0	d+1,1	…	c,1
d,0	$-r_0-(c-d)\mu_v$	r_0	$(c-d)\mu_v$			
d,1	r_1	$-r_1-(c-d)\mu_v$		$(c-d)\mu_v$		
d+1,0	λ_v		$-\lambda_v-r_0-(c-d-1)\mu_v$	r_0		
d+1,1		λ_v	r_1	$-\lambda_v-r_1-(c-d-1)\mu_v$		
…						
c,1						$-\lambda_v-r_1$

Fig. 5.9 The Matrix Q

For example, if j=10 and m=1, with c=22 and d=1, there are 22 capacity units with 1 reserved for Class B packets only, and there are 12 CBR connections in the system as there are 10 capacity units available for Class B packets. The Class B arrival state is 1. The data service process then has parameter 10μ (since 10 capacity units are allocated to serving Class B data) and the Class B data arrival process has parameter λ_1.

Assuming the queue is stable (mean arrival rate < mean service rate), to determine the queuing performance of the DiffServ system, the stationary probability vector $x=(x_0,x_1,x_2,\ldots)$, which describes the probability distribution of the queue size, needs to be determined.

Using Neuts' formalisation of the M/M/1 queue in a Markovian environment [90], this problem is translated into finding the minimal solution for the matrix *R*, which satisfies the matrix equation:

$$R^2\Delta(\mu)+R(Q-\Delta(\lambda+\mu))+\Delta(\lambda)=0$$

where *Q* is the transition probability matrix of the Markovian environment, and $\Delta(z)$ the diagonal matrix of the vector *z*. The matrix R can be evaluated using a cyclic reduction algorithm described in the next subsection. The stationary probability vector x of the stable queue is given by:

$$x_k=\pi(I-R)R^k \quad for\ k\geq 0$$

where π is the stationary probability vector of *Q* (Eq. (6.2.5) from [90]). The vector π is given by solving $\pi\cdot Q=0$ by, for example, successive relaxation.

5.4.3 Computation of the rate matrix R

In this subsection, we describe the algorithm for the computation of the rate matrix R. The matrix R is the minimal nonnegative solution of the matrix equation

$$A_0+A_1R+A_2R^2=0$$

where $A_0=\Delta(\mu)$, $A_1=Q-\Delta(\lambda+\mu)$, $A_2=\Delta(\lambda)$. Here, minimal nonnegative solution means that for any other nonnegative solution $\hat{R}$, the matrix *R* is entry-wise less than the matrix $\hat{R}$.

This effective numerical method that works in the case where the associated M/M/1 queue is transient or positive recurrent [90]. The method is based on the use of cyclic reduction and is extensively discussed in the B.Meini's earlier paper [12] and later in [13] [14] [51]. Here we will recall the algorithm and its convergence properties and we refer the reader to [12] [14] for details and proofs.

This method has nice properties, namely it is quadratically convergent, that is the approximation error e_j at the j-th step is such that $e_j \le c\sigma^{2^j}$ for suitable constants c>0 and 0<σ<1, and has a very low computational cost. Moreover all the computations involved in this scheme are numerically stable, since they consist of sums and products of nonnegative matrices and inversions of M-matrices (see [12]). The algorithm consists of generating the four sequences of matrices $\{A_0^{(j)}\}_{j\ge 0}$, $\{A_1^{(j)}\}_{j\ge 0}$, $\{A_2^{(j)}\}_{j\ge 0}$, $\{\hat{A}^{(j)}\}_{j\ge 0}$ according to the following recurrences:

$$\begin{aligned}
A_0^{(j+1)} &= -A_0^{(j)}\left(A_1^{(j)}\right)^{-1}A_0^{(j)}\\
A_1^{(j+1)} &= A_1^{(j)} - A_0^{(j)}\left(A_1^{(j)}\right)^{-1}A_2^{(j)} - A_2^{(j)}\left(A_1^{(j)}\right)^{-1}A_0^{(j)}\\
A_2^{(j+1)} &= -A_2^{(j)}\left(A_1^{(j)}\right)^{-1}A_2^{(j)}\\
\hat{A}^{(j+1)} &= \hat{A}^{(j)} - A_2^{(j)}\left(A_1^{(j)}\right)^{-1}A_0^{(j)},\ j\ge 0,\\
A_0^{(0)} &= A_0,\ A_1^{(0)} = A_1,\ A_2^{(0)} = A_2,\ \hat{A}^{(0)} = A_1.
\end{aligned} \tag{5.1}$$

In [14] [12] it is shown that these sequences of matrices are such that:

1. for any $j\ge 0$ it holds $\hat{A}^{(j)}R + A_2^{(j)}R^{2^j+1} = -A_0$;
2. if the associated M/M/1 queue is positive recurrent, then the sequences $\{A_0^{(j)}\}_{j\ge 0}$ converges in norm to zero as σ^{2^j}, where $\sigma = \max\{|\alpha|:\det(A_0 + \alpha A_1 + \alpha^2 A_2) = 0,\ \alpha\in C, |\alpha|<1\}$ is the largest modulus eigenvalue of R;
3. if the associated M/M/1 queue is transient, then the sequence $\{A_2^{(j)}\}_{j\ge 0}$ converges in norm to zero as σ^{2^j}, where $\sigma = 1/\min\{|\alpha|:\det(A_2 + \alpha A_1 + \alpha^2 A_0) = 0,\ \alpha\in C, |\alpha|>1\}$;
4. the sequence $\left\{\left(\hat{A}^{(j)}\right)^{-1}A_2^{(j)}R^{2^j+1}\right\}_{j\ge 0}$ converges in norm to zero as σ^{2^j}, where $0<\sigma<1$ is one of the two values defined above.

From these properties, an approximation of R is given by $-\left(\hat{A}^{(j)}\right)^{-1}A_0$, for a moderately large value of j. Moreover, in order to stop the computation, we check the minimum value between the norms of $A_0^{(j)}$ and $A_2^{(j)}$: if this value is smaller than a threshold value ε, then $-\left(\hat{A}^{(j)}\right)^{-1}A_0$ will be an approximation of R. The resulting algorithm can be synthesized by the following scheme:

Algorithm: Cyclic reduction for computing R

Input The matrices A_0, A_1, A_2 and an error bound $\varepsilon > 0$ for the stopping condition.

Output An approximation $\tilde{R}$ of R.

Computation

1. Set $j = 0$, $A_0^{(0)} = A_0$, $A_1^{(0)} = A_1$, $A_2^{(0)} = A_2$, $\hat{A}^{(0)} = A_1$.

2. Compute $A_0^{(j+1)}$, $A_1^{(j+1)}$, $A_2^{(j+1)}$, $\hat{A}^{(j+1)}$ by means of (1) and $r = \min\left\{\left\|A_0^{(j+1)}\right\|, \left\|A_2^{(j+1)}\right\|\right\}$, where, for a matrix $B = (b_{h,k})_{h,k=1,\ldots,n}$, $\|B\| = \max_{h=1,K,n} \sum_{k=1}^{n} \left|b_{h,k}\right|$.

3. If $r \geq \varepsilon$ set $j = j + 1$ and go back to step 2; otherwise, set $\tilde{R} = -\left(\hat{A}^{(j+1)}\right)^{-1} A_0$.

5.4.3 Multiple CBR applications

So far we have considered a model of DiffServ where there is only one connection rate for CBR connections. However, networks may have a more heterogenous application mix present, which include CBR voice over IP sessions as well as CBR video sessions and perhaps other CBR applications. In this section, we will describe how the original model can be extended to include a variety of CBR connection processes.

We have been considering a link of capacity C Mbps, which is divided into c units of capacity of s Mbps each. We will continue the constraint that a CBR rate can be described as an integer multiple of the basic s capacity unit. We can describe an arbitrary CBR process i, by its transmission rate k_i.s Mbps, mean holding time $1/\mu_i$ and arrival parameter λ_i. Given that a system has N different CBR processes of this type, we can define a new Markovian environment Q matrix of $N + 1$ dimensions with a state vector such as:

$$(j_1, j_2, \ldots, j_N, m)$$

where, the state j_i, a little cryptically, describes the amount of capacity units not used by the connections of CBR process *i*. If we consider u_i to be the number of type *i* CBR processes being served, then $c\text{-}u_i k_i = j_i$. Note that the total capacity used by all of the CBR processes must leave *d* capacity units reserved for the MMPP data process:

$$c - \sum_{i=1..N} u_i k_i \geq d$$

This constraint limits the number of possible assignments of capacity to the different CBR processes. By constructing the new Q matrix, the problem of multiple CBR connection types can be solved by the same approach as described previously for single CBR connection problem, however with increased state space.

We can determine the upper-bound, *S*, on the number of states in the Markovian environment matrix Q, that *N* CBR connections introduces by:

$$S = \left(\frac{c-d}{k_1}\right) \cdot \left(\frac{c-d}{k_2}\right) \cdot \left(\frac{c-d}{k_3}\right) \cdot \Lambda \cdot \left(\frac{c-d}{k_N}\right)$$

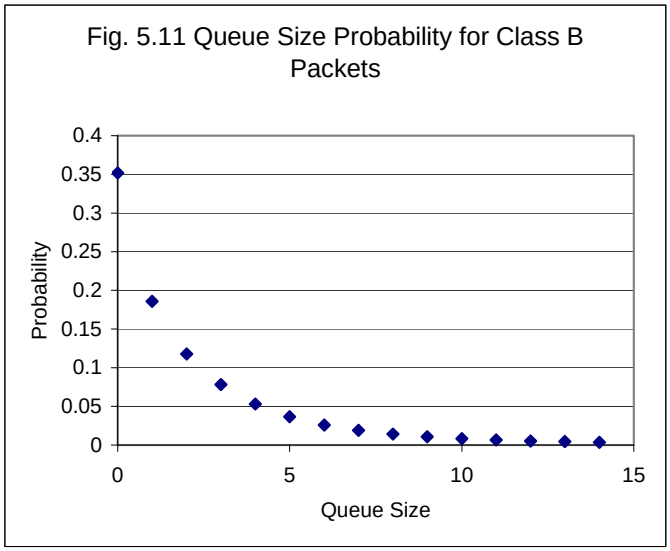
Note that when N is 1, and $k_1 = 1$, as in the analysis formulated in the previous sections, the number of states is just $S = c - d$. The state space grows quickly with *N*. However, since it may only be necessary to consider 2 or 3 different CBR application types on the DiffServ link, the problem remains tractable for useful scenarios.

5.5 Simulation Results

To illustrate the analytical method, we will perform an example numerical analysis. We will only consider the case of 1 CBR connection type. The link capacity is 24 Mbps, with 1 Mbps reserved for Class B packets only. We model a videoconferencing application that uses 1Mbps connections, so that each Class A CBR is 1 Mbps. The mean duration of a CBR connection is 3 minutes. The Class A (CBR connection) load is chosen so that the capacity available to the CBR connections (23 Mbps) has a utilization of 40%. All of the parameters are listed in Fig. 5.10. The queue size probability distribution is shown in Fig. 5.11.

Fig 5.10 Example Model Parameters

Parameter	Value	Meaning
λ_0	10 Mbps	Class B state 0 packet generation rate
λ_1	1 Mbps	Class B state 1 packet generation rate
r_0	0.1	State 0 to 1 transition rate
r_1	1.0	State 1 to 0 transition rate
μ	1 Mbps	Mean data service rate per unit of capacity
c	24	Total capacity units of link
d	1	Capacity units reserved for Class B data only
μ_v	1/(3*60)	1/(CBR connection hold time)
λ_v	$0.4 \cdot (c-d) \cdot \mu_v$	CBR connection establishment rate



The Class B queue stationary probability vector x is a stem for a variety of other performance results about the system. Further results such as packet delay statistics for class B packets, and delay as a function of system utilization can easily be obtained from the stationary probability vector x.

The mean Class B data packet delay is found using Little's law, where the Packet delay is the time from the generation of the packet to the time the last bit is sent. The mean delay is obtained as follows:

$$mean\ class\ B\ packet\ \ arrival\ rate = \frac{\lambda_0 r_1 + \lambda_1 r_0}{r_0 + r_1}$$

$$mean\ queue\ \ size = \sum_{i=1}^{i=\infty} x_i \cdot i$$

$$mean\ class\ B\ delay = \frac{mean\ queue\ size}{mean\ \ packet\ arrival\ rate}.$$

The probability of the backlog exceeding l packets can easily be obtained from x, and such a results is useful in dimensioning the buffer sizes:

$$\Pr(x > l) = 1 - \sum_{i=1}^{i=l} x_i$$

5.6 Conclusion

The focus of this chapter was to explore the capacity provisioning dimension of the congestion control problem. We developed an analytical tool for computing the performance of a DiffServ link, which aids in provisioning capacity to achieve the desired level of service for data services in the presence of higher priority voice or video traffic. Matrix geometric techniques were used to obtain queuing performance results.

