Available Bit Rate Services in ATM Networks

by

Nasir Ghani

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical Engineering

Waterloo, Ontario, Canada, 1997

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This thesis is concerned with designing a comprehensive strategy for supporting the ABR traffic class in ATM networks. This class has been defined for reliable data service support in future high speed networks. Since data traffic is relatively delay tolerant, the use of feedback control schemes is appropriate. Among other things, it is necessary that the proposed algorithms provide fair resource (bandwidth) distribution among competing connections and exhibit good scalability to many different network scenarios.

Although a number of ABR flow control proposals have appeared in the literature, some important and crucial issues still require further investigation. Some of these include operation in dynamic environments, containment of transient congestion effects, and provision of MCR guarantees. Hence there is still a need for designing improved algorithms which address the above issues effectively.

The proposed EDERA (*enhanced distributed explicit rate control*) flow control algorithm is designed to address the needs for ABR data users in ATM networks. The algorithm complies with the ATM Forum guidelines and implements the MCR-plus-equal-share bandwidth fairness criterion. Effective capacity tracking algorithms coupled with improved congestion indicators allow the scheme to function in dynamic network environments. A simple, robust rate allocation algorithm is used to achieve fair bandwidth distribution. Simulation results show that the scheme performs very well in many scenarios and outperforms the draft EPRCA (*enhanced proportional rate control algorithm*) proposal in the ATM Forum. Scalability is good and queue sizes are also properly controlled.

To further improve ABR services integration with other traffic types, a complete cell-level scheduling strategy is also proposed. Namely, a hierarchical approach called *hierarchical fair queueing* (HFQ) is used to ensure that the throughput guarantees for data traffic are met along with the delay guarantees of real-time traffic. Numerical results show that the scheme gives improved delay performance for real-time connections over a non-hierarchical strategy.

The convergence behaviour of the algorithm (under some fixed network conditions) is analyzed, and the conditions for non-oscillatory steady-state rate behaviour are also formulated. Furthermore, bounds on the worst case transient queue buildups are also derived in order to help in the buffer dimensioning problem. However, the results indicate that the complex behaviour of the scheme makes it difficult to derive tight bounds in the generic network case.

Acknowledgements

During the last three years, there have been many wonderful people who have helped me along the way and made this journey a memorable one. To mention all of them would be difficult, but not to mention some would be inexcusable.

To start, I would like to thank my supervisor Professor Jon Mark for his continual guidance and support throughout my study. Truly, your encouragement, warm words, and dedication are inspirational. I will always remember your good will and generosity towards me, without which the completion of this degree would not have been possible. I am indebted to you. Thank you kindly and God bless you.

I would also like to thank the members of my PhD committee for their thoughtful comments and suggestions regarding my research work, namely Dr. Jim Field, Dr. Tomas Kunz, Dr. San-Qi Li, and Dr. George Kesidis. I appreciate your taking the time to carefully review my work. In addition, I would like to thank the Department Chair, Dr. Sujeet Chaudhri, for his long-standing, sincere friendship and timeless advice. I would not have returned to pursue this degree without your strong encouragement.

Other members of the University of Waterloo staff also deserve specific mention. In particular, I would like to thank Mr. Kim Martin for his assistance in obtaining various technical documents. To members of the department's administrative staff, I want to express my gratitude for their help during my studies. These include Mrs. Wendy Boles, Ms. Gini Ivan-Roth, Ms. Jenniffer Werth, Mrs. Maureen Searle, and Mrs. Donna O'Brecht. Also, special thanks to Mrs. Kay Singh for her steady optimism and reassuring words. Thank you all for cheering me up during my stay.

I would also like to thank many of my colleagues for their insightful advice and warm encouragement. In particular, I want to thank various members of Professor Mark's group, namely Mr. Michael Cheung, Dr. Robert Lehr, Dr. Meenaradchagan Vishnu, Dr. Majid Barazande-Pour, Dr. Atushi Yamada, Mr. Tung Chung Wong, Mr. Vincent Wong, and Mr. James Qui. Your humour and overall support were well appreciated. To past members, Dr. Byoung-Joon Lee and Dr. Jing-Fei Ren, also, many thanks for your vital guidance during the start of my doctoral studies.

To my personal friends, I will not forget the friendships we made and the good times we had. I reserve specific mention for my friends Mr. Salim Manji, Mr. Shamrez Shaikh, and Mr. Peter Sellmer for their sincere friendships. Thank you for your good will, dedication, and time.

The financial support I received from various funding agencies was invaluable in helping during my studies. In particular, I want to acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC), Association of Universities and Colleges of Canada (AUCC), Canadian Council of Professional Engineers (CCPE), Manulife Financial, and the Department of Electrical and Computer Engineering. Thank you for making my student life much easier.

To my family, my endless gratitude for all their moral support and unwaivering belief in me. To my brother, Nur, and sister, Mansura, I will always recall your kind, uplifting words. To my parents, my sincere gratitude for your love and support.

Finally, I want to thank God for answering my prayers and guiding me through my difficulties. Through many times of doubt, my faith has been re-affirmed.

Dedication

To My Parents,

My deepest appreciation for your love, prayers, and support.

Contents

List of Tables

List of Figures

List of Symbols

MCR	Minimum cell rate of an ABR connection
PCR	Peak cell rate of an ABR connection
ACR	Allowed cell rate of an ABR connection
$SASR$	Segmental average source rate of an ABR connection
AIR	Additive rate increment at source end systems
N_{rm}	Inter-RM cell spacing at source end-system
ER_{cell}	Explicit rate value in backward RM cell
ER_{sw}	Explicit rate fair share computed at network switch
ER_{min}	Minimum rate guarantee at network switch
μ	Output link transmission capacity
T_{sw}	Measurement interval at network switch
ρ	Fair share utilization threshold
β	Fixed bandwidth scaling constant during congestion
γ	Variable bandwidth scaling constant during congestion
QT	Queue congestion threshold for fixed bandwidth scaling
ΔQ	Queue step-size for variable bandwidth scaling
$C_a(k)$	Instantaneous available capacity estimate for ABR traffic

$\overline{C_a(k)}$	Smoothed available capacity estimate for ABR traffic
$C_u(k)$	Usable capacity for