The analysis presented in this section allows the network engineer to calculate the expected packet delay for data applications given a model of the applications sharing the link. By being able to predict the performance of the DiffServ link under certain load of applications, the network engineer can evaluate the performance expected, and decide on a) the network capacity required b) the minimum reservation of bandwidth for class B data packets, to meet the quality of service requirements. Dimensioning the links in the network to meet the demand is necessary to control delay for inelastic applications.

The convergence of Voice and Data makes the accurate provisioning of capacity on a DiffServ access link an important problem. DiffServ can only improve the performance of applications if there is adequate capacity provisioned in each service level to meet the performance requirements.

CHAPTER 6

Chapter 6: MaxNet, A New Congestion Control Architecture

6.1 Introduction

In this chapter, we will introduce a new model of performing congestion control on a source-flow controlled best-effort network. We will redefine how congestion information is signalled. In the previous chapters, we have developed congestion control techniques for the Internet, where congestion is signalled either by packet dropping or ECN marking by each link on the end-to-end path. We will now explore the structural constraints of the Internet congestion signalling model and develop a new architecture for signalling congestion on an Internet-like network that we call MaxNet. MaxNet differs from Internet because source rates are controlled only by the most severely bottlenecked link on the end-to-end source to destination path.

The Internet communicates the congestion signal one bit at a time by packet dropping or marking ECN capable packets. The origins of this method of communicating congestion information are rooted in the fact that when the arriving traffic to a switch or router is greater than the outgoing capacity, the buffers will eventually overflow and a portion of the arriving packets will be lost. The rate of the individual packet loss events that the sources detect is interpreted as the congestion signal, which is a measure of the congestion of a network path. The addition of ECN abstracted this idea by substituting packet loss by packet marking. MaxNet abstracts the idea of congestion signal further and allows each link to communicate an arbitrary number of bits of congestion state.

MaxNet redefines how congestion signaling is performed and results in unique fairness and scaling properties. MaxNet is in keeping with the very successful Internet philosophy of having a simple network infrastructure. As stated in [115] "Providing a minimalist service model and having the 'stateless waist' in the protocol hourglass allows the Internet to scale with both the size of the network and heterogeneous applications and technologies. Together, they are two of the most important technical reasons behind the success of the Internet." MaxNet is an Internet like network because like the Internet it is a stateless network, where the only connection state is kept in the source and destination hosts, and sources transmit according to their utility functions. Connection state only exists in source and destination hosts and the network keeps no information about the relationship between the packets it transmits.

In this chapter, we will show that MaxNet is able to achieve Max-Min fairness. Although the scope of MaxNet goes beyond Max-Min fairness, we will discuss how MaxNet differs from existing solutions for achieving Max-Min fairness. Achieving Max-Min fairness on Internet like networks has till now only been shown for some special cases, where source utility functions are either globally manipulated, or in the limit of a particular family of utility functions. In this chapter we will discuss how, unlike the Internet, MaxNet is able to achieve Max-Min fairness for arbitrary well behaved homogenous sources, with no global coordination. Extensive research has been performed on achieving Max-Min fairness in ATM networks. We will review some of this research, and show that these methods cannot be translated to the connectionless philosophy of an Internet like network. MaxNet is unique in that it is a fully distributed algorithm and does not require per-flow information storage or processing at the links to achieve Max-Min fairness.

In Chapter 3, we described an analysis of network stability presented by Paganini *et al.* in [100] which describes the dimensions of network topology which affect the stability of the control system: 1) Capacity, 2) Delay 3) Number of bottlenecked links on end-to-end path 4) Number of sources. In [100] it is shown that a gain in the congestion control loop is required for each of these dimensions to remove the dependence of stability on each dimension. In particular, as described in

Chapter 3, the congestion signal is scaled at the source by a factor $1/M_i$, where M_i is the number of bottlenecked links on the end-to-end path of the connection from source i. This gain makes the system invariant to the number of bottleneck links in the path. The current Internet does not provide a means of measuring and communicating M_i to the source algorithms. In this case, to remain stable, the congestion control system must operate with an M_i which is set to some upper-bound. Operating with an upper-bound of the number of bottleneck links, results in conservative control, and slower than optimal convergence of source rates. A possible scheme for measuring M_i is introduced in [100] which adds dedicated signalling in links and sources. In the MaxNet network, because only the most severely bottlenecked link on the end-to-end link generates the congestion signal, M_i is always 1. We will prove that MaxNet removes the need to scale the congestion control signal by the number of bottlenecked links on the end-to-end path. Furthermore, we will prove that MaxNet is arbitrarily scalable, and is stable and robust under any network topology.

In this chapter, we also analyse the consequences of the Internet's one-bit communication scheme on the speed and accuracy of congestion control and show how MaxNet can achieve more efficient congestion signal communication.

6.2 Internet Congestion Signalling

Congestion control on the Internet has been widely analysed using macro-economic and control systems theory [76] [100] [62]. In general, source and link processes are described by

$$\text{Source } i\text{: } r_i(t+1) = D_i(S_i(t)) \tag{6.1}$$

$$\text{Link } l\text{: } d_l(t+1) = G(y_l(t),d_l(t)) \tag{6.2}$$

where $r_i(t)$ is the rate of source i at time t, $S_i(t)$ is the network's congestion signal communicated to source i, effectively a price per byte, $d_l(t)$ is the congestion signal link l generates, and $y_l(t)$ is the aggregate traffic arriving at link l.

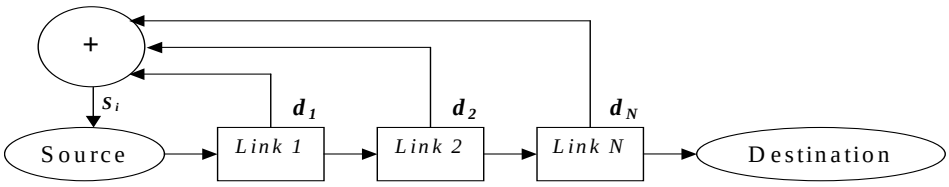
The sources transmit at a rate determined by their demand function, $D_i(S_i(t))$ based on the congestion feedback $S_i(t)$. Each demand function D_i is associated with a utility function U_i, as D_i is the inverse of the marginal utility function $D_i = U_i'^{-1}$. The links set the congestion price $d_l(t)$, by evaluating the link congestion algorithm $G(y_l(t),d_l(t))$ to control demand. The congestion signal communicated to the source i, $S_i(t)$, is calculated using the congestion prices $d_l(t)$ of each link on the end-to-end communication path of the source. On the current Internet, as shown below, and in a number of proposals [76] [100] [62], this aggregate congestion price is a sum of the congestion prices of each of the links on the end-to-end path:

$$S_i(t) \approx \sum_{all\ bottleneck\ links\ on\ end-to-end\ path} d_l(t)$$

Fig. 6.1 shows the system for one source. We will use the term SumNet to refer to networks that use the sum, or approximately the sum, of link prices as the feedback signal.

The current Internet is a SumNet because of the way congestion information is signalled. To communicate the link price $d_l(t)$, each link l generates congestion notification (packet dropping or ECN marking) randomly at a mean rate $m_l(t)$ at time t. In the current Internet the notification rate is effectively the link price $d_l(t) = m_l(t)$. Let $M(t)$ be the rate of congestion notification received by a source of a given end-to-end path with N bottleneck links.

Fig. 6.1 SumNet Control Loop



Assuming each bottleneck link produces congestion notifications independently, we obtain:

$$M(t) = 1 - \prod_{i=1}^{N} (1 - m_i(t)). \tag{6.3}$$

Assuming $m_i(t) << 1$, we obtain approximately:

$$M(t) \approx \sum_{i=1}^{N} m_i(t).$$

In Section 6.3 we introduce MaxNet that, as shown in Section 6.4, leads to Max-Min fairness for homogenous sources. In Section 6.9.1, we show that SumNet networks do not achieve Max-Min fairness for all source demand functions.

6.3 MaxNet Architecture

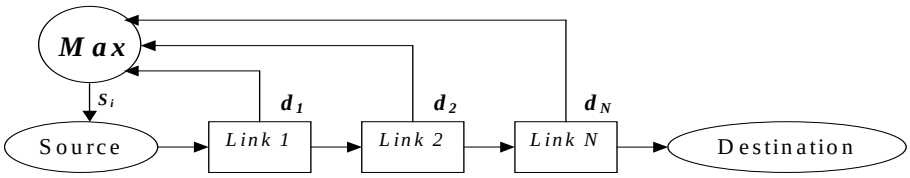
We will now describe the MaxNet congestion control architecture. The MaxNet architecture uses the maximum price of all the link prices on the end-to-end path as the feedback signal to the source. To achieve this, the packet format must include bits to communicate the complete congestion price.

Packet Header	**Congestion Price**	**Data Bits**
H Bits	*S* Bits	*D* Bits

Fig. 6.2 MaxNet Packet Format

Each link replaces the current congestion price in the packet if the link's congestion price is greater than the one in the packet. In this way, the maximum congestion signal on the path is communicated to the destination, which relays the information to the source in acknowledgement packets. Fig. 6.3 presents the control loop of MaxNet:

Fig. 6.3 MaxNet Control Loop



We will now show the source rate allocation that MaxNet achieves for general sources and later show that the rate allocation is Max-Min fair for a network of sources with homogenous demand functions. For this proof, the only requirement on the link congestion algorithm $G(y_l(t),d_l(t))$ is that the congestion price converges towards a steady state price at the point when the aggregate packet arrival rate, $y_l(t)$, is equal to the link target capacity C_l. The most simple form of such a process is the integrator, which has been studied in [100] and [76]:

$$d_l(t+1) = d_l(t) + \mu_l(y_l(t) - C_l) \tag{6.4}$$

6.4 MaxNet Rate Allocation

6.4.1 General Rate Allocation

In this section, we describe the source rate allocation for heterogenous sources on a MaxNet network. On a MaxNet network, each link marks the packet with its own price if it is greater than the price already marked in the packet, therefore each flow rate is controlled only by the maximum price of all prices encountered on the end-to-end path. Let L_l be the set of flows through the link l, T_f be the set of the links that flow f traverses and let N_l be the number of flows controlled by link l, i.e., flows whose maximum price (path price) is equal to the price at link l. Then the price control law for link l with price d_l, from (6.4), is:

$$d_l(t+1) = d_l(t) + \mu_l\left(\sum_{x \in L_l} D_x(Max\{d_f : f \in T_x\}) - C_l\right) \tag{6.5}$$

In general, a source *i*, with demand function $D_i(S)$ will achieve rate r_i in steady state, where:

$$r_i = C_l \frac{D_i(Max(d_f \quad f \in T_i))}{\sum_{x \in L_l} D_x(Max(d_f \quad f \in T_x))} \quad i \in L_l$$

For the case where the source *i* is bottlenecked at the most severe bottleneck in the network *l*, with price d_0, it can be seen that the rate r_i is in proportion to the magnitude of the demand function relative to other demand functions:

$$r_i = C_l \frac{D_i(d_0)}{\sum_{x \in L_l} D_x(d_0)} \quad i \in L_l$$

Since sources receive bandwidth in proportion to their demand functions, MaxNet can achieve differentiated bandwidth allocation for a multi-service network. In the following section we show the unique properties of MaxNet when all of the source demand functions are the same.

6.4.2 Max-Min Fair Rate Allocation

We now show that the flow rates of MaxNet in the steady state are Max-Min fair for sources with the same demand function. The assumption of homogeneity is a starting point for the analysis of MaxNet. This assumption of a single source algorithm is supported by the situation on the Internet where most traffic is generated by TCP [116], with TCP/Reno being widely deployed.

Assume all sources share the same demand function $r_i = D(S_i)$, where r_i is the transmission rate of source *i* and S_i is the network congestion price seen by this source. The function $D(S_i)$ for $S_i \geq 0$ is assumed to be continuous, positive and decreasing. Let vector *X* contain all the source flow rates through the network arranged into subsets which contain flows of equal rates. Within the subset *y* of components of *X*, elements (flow rates) are denoted x_{ya}, x_{yb}, … The subsets are ordered as follows:

$$X = (x_{0a}, x_{0b}, \ldots), (x_{1a}, x_{1b}, \ldots), \ldots (x_{Na}, x_{Nb}, \ldots)$$
$$x_{0a}=x_{0b}=\ldots < x_{1a}=x_{1b}=\ldots < x_{Na}=x_{Nb}=\ldots . \quad (6.6)$$

Let X_z be the vector $(x_{za}, x_{zb}, \ldots)$. Also let x_z be the rate $x_z = x_{za} = x_{zb} = \ldots$. Let vector P contain the congestion prices that correspond to each source. Each element in P corresponds to an element in X such that $x_{yz} = D(p_{yz})$ or $p_{yz} = D^{-1}(x_{yz})$. Thus,

$$P = (D^{-1}(x_{0a}), D^{-1}(x_{0b}),\ldots),$$
$$(D^{-1}(x_{1a}), D^{-1}(x_{1b}),\ldots), \ldots$$
$$(D^{-1}(x_{Na}), D^{-1}(x_{Nb}),\ldots)$$

Let p_z be the value $p_z=p_{za}=p_{zb},\ldots$. Given (6.6) and the assumptions regarding $D(S_i)$ being a decreasing function, the lowest rate in the network experiences the highest price, and the prices are ordered in reverse order to rates:

Deleted: *vector*

Deleted: *(*

Deleted: ??????

Deleted: ¶

$$p_{0a}=p_{0b} \ldots > p_{1a}=p_{1b}=\ldots > p_{Na}=p_{Nb}=\ldots$$
$$p_0 > p_1 > p_N . \quad (6.7)$$

Note that for every bottlenecked link l with price d_l there is one element with value d_l in array P for each flow controlled by link l.

The network must have at least one link corresponding to the maximum price p_0. Let W_z be the set of links with price p_z. All the flows through link(s) in W_0 will be marked with the congestion price p_0 because it is the maximal in the network. Given that the link price algorithm in (6.5) is employed at every link, the actual capacity allocated to the flows of rate x_0 in steady state is:

Deleted: Bartek: P_z is not a price - it is a vector!!!

Deleted: and a

Deleted: Again Bartek: P_z is not a price - it is a vector!!!

Inserted: Again Bartek: P_z is not a price - it is a vector!!!

$$p_0(t+1) = p_0(t) + \mu_0\left(D(p_0)\cdot N_l - C_l\right) \quad l \in W_0 .$$

In steady state $p_0(t+1) = p_0(t)$, thus:

Deleted: Bartek, now you say that $P_0(t+1)$ is capacity. It is defined as Vector of equal prices.

Inserted: Bartek, now you say that $P_0(t+1)$ is capacity. It is defined as Vector of equal prices.

$$p_0 = D^{-1}\left(\frac{C_l}{N_l}\right) \quad l \in W_0 .$$

Notice that the flows of rate x_0 share their bottleneck links equally. If we apply the Max-Min condition only to this set of minimum rate flows (the elements of the vector X_0), we see that they are Max-Min fair, since they are feasible and no rate can be increased without decreasing another rate within the flow represented by the vector X_0. We now extend this argument to include flows with price p_1.

The links in W_1 are the links with the second highest congestion prices p_1, $p_0 > p_1 > p_n \;\; n \neq 0,1$. All flows through link(s) W_1 are either already controlled by nodes with price p_0, $p_0 > p_1$, or are controlled by the W_1 links, since they have the next highest congestion price in the network. Let a_{il} be the number of flows traversing link l that are controlled by a link with path price p_i. Then the price control law for links with price p_1 is:

$$p_1(t+1) = p_1(t) + \mu_1\left(D(p_1)\cdot N_l + a_{0l}\cdot D(p_0) - C_l\right) \quad l \in W_1$$

and in steady state:

$$p_1 = D^{-1}\left(\frac{C_l - a_{0l}\cdot D(p_0)}{N_l}\right) \quad l \in W_1 . \tag{6.8}$$

Note that the allocation of rates to flows passing this link is Max-Min fair. The minimum flows through this link i.e., those with a congestion price p_0, are not controlled nor bottlenecked at this link, and we have already shown that their rates are maximized. All other flows receive an equal share of the remaining bandwidth by (6.8), which is also a Max-Min fair allocation. We now extend the set of flows to include all the flows in the network to show global Max-Min fairness by induction.

Let us assume that capacity allocation to flows with prices p_{k-1}, p_{k-2} … p_0 is Max-Min fair, then we show that capacity allocation to flows with price p_k is also

Max-Min fair. In general, links with price p_k will have a set of flows traversing them which are controlled by other links whose prices are higher, p_{k-1}, p_{k-2} … p_0, and a set of flows which are controlled by them with the price p_k. The reasoning which produced (6.8) can be generalising so that any link l with price p_k will set p_k according to:

$$p_k = D^{-1}\left(\frac{C_l - (a_{(k-1),l} \cdot D(p_{k-1}) - a_{(k-2),l} \cdot D(p_{k-2}) - ... - a_{0,l} \cdot D(p_0))}{N_l}\right) \quad l \in W_k \quad (6.9)$$

Notice that (6.9) allocates capacity so that flows with smaller rates than those controlled by p_k, $p_k < p_r$ $k > r$, are not bottlenecked at the W_k link(s). We have assumed these flows already have Max-Min allocation at other links. The remaining capacity is allocated equally to the flows which are bottlenecked at the D_k link(s), thus satisfying the Max-Min condition for such flows. Therefore the Max-Min condition is satisfied by all flows traversing the W_k link(s). Because we have shown that the p_0 allocation is Max-Min fair, and that, if p_{k-1} is fair then p_k is fair, it follows by induction that all allocations are Max-Min fair and global fairness is achieved.

QED.

Note that this proof is for steady state. Stability around this steady state equilibrium point is studied in Section 6.5 and transient effects are further simulated in Section 6.6.

6.4.3 Weighted Max-Min Fairness

We will now show that MaxNet can achieve weighted Max-Min fairness. Let the source demand function for source i be $r_i = a_i.D(S_i)$, where a_i is the 'weight' for source i and D is the well behaved demand function shared by all sources in the network. We can consider the demand function $r_i = a_i.D(S_i)$, to be a_i virtual sources with demand function $D(S_i)$. We have proven Max-Min fairness when all demand functions are $r_i = D(S_i)$. This means that weighted Max-Min fairness is achieved for the actual sources, since they behave as a_i homogenous sources each.

6.5 MaxNet Stability and Robustness

In this section, we analyse the stability and robustness properties of MaxNet. Our analysis applies the stability and robustness analysis of SumNet in [100] to the case of MaxNet. This section shows that MaxNet congestion control is arbitrarily scalable and maintains stability for arbitrary network topologies and arbitrary amounts of delays. Furthermore, whilst in [100] SumNets are shown to need to estimate and communicate the number of bottlenecked links on the end-to-end path to achieve this property of arbitrary scaling, we show that MaxNet does not need to communicate or estimate the number of bottleneck links on the end-to-end path to be arbitrarily scalable.

As in the analysis of [100], we consider a network of L communication links shared by a set of S sources. The network modelled is shown in Fig. 6.4. It is modelled in the Laplace domain. Each link, l, has capacity c_l, aggregate arrival rate y_l and price d_l. Each source, i, has source rate r_i and receives aggregate price q_i. The forward routing matrix describes the links loaded by each source:

$$\left[\overline{R}_f(s)\right]_{l,i} = \begin{cases} e^{-\tau_{i,l}^f s} & \text{if source } i \text{ uses link } l \\ 0 & \text{otherwise} \end{cases}$$

where $\tau_{i,l}^f$ is the delay from source i to link l. The backward routing matrix describes the aggregation method and feedback path of the congestion signals from links to sources. In [100] where a SumNet is considered, all of the bottleneck links on the end-to-end path of a source have a backward feedback path to the source. With MaxNet, only one link, the link with the maximum congestion signal on the end-to-end path of the source, has a backward path to the source. The MaxNet backward routing matrix changes if the maximum bottleneck links on a path changes; however, over each period when the order of bottleneck links remain unchanged, it is described by:

$$\left[\overline{R}_b(s)\right]_{l,i} = \begin{cases} e^{-\tau_{i,l}^b s} & \text{if source } i \text{ uses link } l \text{ and } d_l = Max(d_f \quad f \in T_i) \\ 0 & \text{otherwise} \end{cases}$$

Note that the RTT is the sum of the forward and backward delays:

$$\tau_i = \tau_{i,l}^b + \tau_{i,l}^f \tag{6.10}$$

The closed loop system as shown in Fig. 6.4 can then be described by:

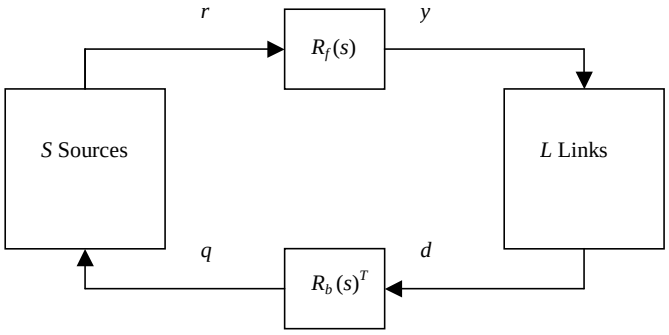
$$y(s) = \overline{R}_f(s) \cdot r(s)$$
$$q(s) = \overline{R}_b(s)^T \cdot d(s)$$

In this analysis (as in [100]) the dynamic properties around an equilibrium point, r_0, y_0, d_0 and q_0 are studied. Consider small perturbations around the equilibrium, $r = r_0 + \delta r$, $y = y_0 + \delta y$, $d = d_0 + \delta d$, $q = q_0 + \delta q$. Assuming the set of bottlenecks is unchanged by this small perturbation, δd is only non-zero for bottleneck links [100]. Therefore, as in [100], for the local analysis, we can write the reduced model:

$$\delta y(s) = R_f(s) \cdot \delta r(s)$$
$$\delta q(s) = R_b(s)^T \cdot \delta d(s)$$

where the matrices R_f and R_b and the vectors δr and δd are obtained by eliminating the rows corresponding to non-bottleneck links. In this small signal model, the backward and forward routing matrices are of full row rank, ruling out degenerate cases as detailed in [100].

Fig. 6.4 General Flow Control Structure Based on Pricing Signals



As described in Chapter 3, the analysis in [100] proves that the SumNet closed loop system of Fig. 6.4 is stable and robust so long as each source *i* has a gain:

$$\kappa_i = \frac{\alpha_i r_{0i}}{M_i \tau_i} \tag{6.11}$$

where $0 < \alpha_i < 1$ is a gain parameter, r_{0i} is the steady state rate, τ_i the RTT and M_i is the number of bottleneck links in source *i*'s path. This gain imposes some restrictions on the shape of the demand function. The relationship between this gain and the types of source algorithms possible is described in detail in [100]. Each link must also have a gain scaled by the capacity:

$$d_l = \frac{1}{c_l s} y_l$$

We will prove that the same stability and robustness also holds for MaxNet, without needing to scale by M_i, with a simpler source gain:

$$\kappa_i = \frac{\alpha_i r_{0i}}{\tau_i}$$

Eliminating M_i has several advantages. Firstly, the additional signalling infrastructure required to determine M_i, as proposed for SumNet in [100] is removed. Without this signalling infrastructure, to remain stable, SumNets must assume an

upper-bound on M_i and have a slow conservative control policy. With MaxNet M_i is always 1 and this avoids additional signalling infrastructure or a conservative control policy.

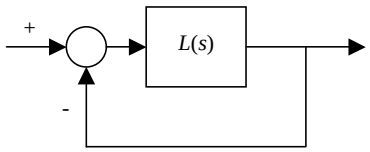
The overall multivariable feedback loop that describes the small signal model of Fig. 6.4 is shown in Fig. 6.5. The closed loop transfer function is:

$$L(s) = R_f(s) \cdot \mathrm{K} \cdot R_b^T(s) \cdot C \frac{I_L}{s}$$

where I_L is an identity matrix of size L and the gains are:

$$\mathrm{K} = \mathrm{diag}(\kappa_i), \qquad C = \mathrm{diag}\left(\frac{1}{c_l}\right)$$

Fig. 6.5 Overall Feedback Loop



Let $F(s) = \frac{R_f(s) \cdot \mathrm{K} \cdot R_b^T(s) \cdot C}{\gamma}$ where $L(s) = \gamma F(s) \frac{I_L}{s}$, then in [100] it is proven that given the below conditions, the feedback system is stable for all $\gamma \in (0,1]$:

(i) $F(s)$ is analytic in $\mathrm{Re}(s) > 0$, and $\|F(s)\| \le \beta$ in $\mathrm{Re}(s) > 0$, where β is a finite real number.
(ii) $F(0)$ has strictly positive eigenvalues.
(iii) For all $\gamma \in (0,1]$, -1 is not an eigenvalue of $L(jw)$ $w \neq 0$.

The full proof of stability given these conditions is described in [100]. We will not repeat this proof here, but will prove that MaxNet satisfies the conditions and is therefore also stable and robust.

Just as in [100], condition (i) is automatically satisfied. What remains is to prove (ii) and (iii). To do this, we need to impose the order on the price vector which arises from the MaxNet signalling, and results in structured forward and backward routing matrices.

Without loss of generality, let the price vector d be ordered, such that $d_0 \geq d_1 \geq \mathrm{K} \geq d_L$. Associated with each source i, there is one link n_i which is the maximum bottleneck link that controls that source. Then let the sources be ordered such that $n_1 \leq n_2 \leq \mathrm{K} \leq n_S$. Since every bottleneck link has at least one source it controls:

$$n_1 = 1 \quad n_S = L \quad n_i \leq n_{i+1} \leq n_i + 1 \tag{6.12}$$

Note, that we now use L as the number of bottleneck links only. We can now proceed to prove conditions (ii) and (iii).

Proposition 1. $F(0)$ has strictly positive eigenvalues.

Proof: Rewrite $F(s)$ as the product of two matrices which have all of the zero elements of the forward and backward routing matrices:

$$eig(F(s)) = eig(\frac{R_f(s) \cdot \mathrm{K} \cdot R_b^T(s) \cdot C}{\gamma}) = eig(\hat{R}_f(s) \cdot \hat{R}_b^T(s))$$

where:

$$\hat{R}_f(s) = \frac{R_f(s) \cdot \mathrm{K}}{\gamma} \qquad \text{note that,} \qquad \left[\hat{R}_f(s)\right]_{ij} = 0 \quad \textit{if} \quad \left[R_f(s)\right]_{ij} = 0$$

$$\hat{R}_b^T(s) = R_b^T(s) \cdot C \qquad \text{note that,} \qquad \left[\hat{R}_b^T(s)\right]_{ij} = 0 \quad \textit{if} \quad \left[R_b^T(s)\right]_{ij} = 0 .$$

Using the fact that the eigenvalues of a lower triangular matrix are its diagonal elements, we will now proceed to show that $F(0)$ has strictly positive eigenvalues by showing $F(s)$ is a lower triangular matrix and when $s = 0$, the diagonal elements are strictly positive. For a lower triangular matrix, $G(s)$, all elements above the diagonal must be zero:

$$\left[G(s)\right]_{ij} = 0 \quad for\ j > i$$

The elements of *F*(*s*) are:

$$\left[F(s)\right]_{ij} = \sum_{z=1}^{S} \left[\hat{R}_f(s)\right]_{iz} \cdot \left[\hat{R}_b^T(s)\right]_{zj}$$

Since each source only receives feedback from one bottleneck link, we know that:

$$\left[\hat{R}_b^T(s)\right]_{zj} = 0 \quad if \quad n_z \neq j$$

By definition, a source does not use links with higher congestion prices than the maximum bottleneck link, so:

$$\left[\hat{R}_f(s)\right]_{iz} = 0 \quad if \quad n_z > i$$

Since,

$$\left[F(s)\right]_{ij} = 0 \quad if \quad \left[\hat{R}_f(s)\right]_{iz} \cdot \left[\hat{R}_b^T(s)\right]_{zj} = 0 \quad for \quad z = 1,2\text{K } S$$

any element of F(s) will necessarily be zero under any of the three conditions:

$$\left[F(s)\right]_{ij} = 0 \quad if \quad (n_z \neq j)\ or\ (n_z > i) \quad for \quad z = 1,2\text{K } S$$

Given the condition that $j > i$, the it is clear that $\left[F(s)\right]_{ij} = 0$ is true, which makes *F*(*s*) a lower triangular matrix. The diagonal elements of *F(s)* are:

$$\left[F(s)\right]_{ii} = \frac{1}{\gamma \cdot c_i} \sum_{z=1}^{S} \kappa_z \left[R_f(s)\right]_{i,z} \left[R_b^T(s)\right]_{z,i} = \frac{1}{\gamma \cdot c_i} \sum_{z=1}^{S} \begin{cases} \dfrac{\alpha_z r_{0z}}{\tau_z} e^{-\tau_{zi} s} & if\ n_z = i \\ 0 & otherwise \end{cases}$$

At *s* = 0, this reduces to:

$$[F(0)]_{ii} = \frac{1}{\gamma \cdot c_i} \sum_{z=1}^{S} \begin{cases} \frac{\alpha_z r_{0z}}{\tau_z} & if \ n_z = i \\ 0 & otherwise \end{cases}$$

Note that for each i, i=1,2 …L at least for one source z, $n_z = i$, since a bottleneck link must be the maximum bottleneck for at least one source in the network, otherwise it is not a bottleneck at all. Equivalently, for at least one source z, $n_z = i$, to satisfy conditions (6.12). Since $[F(0)]_{ii}$ is a non-empty sum of at least one positive element and zero elements, $[F(0)]_{ii} > 0$, satisfying condition (ii). QED.

Proposition 2. For all $\gamma \in (0,1]$, -1 is not an eigenvalue of $L(jw)$ w≠0.

Proof: Using (6.10) rewrite the backward routing matrix in terms of the forward and total delays only:

$$R_b^T(jw) = diag(e^{-\tau_i s})\tilde{R}_b^T(jw)$$

where the matrix $\tilde{R}_b(s)$ has the same sparsity structure as $R_b(s)$:

$$[\tilde{R}_b(s)]_{l,i} = \begin{cases} e^{\tau_{i,l}^{f} s} & if \ source \ i \ uses \ link \ l \ and \ d_l = Max(d_f; f \in T_i) \\ 0 & otherwise \end{cases}$$

Define the matrices Λ_0 and R_0:

$$R_0 = diag(r_{0i}) \qquad \Lambda_0 = diag(\lambda_i(s)) \qquad \lambda_i(s) = \frac{\alpha_i \cdot e^{-\tau_i s}}{\tau_i s}$$

Then we can rewrite $L(s)$ as:

$$L(jw) = R_f(jw) \cdot R_0 \cdot \Lambda(jw) \cdot \tilde{R}_b^T(jw) \cdot C$$

Split the matrix $L(jw)$ such that:

$$L(jw) = \overset{\downarrow}{R}_f(jw) \cdot \overset{\downarrow}{R}_b^T(jw)$$

where

$$\hat{R}_f(jw) = R_f(jw)\cdot R_0\cdot\Lambda(jw) \qquad \hat{R}_b^T(jw) = \tilde{R}_b^T(jw)\cdot C$$

Note that since

$$\left[\hat{R}_f(s)\right]_{ij} = 0 \quad if \quad \left[R_f(s)\right]_{ij} = 0$$
$$\left[\hat{R}_b^T(s)\right]_{ij} = 0 \quad if \quad \left[R_b^T(s)\right]_{ij} = 0$$

the matrix $L(jw)$ is lower triangular, because the same proof applies as was used to show that $F(s)$ is lower triangular. To determine the eigenvalues of $L(jw)$ we need only to determine its diagonal elements.

$$\left[L(jw)\right]_{ii} = \sum_{z=1}^{S}\left[\hat{R}_f(jw)\right]_{iz}\cdot\left[\hat{R}_b^T(jw)\right]_{zi}$$
$$\left[L(jw)\right]_{ii} = \frac{1}{c_i}\sum_{z=1}^{S}\begin{cases} r_{0z}\cdot\lambda_z(jw) & if\ n_z = i \\ 0 & otherwise \end{cases}$$

We can show that $-1 \notin eig(L(jw))$ by showing

$$\mathrm{Re}\left[L(jw)\right]_{ii} > -1.$$

In [100] it is proven that if $0 < \alpha_i < 1$ then $\mathrm{Re}\left[\lambda_i(jw)\right] > -1$. Let us choose a λ such that:

$$\mathrm{Re}\left[\lambda_i(jw)\right] \geq \mathrm{Re}\left[\lambda(jw)\right] > -1 \ for\ i = 1\mathrm{K}\ S$$

then,

$$\left[L(jw)\right]_{ii} \quad \geq \quad \lambda\cdot\frac{1}{c_i}\sum_{z=1}^{S}\begin{cases} r_{0z} & if\ n_z = i \\ 0 & otherwise \end{cases} \tag{6.13}$$

Note that,

$$0 \quad < \quad \frac{1}{c_i}\sum_{z=1}^{S}\begin{cases} r_{0z} & if\ n_z = i \\ 0 & otherwise \end{cases} \quad \leq \quad 1$$

since this represents the sum of all steady state flow values which are bottlenecked at link i, which is clearly less than the capacity c_i. Since for all $\gamma \in (0,1]$, $\mathrm{Re}\left[\lambda(jw)\right] > -1$, it follows by (6.13) that -1 is not an eigenvalue of $L(jw)$ w≠0. QED.

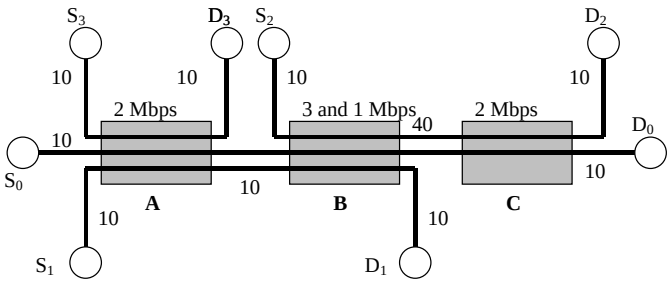
Having shown MaxNet satisfies the three conditions of the proof in [100], we have established that MaxNet is stable for arbitrary topologies and network delays, without having to communicate the number of bottlenecked links on the end-to-end path.

As discussed in detail in [100], the scaling requirement (6.11) on the feedback to the source has implications on the source flow control algorithm and possible utility functions. It is shown that it is still possible to achieve a wide variety of source behaviour given this feedback requirement. In particular, source algorithms like TCP/RENO or TCP/VEGAS have been modelled by source control laws which satisfy these requirements. Furthermore [100] discusses how it is possible to have source flow control laws, whose steady state flow rate is invariant to RTT delays.

6.6 Simulation Results

To gain insight into the transient behaviour of MaxNet, a fluid-flow model of a MaxNet network was simulated. The network topology simulated is shown in Fig. 6.6. The network has four sources, S_0 to S_3, which transmit to four destination hosts D_1 to D_4 respectively. Each source has a source demand function described by (6.20) where r_{max}=15 and k=0.007. There are three bottleneck links, A,B,C of initial capacity 2 Mbps, 3 Mbps and 2 Mbps respectively. Bottlenecked links have no delay. Non-bottleneck links which interconnect the bottleneck links and sources, have finite delay and infinite capacity. The number next to each non-bottleneck link represents the delay of the link in simulation time step units. The simulation was run over 8000 time step units, and the source rates are plotted in Fig. 6.7 to 6.10. To generate a transient response, the capacity of link B is reduced from 3Mbps to 1Mbps at time step 4000.

Firstly, note that the steady state rates before and after the transient are Max-Min fair. This agrees with our steady state analysis. Despite the large change in capacity of link B, the convergence to new rates occurs without significant oscillatory behaviour. The simulated network is also stable under this transient, which gives some confirmation to our stability analysis.


S3
D3
S2
D2
10
10
10
10
2 Mbps
3 and 1 Mbps
40
2 Mbps
D0
10
S0
10
A
10
B
C
10
10
S1
D1

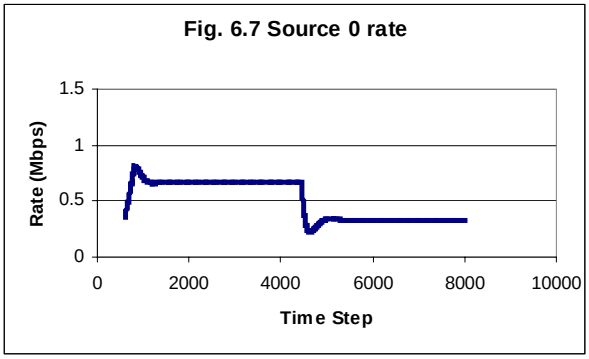


Fig. 6.7 Source 0 rate
Rate (Mbps)
1.5
1
0.5
0
0
2000
4000
6000
8000
10000
Time Step

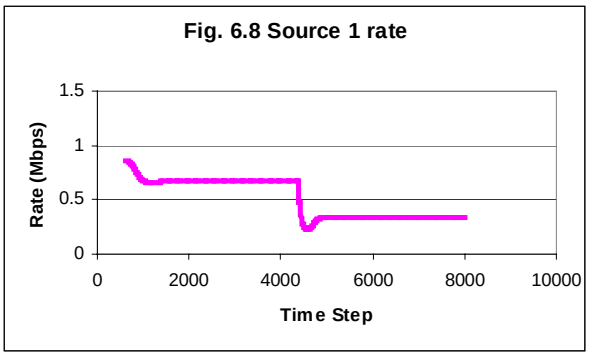


Fig. 6.8 Source 1 rate
Rate (Mbps)
1.5
1
0.5
0
0
2000
4000
6000
8000
10000
Time Step

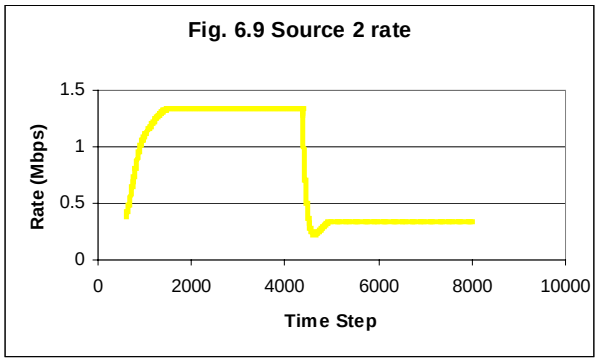
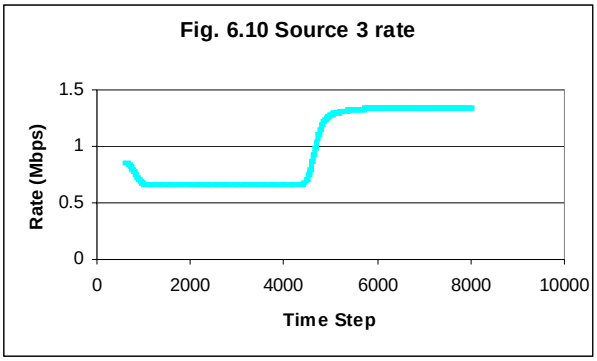


Fig. 6.9 Source 2 rate
Rate (Mbps)
1.5
1
0.5
0
0
2000
4000
6000
8000
10000
Time Step

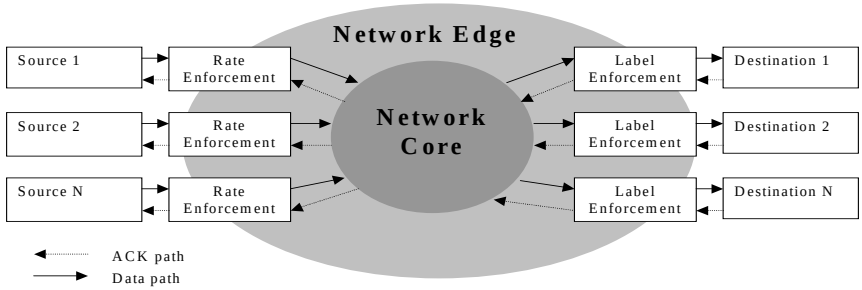


6.7 Enforcing Fairness on Unresponsive Flows

In chapter 2, we discussed a number of schemes, such as CSFQ and CHOKe, which enforce flow rate equality so that malicious users who do not respond to congestion control are controlled. We have explained that these schemes discard packets locally at the links where the algorithm resides. As discussed this link-local approach is not efficient and unable to prevent congestion collapse due to malicious users, because network resources up to the link that performs the discarding are wasted.

MaxNet provides a platform for policing malicious users efficiently over a network where the core is connectionless and has no per-flow processing or storage. The policing architecture relies on the assumption that the network can be trusted. There are two components to the architecture 1) The Rate Enforcement module which sits at the entry of the flow to the network, and 2) The Label Enforcement module which sits at the exit of the flow to the destination host.

Fig. 6.11 Rate Enforcement in MaxNet Network



The Rate Enforcement (RE) module sits at the edge of the network, perhaps at the access ISP, and performs per-flow monitoring on the connections between the source and the network. Per-flow monitoring is achievable and scalable at the edge, since only a small number of flows exist at each access point. The RE module measures the rate of each flow. ACK packets from the destination, which contain the end-to-end path congestion signal $S(t)$, pass through the RE module. The RE module is programmed with a 'fair' demand function, which maps a congestion signal value to an allowable transmission rate $x_f(t) = D_f(S(t))$. Packets transmitted above the fair rate are discarded at the RE module.

The Label Enforcement (LE) module sits at the edge of the network, between the network and the destination host. The single role of the LE module is to ensure that the congestion value in the packet transported in the Data packet to the destination host, is of the same value as of the ACK packet returned back to the source. This is done by per-flow monitoring of the flows, by storing the congestion signal going to the destination and comparing it with the information relayed back. This ensures that malicious users do not modify the destination algorithm so that a lower congestion value is returned to the source algorithm.

MaxNet makes this architecture for policing streams at the edge possible. Unlike the link-local schemes, the edge policing method can prevent malicious users wasting resources along the network by controlling their traffic before it enters the network.

6.8 Congestion Signalling Scheme Efficiency

6.8.1 Introduction

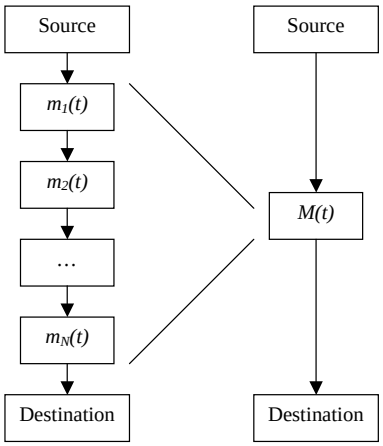
In this section, we will analyse the method of communicating the congestion signal in SumNets and MaxNet. The Internet, and existing proposals of SumNet networks use a one-bit packet marking scheme. The marking is applied randomly and each one-bit mark is independent of other marks. MaxNet uses a packed bit format, which stores a complete congestions signal value in the packet. We will show that the MaxNet scheme is more efficient at communicating congestion information than independent marking schemes such as the Internet or REM.

The current Internet's packet marking or dropping is a one bit coding scheme which communicates and sums the end-to-end link prices on the source to destination path. On the current Internet, each packet sent along the end-to-end link communicates one bit of information about the congestion state of the path. The number of packets needed to obtain a measure of the aggregate path price $S_i(t)$, within some accuracy bounds, bounds the minimum possible time for rates to be controlled with said accuracy. In this section, we analyse the number of bits required to communicate the aggregate price to a specified precision, for random linear marking used on the Internet and Random-Exponential Marking REM proposed in [76]. We will compare both of these independent marking schemes to the packed-bit format used in MaxNet.

6.8.2 Linear Marking

The Internet uses linear packet marking which is random and independent at each link l, with a marking rate $m_l(t)$ that corresponds to the link congestion price $p_l(t)$, $m_l(t) = p_l(t)$. The effective aggregate path marking rate for source i is $M_i(t)$ as given by (6.3). We will model the links on the end-to-end path by a single equivalent link, whose marking rate is $M_i(t)$ as in Fig 6.12. The range of $M_i(t)$ is between 0 and 1.

Fig 6.12. Single Link Equivalent Path



As discussed, for linear marking, the aggregate path price $S_i(t) = M_i(t)$, is not an exact sum of the individual link prices (6.3). Regardless of the number of packets transmitted, the exact sum will not be communicated to the source. Therefore, the performance measure which we determine is the number of packets that need to be sent, so that the estimate of $M_i(t)$ at the source, $\hat{M}_i(t)$, is within $\pm\Delta M$ of the actual $M_i(t)$.

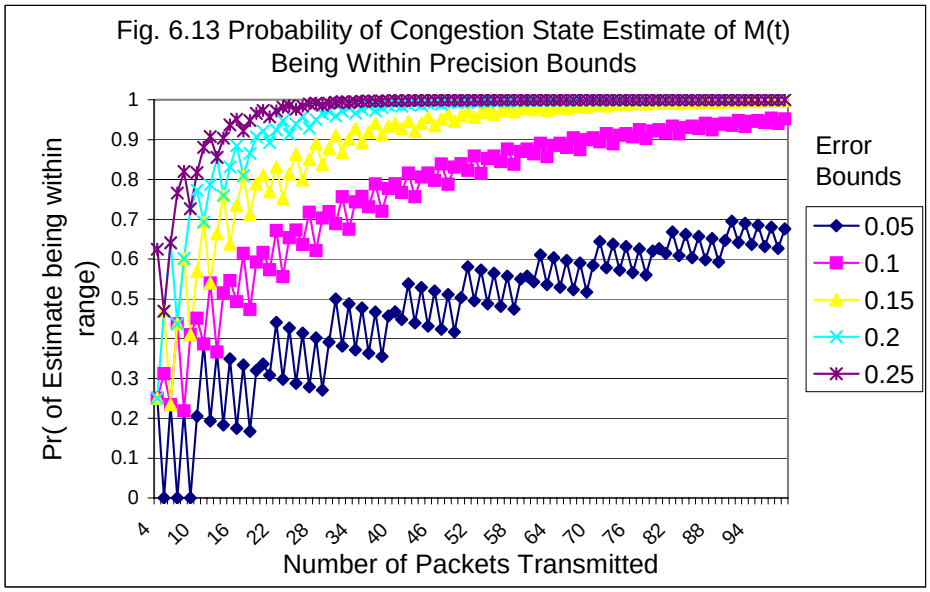
Sources can estimate $M_i(t)$ by counting the percentage of packet drops or marks. Note that since the concept of a demand function is an abstraction of a source algorithm such as TCP, TCP does not actually count marks to estimate the congestion signal. However, a hypothetical algorithm which has the same demand function as TCP, but establishes $M(t)$ in the minimum time must count packet drops or marks.

The performance of linear one-bit marking is described by (6.15). (6.15) gives the probability that the estimate aggregate marking rate $\hat{M}_i(t)$, is within the accuracy bound $\pm\Delta M$ given that the N packets have been transmitted during a period when each of the link marking rates have not changed:

$$\Pr(M_i(t) - \Delta M < \hat{M}_i(t) < M_i(t) + \Delta M) = \sum_{S=\lceil N\cdot(M_i(t)-\Delta M)\rceil}^{S=\lceil N\cdot(M_i(t)+\Delta M)\rceil} \binom{N}{S} M_i(t)^S \cdot (1 - M_i(t))^{N-S} \qquad (6.15)$$

Note that (6.15) is essentially the binomial distribution. Fig. 6.13 shows the plot of (6.15) for increasing N. Each series in the graph shows the probability of the estimate being within the error bounds given that the specified number of packets have been received. (The jaggedness of the graph arises from the fact that the bounds of the summation must be integers since only an integer number of packet drops or marks can occur.)

Notice that it takes >10 packet to have an estimate of the congestion state that begins to be reasonable, i.e., better than +-25% of the actual value. Fundamentally this means that short transfers on the Internet cannot possibly adjust their rate accurately to match the capacity with a one-bit packet marking or dropping scheme. This is particularly a problem since most transfers are in fact short HTTP requests.



To obtain a simple lower bound on the number of bits required to communicate the congestion signal within an accuracy bound, we can assume that packet marking is deterministic with rate $M_i(t)$. Then the number of bits required is:

$$N_{Independent\ Marking} = \frac{1}{\Delta M}$$

6.8.3 Exponential Marking

Unlike the linear marking scheme, the REM scheme marks packets with a probability which is exponentially related to the link price. To determine the performance of the REM congestion communication mechanism, some assumptions need to be made about the bounds on prices at each link. Let us assume an end-to-end path has l links, and the price d_l at each link is bounded between 0 and d_{max}, so that the maximum aggregate path price is $l.d_{max}$. The aggregate path price $S_i(t)$ is to be communicated to the source with an accuracy of ΔS across the full range of aggregate price 0 to $l.d_{max}$. For a variation ΔS, of path price $S_i(t)$, there is a corresponding permissible variation in the marking rate, ΔM. The number of bits required to communicate the price accurately is maximum when ΔM is minimum.

The minimum ΔM occurs when the path price is maximum, because the exponential marking scheme compresses the most information at the maximum price into a small region of marking rates. The minimum ΔM can thus be obtained:

$$\Delta M = (1-\phi^{-l\cdot d_{\max}}) - (1-\phi^{-l\cdot d_{\max}+\Delta S}) \tag{6.16}$$

To obtain the probability that a price is communicated with said accuracy limits given N packets have been transmitted, ΔM from (6.16) can be substituted into (6.15). However, this is not readily numerically computable. Note that since REM scales a wide range of $S_i(t)$ into smaller range of marking rates $M(t)$, it requires more packets than linear marking to communicate $S_i(t)$ with the same accuracy.

For our purposes, we are more interested in the nature of the REM communication mechanism, than exact models, so a lower bound on number of packets needed to communicate the aggregate price suffices. Like linear marking, the lower bound can be obtained if the marking is assumed to be deterministic. In this case the number of bits required is inversely proportional to the required accuracy ΔM:

$$N_{REM\ marking} \quad \rangle \quad \frac{1}{\Delta M} = \frac{\phi^{l\cdot d_{\max}}}{\phi^{\Delta S}-1} \tag{6.17}$$

Notice that the number of bits required grows exponentially, with (a) the maximum price and (b) with the number of links on the end-to-end path.

6.8.4 MaxNet Packed bit Marking

MaxNet uses a packed bit format to communicate the congestion signal. The congestion price is a binary number stored in the packet. Let us assume that the maximum congestion signal to be communicated is $S_{MaxNet\ Max}$. If the signal $S_i(t)$ needs to be communicated with $\pm\Delta S$ accuracy, then there are at least $\frac{S_{MaxNet\ Max}}{\Delta S}$ levels to be communicated. The number of bits required to communicate this signal, assuming a packed-bit format is:

$$N_{MaxNet} = \log_2 \frac{S_{MaxNet\ Max}}{\Delta S} \tag{6.18}$$

As discussed in previous Sections, the number of bits required to communicate the congestion signal with the same precision for the linear and exponential marking schemes is at least:

$$N_{Independent\ Marking} = \frac{S_{Independent\ Marking\ \ Max}}{\Delta S} \tag{6.19}$$

Since $N_{MaxNet} << N_{independent\ \ Marking}$, MaxNet is able to communicate more congestion information in less bits. Although more bits are required in an individual packet, the congestion signal does not have to be communicated at every packet transmission. Every X^{th} packet may contain the congestion signal field, where X depends on the trade-off between the delay of communicating congestion information and the bandwidth reserved for communicating this information. If the congestion signal is communicated in the first packet transmitted on a connection, then even short connections can be congestion controlled.

Also, unlike the independent marking schemes, the number of bits required to transmit the congestion signal with a given accuracy does not grow with the number of bottlenecked links on the end-to-end path. This is because for MaxNet the congestion signal is not additive with the number of bottleneck links on the end-to-end path.

6.9 Other Max-Min fair strategies

As shown in Section 6.4, MaxNet leads to a Max-Min rate distribution for homogenous demand functions. In this section, we will compare other methods of achieving Max-Min fairness to MaxNet. The two main schools of research into achieving Max-Min fairness have been for the two types of networks, ATM and Internet. We will show that MaxNet is unique in that it achieves Max-Min fairness for a range of finite homogenous demand functions, in a completely distributed

manner and requires no information about flow connection state in routers and switches.

6.9.1 Achieving Max-Min fairness in SumNets

We will begin by a counter-example which proves that SumNets do not lead to a Max-Min fair rate distribution for general homogenous demand functions. Consider a two-link network as in Fig. 6.14. All three sources have the same demand function:

$$r_i(t+1) = r_{\max} e^{-k \cdot S_i(t)} \tag{6.20}$$

where r_{max} is a constant that sets the maximum transmission rate and here r_{max}=15, and k bounds the feedback gain, and here k = 0.008. Each link l has the same, well studied [100] [76], $G(y_l(t),d_l(t))$ process:

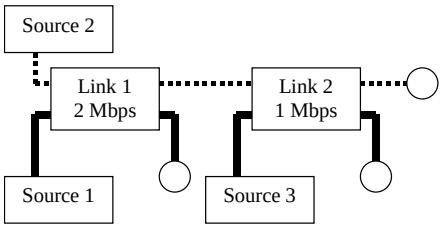
$$d_l(t+1) = d_l(t) + \alpha_l (y_l(t) - C_l) \tag{6.21}$$

where α_l determines the convergence rate and stability and C_l is the target link capacity. From (6.20) and (6.21), the steady state solution for the network can be obtained by solving:

$$r_1 = r_{\max} e^{-k \cdot S_1(t)}, \; r_2 = r_{\max} e^{-k \cdot (S_1(t) + S_2(t))}, \; r_3 = r_{\max} e^{-k \cdot S_2(t)}$$
$$C_1 = r_1 + r_2, C_2 = r_2 + r_3.$$

The solution for the case where for all i, $r_i>0$ is: Source 1 = 1.88 Mbps, Source 2 = 0.11 Mbps and Source 3= 0.88 Mbps. Clearly, a Max-Min allocation is not achieved. Because Source 2 traverses two of the links, its congestion price is the sum of the prices of Links 1 and 2, and its rate must be less than that of Source 3, which only traverses Link 2. (Max-Min solution was presented in Chapter 2).

Fig. 6.14. Global Max-Min Not Achieved for SumNet



As shown by F. Kelly, the rate distribution is such that it maximises the total utility. If achieving Max-Min rate distribution is an objective, the only way to achieve Max-Min rate allocation is to choose utility functions such that the maximum total utility coincides with a Max-Min fair rate distribution. Work performed by Low in [76] and by Kelly in [64] shows two ways of achieving this.

In [64] Kelly proves that Max-Min fairness is approached in the limit of a very specific homogenous utility function:

$$U(x) = -(-\log x)^{\alpha} \quad \alpha \to \infty \tag{6.22}$$

Since a Max-Min rate allocation is approached in the limit as $\alpha \to \infty$, in practice, this solution can only be an approximate Max-Min fair solution. There are a number of practical concerns with approximating Max-Min fairness in this way. As $\alpha \to \infty$, the precision and range of the congestion signal necessary to control the sources also approaches infinity. This means that as $\alpha \to \infty$, and the approximation is closer to Max-Min, the time required to reach the solution also goes up.

In [76] Low *et al.* shows that by appropriately choosing each utility function, Max-Min rate allocation is possible. If each source *l* has a utility function of the form $U(x) = -(-\alpha_l \log x)$, then it is possible to manipulate α_l so that a Max-Min solution is achieved. This requires that each α_l is scaled to align the global utility maximum with the Max-Min fairness criterion. However, this approach implies the use of global information, as the scaling factor for each utility function must be coordinated with every other factor. It is perhaps conceivable that distributed schemes to achieve this are possible, but none have been suggested. With MaxNet you only have to

agree that one type of demand function will always be used, and no further global coordination is required. Furthermore, sources may be added and removed from the network without having to reconfigure all of the other sources on the network.

6.9.2 ATM Max-Min flow control strategies

Extensive research into achieving Max-Min optimality has been performed for ATM networks. Indeed, Max-Min fairness has been specified for the best-effort class of traffic in ATM by the ATM forum in the TN 4.0 specifications [117]. However, the ATM network has some fundamental differences to the Internet which make the Max-Min solutions developed in the ATM world not applicable to the Internet.

The primary difference between ATM and the Internet is that ATM is connection oriented. Flows secure connections called Virtual Circuits VC on their end-to-end communication path. Each ATM switch on the path is able to identify the establishment and release of a VC, as there is explicit signaling protocol for this. However, on connection-less networks such as the Internet, each packet is independent of others, and the concept of connection does not exist inside the network. Connections state only exists in source and destination hosts, and the network does not assume there is any relationship between packets that it transmits.

In ATM, storing and processing information about each flow is possible because the number of VCs in an ATM switch is significantly less than the number of TCP connections in an Internet Router. An IP router may have hundreds of thousands or millions of connections across it, whereas an ATM switch may have many thousands of connections. This stems from the fact that on average, VCs are much longer lived and used to carry more traffic than a typical TCP connections. Indeed, when ATM is used to carry IP traffic, a single VC is used to carry an aggregate of many TCP connections. Most TCP connections are very short, because the bulk of the Internet Traffic is WWW access, where a single TCP connection transmits one HTML page or one image within a HTML page. On the other hand, ATM VCs were designed to provide provisioned capacity to flow aggregates.

Congestion control schemes which require that IP routers store or process information for each TCP flow (per-flow schemes) are not scalable to be applied on the Internet, because of two reasons: 1) the processing and storage required 2) the difficult of identifying flows. Gigabit core IP routers struggle to perform simple task like address lookup, to decide which port to forward a packet to with a particular IP address. The type of per-flow processing and storage possible in ATM is not practicable on Internet devices.

Many of the ATM Max-Min proposals require per-VC information storage and processing the switch. These schemes may store such information as the current rate of the VC and whether it is bottlenecked at this switch. These include [4],[2],[54],[59],[60],[61],[67],[118]. The work of A.Charny [2] is representative of this class of Max-Min algorithms. Her framework has per-flow-state in the link, as stated in [2] pp 22-23, p22 “Each Link maintains a list of users”, and on p23 “For Each user the link stores its last seen stamped rate”. These types of algorithms for ATM typically compute the rate allocation of a flow in the switch, and communicate the rate allocation back to the source. The switches need to have substantial intelligence, and this cannot be translated to an Internet like network.

More scalable ATM schemes do not require per-VC information storage in the switch, however they require per-VC signaling and information processing in the switch [118]. The ASAP protocol presented in [118] does not keep per-VC information at the switch, however it does requires that the number of VC connections present be counted. The state of the number of constrained flows at a link, and their total bandwidth needs to be maintained. This relies on the network to be able to detect flows establishing and releasing connection. ATM provides signalling for this, but this does not fit into the Internet connectionless architecture, where TCP/IP flows do not signal a start or end of transmission to the network. The processing of packets to detect the start and end of flows on the Internet in the core network equipment is not feasible.

6.10 Application of MaxNet

Although it is descended from the Internet, so far we have treated MaxNet as a new and separate kind of network. In this section, we will discuss some applications of MaxNet that motivate its integration into the Internet. We will also discuss how MaxNet may inter-operate with the Internet Protocol. Application of MaxNet to the existing Internet Protocol is not trivial, but because of its single bottleneck properties, it is a desirable transport control protocol in a number of scenarios where TCP is currently used and its performance is not ideal.

The key application of MaxNet is in multi-hop networks, where the number of hops on an end-to-end path is large. As shown in Section 6.9.1, and in literature [6], as the SumNet network size increases, the share of capacity that TCP sessions crossing a large number of bottleneck links receive, declines. In SumNets, there is an inherent disadvantage for flows crossing many hops. This is because such flows use more network resources, so a utility optimal solution penalizes these flows more. Max-Min or Weighted-Max-Min fairness treats users equally regardless of the number of hops they are away from the destination. This is an important property if the QoS of a user is to be maintained regardless of their position in the network, particularly for applications such as multi-hop ad-hoc wireless networks.

In the following section, we give an overview of possible architectures for deploying MaxNet on the existing Internet Protocol network. We explore a partial deployment, where only some network components are aware of MaxNet.

6.10.1 Partial Deployment – MaxNet / IP interaction

Logistically a complete deployment on an existing network as large as the Internet is not practicable. Each router, switch, and source implementation on all computers would have to be upgraded. However, for applications such as multi-hop ad hoc wireless networks, MaxNet may be deployed partially across a segment of the network within which flows enjoy Max-Min fairness. In this case, MaxNet can be viewed as a link-layer protocol which carries IP as depicted in Fig. 6.15.

Fig. 6.15 MaxNet, IP Encapsulation

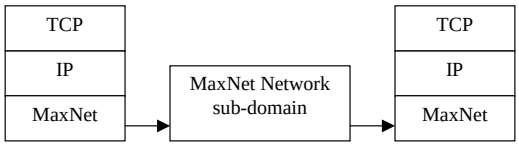
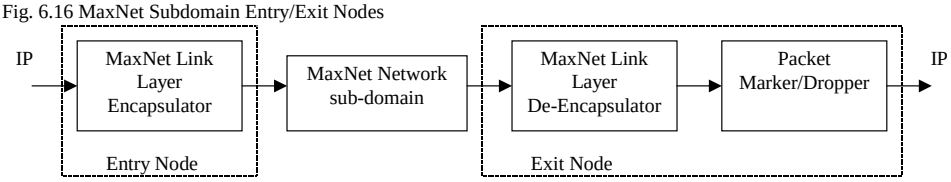


Fig. 6.16 MaxNet Subdomain Entry/Exit Nodes



At the entry into such network domains, the IP packet is encapsulated into a packet that contains congestion signal information. Within the MaxNet network domain, congestion signalling is performed using MaxNet process. No IP packet marking or dropping is performed within the MaxNet domain, (unless a queue overflows in transient conditions). At the exit of the MaxNet Network domain, the IP packets are removed from their encapsulation. The congestion signal inside the packet is translated from the MaxNet packed-bit format into packet marking/dropping by using the congestion signal value as a marking/dropping probability.

Simulations using NS-2 were performed to study MaxNet and IP interaction. The network in Fig. 6.17 was simulated. Sources 1 to 8 each generate 10 TCP/IP connections, with connections from sources 1-4 terminating at D_1, and connections from sources 5-8 generating cross traffic to destinations D_2-D_5 respectively. The MaxNet sub-domain consists of links 1,2,3,9,10,11,12, and links 5,6,7,8,4 are the exit nodes which remove the MaxNet encapsulation and mark/drop the IP packets according to the congestion level in the MaxNet packet. The target utilisation of all links was 0.9.

Two experiments were performed with different link delay parameters, as shown in Fig. 6.18. Each experiment consisted of one simulation of a network with a MaxNet sub-domain as described in the previous paragraph, and one simulation of the same network with only Internet independent packet marking at all links in the network.

In experiment 1, the delays between sources 1,2,3,4,8 and their destination were equalized, such that all sources sharing link 4 had a source-to-destination propagation delay of 20 ms. The throughput of these sources over a 10 sec period was measured, as shown in Fig 6.19. Note both MaxNet and Internet congestion control, achieved the target throughput of 0.9 x 10Mbps x 10 sec = 90 Mbits across link 4. However, the allocation of flow rates for both schemes was different. Although the MaxNet rates were not exactly Max-Min fair, the ratio of the highest to lowest rate was only 20.00/16.65 = 1.20. A Max-Min fair allocation would be that all flows through link 4 are equal. For the Internet the rates were more disparate, such that this ratio was 26.24/10.92 = 2.4.

Although MaxNet produces a rate allocation significantly closer to Max-Min fairness than Internet independent marking when it is used to carry TCP/IP, it cannot achieve exact Max-Min fairness with TCP/IP because of TCP's sensitivity to RTT. As described in Chapter 3, Section 3.5, TCP rate is not just dependent on the congestion signal, but on the RTT as well. MaxNet can remove the sensitivity to the number of bottleneck links on the end-to-end path, but it cannot remove the dependence of RTT from TCP. For this a new source algorithm would be required. As discussed in [100] it is possible to have stable source algorithms whose steady state flow rate is independent of the RTT. However, even with the TCP algorithm, MaxNet can reduce the gross discrepancies between rates of sources which pass through a different number of bottleneck links. This is shown in experiment 2.

In this experiment, the delays of connections going through link 4 are not equalised, as links 9-12 now all have 5 ms delay, resulting in a spread of delays from 10 to 25 ms for connections passing link 4, as shown in Fig. 6.18. Fig 6.20 shows the results for this experiment. Note that the ratio of the highest to lowest connection throughputs for Internet congestion control is large: 36.02/6.86 = 5.25. For MaxNet control this is significantly less 24.9/13.79 = 1.80. This indicates that MaxNet can be give significantly more equal rate allocation, than TCP/IP with independent packet marking/dropping, even across different path delays.

Fig. 6.17 MaxNet / IP Interaction Simulated Network Topology

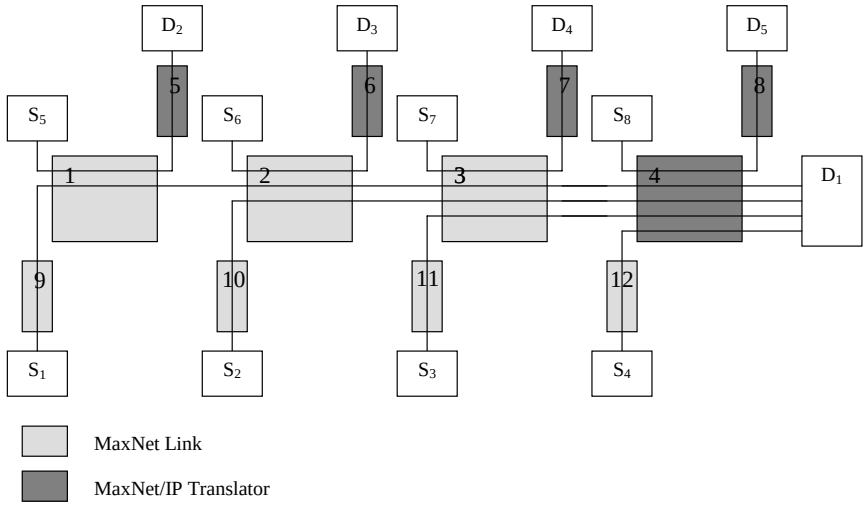


Fig. 6.18 Network Parameters

Experiment 1			**Experiment 2**		
Link	**Capacity**	**Delay (ms)**	**Link**	**Capacity**	**Delay**
1,2,3,4	10 Mbps	5	1,2,3,4	10 Mbps	5
9	100 Mbps	0	9	100 Mbps	5
10	100 Mbps	5	10	100 Mbps	5
11	100 Mbps	10	11	100 Mbps	5
12	100 Mbps	15	12	100 Mbps	5
5,6,7	100 Mbps	5	5,6,7	100 Mbps	5
8	100 Mbps	15	8	100 Mbps	15

Fig. 6.19 Throughput across Link 4 - Experiment 1

MaxNet		**Internet**	
Source	**Throughput (Mbits)**	**Source**	**Throughput (Mbits)**
1	16.65	1	10.92
2	16.25	2	11.48
3	18.15	3	15.28
4	19.09	4	26.19
8	20.00	8	26.24

Total	90.14	Total	90.11

Fig. 6.20 Throughput across Link 4 - Experiment 2

MaxNet		**Internet**	
Source	**Throughput (Mbits)**	**Source**	**Throughput (Mbits)**
1	13.79	1	6.86
2	15.01	2	8.9
3	18.93	3	16.5
4	24.9	4	36.02
8	17.31	8	22.12
Total	89.94	Total	90.04

6.11 Conclusion

In this chapter, we introduced MaxNet, which is an Internet like congestion control architecture. We analysed its fairness properties and proved that for well behaved homogenous sources it results in Max-Min fairness. Like the Internet, MaxNet is a distributed architecture that requires no global information or per-flow state in the link. We proved that a MaxNet network is stable and robust, and its stability is invariant to the network size or topology. We also showed that MaxNet has these scalability properties without needing to communicate the number of bottleneck links on a transmissions path.

The broader picture of this contribution is that it has introduced an option in the design of Internet like networks. Previously, SumNets were shown to achieve maximum utility optimality. Now, Internet like networks can achieve Max-Min fairness with MaxNet. The trade-off between maximum utility or Max-Min fairness is a technical, commercial and political decision.

Bibliography

[1] "CISCO Switches Website", http://www.cisco.com/warp/public/44/jump/switches.shtml: CISCO Ltd., 2002.

[2] A.Charny, D.Clark, and R.Jain, "Congestion Control with Explicit Rate Indication" Proceedings of ICC 95, 1995. pp. 1954-1963.

[3] R. G. Addie, M. Zukerman, and T. D. Neame, "Broadband traffic modeling: simple solutions to hard problems" *IEEE Communication Magazine*, pp. 88-95, 1998.

[4] A. Arulambalam, X. Chen, and N. Ansari, "Allocating Fair Rates for Available Bit Rate Service in ATM Networks" in *IEEE Communication Magazine*, 1996, pp. 92-100.

[5] S. Athuraliya, D. Laspsley, and S. H. Low, "An enhanced random early marking algorithm for Internet flow control" Proceedings of Infocom, Israel, 2000.

[6] S. Athuraliya and S. Low, "Optimization Flow Control II: Implementation" *Submitted for publication*, 2000.

[7] S. Athuraliya and S. H. Low, "Optimization Flow Control with Newton-Like Algorithm" *Journal of Telecommunication Systems*, vol. 15, pp. 345-358, 2000.

[8] S. Athuraliya, "Caltech Netlab Website - REM NS implementation", http://netlab.caltech.edu, 2001.

[9] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active Queue Management" *IEEE Network*, vol. 15, pp. 48-53, 2001.

[10] F. Baumgartner, T. Braun, and P. Habegger, "Differentiated Services: A New Approach for Quality of Service in the Internet" Proceedings of IFIP TC-6 Eighth Conference on High Performance Networking HPN'98, Vienna, Austria, 1998. pp. 255-273.

[11] D. Bertsekas and R. Gallager, *Data Networks*, vol. A: Prentice Hall, 1987.

[12] D. A. Bini and B. Meini, "On the solution of a nonlinear matrix equation arising in queueing problems" *SIAM J. Matrix Anal. Appl*, vol. 17, pp. 906-926, 1996.

[13] D. A. Bini and B. Meini, "Improved cyclic reduction for solving queueing problems" *Numerical Algorithms*, vol. 15, pp. 57-74, 1997.

[14] D. A. Bini, L. Gemignani, and B. Meini, "Computations with infinite Toeplitz matrices and polynomials" *Linear Algebra Appl.*, 2001.

[15] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "Architecture for Differentiated Services" The Internet Society RFC 2475, 1998.

[16] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet" The Internet Society, Request for Comments RFC 2309, April 1998.

[17] R. Braden, "Requirements for Internet Hosts -- Communication Layers" Internet Engineering Task Force RFC 1122, Oct 1989.

[18] L. S. Brakmo and L. L. Peterson, "TCP vegas: end to end congestion avoidance on a global internet." *IEEE J.Select. Areas Commun.*, vol. 13, pp. 1465-1480, 1995.

[19] I. Busse, B. Deffner, and H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP" *Computer Communications*, 1996.

[20] D. M. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks" *Computer Networks*, vol. 17, pp. 1-14, 1989.

[21] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for Web Traffic" *SIGCOMM*, pp. 139-150, 2000.

[22] S. T. Chuang, A.Goel, N.McKeown, and B.Prabhakar, "Matching output queueing with combined input and output queueing" *IEEE J.Select. Areas Commun.*, vol. 17, pp. 1030-1039, 1999.

[23] CISCO, "Distributed Weighted Random Early Detection", http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.htm: CISCO Ltd., 1998.

[24] CISCO, "Class-Based Weighted Fair Queueing", http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t5/cbwfq.htm: CISCO Ltd., 1998.

[25] CISCO, "Strategies to Protect Against Distributed Denial of Service (DDoS) Attacks", http://www.cisco.com/warp/public/707/newsflash.html, 2000.

[26] CISCO, "DiffServ- The Scalable End-to-End QoS Model", http://www.cisco.com/warp/public/cc/pd/iosw/ioft/iofwft/prodlit/difse_wp.htm: Cisco Systems Inc., 2001.

[27] CISCO, "Weighted Random Early Detection on the Cisco 12000 Series Router", http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/ios112p/gsr/wred_gs.htm: CISCO Ltd., 2002.

[28] J. Crowcroft and P. Oechslin, "Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP" *ACM Computer Communications Review*, vol. 28, pp. 53-67, 1998.

[29] W. Fang, "Differentiated Services: Architecture, Mechansims and an Evaluation" in *Computer Science*: Princeton University, 2000, pp. 165.

[30] M. A. Farooq and L. Tassiulas, "Balanced-RED: An algorithm to achieve Fairness in the Internet" University of Maryland at College Park, College Park CSHCN T.R. 99-9, March 8 1999.

[31] S. Farrell, "IPN Interplanetary Internet Project", http://www.ipnsig.org/home.htm: IPN Special Interest Group, 2003.

[32] N. Feamster, D. Bansal, and H. Balakrishnan, "On the interactions between layered quality adaptation and congestion control for streaming video" Proceedings of 11th International Packet Video Workshop, 2001.

[33] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: A New Class of Active Queue Management Algorithms" UM CSE-TR-387-99, 1999.

[34] W. Feng and J. Rexford, "Performance evaluation of smoothing algorithms for transmitting pre-recorded variable bit -rate video" *IEEE Trans. Multimedia*, pp. 302-313, 1999.

[35] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness" *INFOCOM*, 2001.

[36] Ferguson, "Denial of Service (DoS) Attack Resources", http://www.denialinfo.com/, 2000.

[37] V. Firoiu and M. Borden, "A Study of Active Queue Management for Congestion Control" *INFOCOM 2000*, pp. 1435-1444, 2000.
[38] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance" *ACM Transactions on Networking*, vol. 1, pp. 397-413, 1993.
[39] S. Floyd and K. Fall, "Router Mechanisms to Support End-to-End Congestion Control", LBL Technical Report. 1997.
[40] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet" *under submission*, 1998.
[41] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management" 2001.
[42] C. H. Foh, B. Meini, B. Wydrowski, and M. Zukerman, "Modeling and Performance Evaluation of GPRS" Proceedings of VTC 2001, Rhodes, Greece, 2001. pp. 2108-2112.
[43] E. Fulp and D. Reeves, "The Fairness and Utility of Pricing Network Resources Using Competitive Markets" *Networks Journal*, 2000.
[44] A. Gersho, "Advances in speech and audio compression" *Proceedings of The IEEE*, vol. 82, pp. 900-918, 1994.
[45] R. Gibbens and F. Kelly, "Resource pricing and the evolution of congestion control" *Available at http://www.statslab.cam.ac.uk/~frank/evol.html*, 1998.
[46] R. J. Gibbens, "The use of games to assess user startegies for differential Quality of Service in the Internet" Cambridge November 1999.
[47] F. Guillemin and P. Robert, "Limit results for Markovian models of TCP" Proceedings of Globecom'2001, San Antonio, 2001.
[48] P. Gupta, "Scheduling in Input Queued Switches: A Survey" *unpublished manuscript.*, 1996.
[49] G. Hasegawa, T. Matsuo, M. Murata, and H. Miyahara, "Comparisons of packet scheduling algorithms for fair service among connections on the Internet" *Journal on High Speed Network*, 2000.
[50] E. Hashem, "Analysis of random drop for gateway congestion control" Laboratory for Computer Science, MIT, Cambridge MA LCS TR-465, 1989.
[51] C. He, B. Meini, and N. H. Rhee, "A shifted cyclic reduction algorithm for QBD's" *SIAM J. Matrix Anal. Appl.*, 2001.
[52] H. Heffes and D. M. Lucantoni, "A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance" *IEEE JSAC*, vol. SAC-4, 1986.
[53] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "A Control Theoretic Analysis of RED" Proceedings of Infocom 2001, Anchorage Alaska, 2001.
[54] Y. Hou, H. Tzeng, and S. Panwar, "A Generalized Max-Min Rate Allocation Policy and Its Distributed Implementation Using the ABR Flow Control Mechansim" Proceedings of Infocom 98, San Francisco, 1998.
[55] HP-Ltd., "HP ProCurve Networking Website", http://www.hp.com/rnd/index.htm: Hewlett Packard, 2002.
[56] J.F.Nash, "Non-Cooperative Games" *Annals of Mathematics*, vol. 54, pp. 286-295, 1951.
[57] V. Jacobson, "Congestion Avoidance and Control" *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, vol. 18, pp. 314-329, 1988.
[58] Jain, R. Ramakrishnan, and Chiu, "Congestion avoidance in computer networks with connectionless network layer." Digital Equipment Corporation DEC-TR-506, 1987.

[59] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan, "ERICA Switch Algorithm: A Complete Discription" *ATM Forum*, pp. 96-1172, 1996.

[60] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "An Efficient Rate Allocation Algorithm for ATM Networks Providing Maxmin fairness" Proceedings of 6th IFIP Int'l Conf. High Performance Networking, 1995. pp. 143-154.

[61] L. Kalampoukas, "Congestion Management in High Speed Networks": University of California at Santa Cruz, April 1997.

[62] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability." *Journal of the Operational Research Society*, vol. 49, pp. 237-252, 1998.

[63] F. Kelly, "Models for a self-managed Internet" *Philosophical Transactions of the Royal Society*, vol. A358, pp. 2335-2348, 2000.

[64] F. P. Kelly, "Charging and rate control for elastic traffic" *European Transactions on Telecommunications*, vol. 8, pp. 33-37, 1997.

[65] P. Key and L. Massoulie, "User policies in a network implementing congestion pricing" Proceedings of Workshop on Internet Service Quality Economic (MIT), 1999.

[66] P. Key and D. McAuley, "Differential QoS and pricing in networks: where flow-control meets game theory" *IEE Proceedings Software*, vol. 146, pp. 39-43, 1999.

[67] Y. Kim, "A Non-Per-VC Accounting Max-Min Protocol for ABR Flow Control (ASAP) and Multiple-Time-Scale Extension": University of California Irvine, 1999.

[68] L. Kleinrock, *Queuing Systems*, vol. 1: Theory. New York: John Wiley & Sons, 1975.

[69] H. Kong, N. Ge, F. Ruan, and C. Feng, "A Control Theoretic Analysis of the AQM algorithm GREEN" *unpublished manuscript, E&E Dept. Tsinghua Univ. Beijing 100084, P.R.China*, 2002.

[70] H. Kong, N. Ge, F. Ruan, C. Feng, and P. Fan, "A Nonlinear Model of the AQM algorithm GREEN" *IEICE*, 2002.

[71] E. Kuumola, "Analytical Model of AF PHB Node in DiffServ Network" Networking Laboratory, Helsinki University of Technology, Helsinki 2001.

[72] K. Laevens, P. Key, and D. McAuley, "An ECN-based end-to-end congestion control framework: experiments and evaluation" Microsoft Research MSR-TR2000 -104, October 2000.

[73] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "Bounds on Average Delays and Queue Size Averages and Variances in Input-Queued CellBased Switches" Proceedings of IEEE INFOCOM 2001, 2001. pp. 1095-1103.

[74] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "On the stability of input-queued switches with speed-up" *IEEE ACM Transactions on Networking*, vol. 9, pp. 104-118, 2001.

[75] C. Liu and R. Jain, "Improving Explicit Congestion Notification with the Mark-Front Strategy" *Computer Networks*, vol. 35, pp. 185-201, 2001.

[76] S. Low and D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence" *IEEE/ACM Transactions on Networking*, vol. 7, pp. 861-875, 1999.

[77] S. Low, "A Duality Model of TCP and Queue Management Algorithms" Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Monterey, CA, 2000.

[78] S. Low, "Network Flow Control" *Tutorial Paper, Publication status unknown*, 2002.
[79] S. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a Scalable Control" Proceedings of Infocom, 2002.
[80] R. Makkar, I. Lambadaris, and e. al, "Empirical Study of Buffer Management Scheme for DiffServ Assured Forwarding PHB," Proceedings of, Las Vegas, Nevada, 2000.
[81] J. E. Marshall, H. Gorecki, K. Walton, and A. Korytowski, *Time-Delay Systems - stability and performance criteria with applications*. New York: Ellis Horwood, 1992.
[82] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options." IETF RFC 2018, April 1996.
[83] Matrix.Net, "The Internet Weather Report", http://www.mids.org/weather/: Matrix Net, 2001.
[84] A. Mauldin, "Global Internet Backbone Growth Slows Dramatically", http://www.telegeography.com/press/releases/2002/16-oct-2002.html: TeleGeography, Inc., 2002.
[85] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons Not to Deploy RED" Proceedings of 7th. International Workshop on Quality of Service IWQoS'99, 1999. pp. 260-262.
[86] M. May, T. Bonald, and J.-C. Bolot, "Analytic Evaluation of RED Performance" Proceedings of INFOCOM, 2000. pp. 1415-1424.
[87] D. McDysan, *QoS & Traffic Management in IP & ATM Networks*: McGraw Hill, 2000.
[88] V. Misra, W. B. Gong, and D. Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED." Proceedings of ACM SIGCOMM'00, Stockholm, Sweden, 2000.
[89] J. Nagle, "Congestion Control in IP/TCP Internetworks" DDN Network Information Center, Menlo Park, CA RFC-896, Jan. 1984 1984.
[90] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Baltimore, MD: The Johns Hopkins University Press, 1981.
[91] K. Nichols, S. Blake, F. Baker, and D. Black, "RFC 2474 definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers" *Internet Engineering Task Force*, 1998.
[92] Nua-Internet-Surveys, "How many online?" http://www.nua.ie/surveys/how_many_online/: NUA.COM Internet Surveys, 2002.
[93] E. Nyberg, S. Aalto, and J. Virtamo, "Relating Flow Level Requirements to DiffServ Packet Level Mechanisms" Helisinki University of Technology, Heliskini COST279 TD(01)04, October 2001.
[94] Open-Source-Project, "The Network Simulator NS2", http://www.isi.edu/nsnam/ns/, 2 ed: DARPA and NSF funded, 2002.
[95] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED" Proceedings of INFOCOM, 1999. pp. 1346-1355.
[96] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP Throughput: A Simple Model and its Empirical Validation" Proceedings of ACM SIGCOMM '98, 1998. pp. 303-314.
[97] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol" Proceedings of International Workshop on

Network and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, 1999.

[98] F. Paganini, "Flow control via pricing: a feedback perspective" Proceedings of Allerton, Monticello, IL, 2000.

[99] F. Paganini, "On the stability of optimization-based flow control" Proceedings of American Control Conference, 2001.

[100] F. Paganini, J. C. Doyle, and S. H. Low, "Scalable Laws for Stable Network Congestion Control" Proceedings of CDC, Orlando, FL, 2001.

[101] R. Pan, B. Prabhakar, and K. Psounis, "CHOKE, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation." Proceedings of INFOCOM, 2000. pp. 942-951.

[102] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case" *IEEE/ACM Transactions on Networking*, vol. 1, pp. 344-357, 1993.

[103] H. R. a. K. Park, "Toward a Theory of Differentiated Services" Proceedings of IEEE/IFIP Eighth International Workshop on Quality of Service, 2000.

[104] P. Potter and M. Zukerman, "Analysis of a discrete multi-priority queueing system involving a central shared processor serving many local queues" *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 194-202.

[105] K. K. Ramakrishnan, S. Floyd, and D.Black, "The Addition of Explicit Congestion Notification (ECN) to IP" Internet Engineering Task Force RFC 3168, September 2001.

[106] J. Rawls, *A Theory of Justice*: The Belknap Press of Harvard University Press, 1971.

[107] R. Rejaie, M. Handley, and D. Estrin, "Quality adaption for congestion controlled video playback over the Internet" *ACM SIGCOMM Computer Communications Review*, vol. 29, pp. 189-200, 1999.

[108] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet" *INFOCOMM 99*, 1999.

[109] J. F. d. Rezende, "Assured service evaluation" *Proceedings of IEEE Globecomm'99*, Dec. 1999.

[110] D. Rossi, C. Casetti, and M. Mellia, "A Simulation Study of Web Traffic over DiffServ Networks" Proceedings of IEEE Globecom 2002, Taipei, TW, 2002.

[111] V. Singhal and R. Le, "High-Speed Buffered Crossbar Switch Design Using Virtex-EM Devices" Xilinx Ltd., Application Note 2000.

[112] W. Stevens, *TCP/IP Illustrated, Vol.1 The Protocols*, 10 ed: Addison-Wesley, 1997.

[113] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms." IETF RFC2001, 1997.

[114] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks" *SIGCOMM*, 1998.

[115] I. Stoica and H. Zhang, "Providing Guaranteed Services without Per Flow Management" Proceedings of SIGCOMM, 1999. pp. 81-94.

[116] K. Thompson, G. J.Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics" *IEEE Network*, 1997.

[117] Traffic-Management-Working-Group, "Traffic Management Specification Version 4.0" The ATM Forum April 1996.

[118] W. Tsai, Y. Kim, and L. Hu, "ASAP: A Non-Per-VC Accounting Max-Min Protocol for ABR Flow Control with Optimal Convergence Speed" *IEEE SICON 98*, pp. 139-153, 1998.

[119] G. d. Veciana, T. Konstantopoulos, and T.-J. Lee, "Stability and performance analysis of networks supporting elastic services" *IEEE/ACM Transactions on Networking (TON)*, vol. 9, pp. 2-14, 2001.

[120] J. Wang, A. Tang, and S. H. Low, "Maximum and asymptotic UDP throughput under CHOKe" *Submitted for publication*, 2002.

[121] W. Weiss, J. Heinanen, F. Baker, and J. Wroclawski, "Assured Forwarding PHB Group" The Internet Society RFC 2597, June 1999.

[122] J. Wroclawski, "The Use of RSVP with IETF Integrated Services" IETF September 1997.

[123] B. Wydrowski and M. Zukerman, "GREEN: An Active Queue Management Algorithm for a Self Managed Internet" Proceedings of ICC 2002, New York, 2002. pp. 2631-2635.

[124] R. H. Zakon, "Hobbes' Internet Timeline v5.6", http://www.zakon.org/robert/internet/timeline/, 5.6 ed: Robert Hobbes, 2002.

[125] B. Zheng and M. Atiquzzaman, "DSRED: A New Queue Management Scheme for Next Generation Networks" Proceedings of LCN 2000: The 25th Annual IEEE Conference on Local Computer Networks, Tampa, Florida, 2000. pp. 242-251.

[126] B. Zheng and M. Atiquzzaman, "DSRED: Improving Performance of Active Queue Management over Heterogeneous Networks" Proceedings of ICC 2001: International Conference on Communications, Helsinki, Finland, 2001. pp. 2375-2379.

[127] B. Zheng and M. Atiquzzaman, "Low Pass Filter/Over Drop Avoidance (LPF/ODA): An algorithm to improve the performance of RED gateways" University of Oklahoma CS-TR-01-001, February 2001.

[128] C. Zhu, O. W.W.Yang, J. Aweya, M. Ouellette, and D. Y. Montuno, "A Comparison of Active Queue Management Algorithms Using OPNET Modeler" *OPNET 2002*, 2002.

[129] T. Ziegler, U. Hofmann, and S. Fdida, "RED+ Gateways for detection and discrimination of unresponsive flows" December 1998.

[130] A. Ziviani, J. F. Rezende, and O. C. M. B. Duarte, "Towards a Differentiated Services Support for Voice Traffic" *GLOBECOM'99*, pp. 59-63, 1999.

[131] M. Zukerman, "Applications of matrix-geometric solutions to the analysis of the bursty data queue in a B-ISDN switching system" *GLOBECOM '88*, vol. 3, pp. 1635-1639, 1988.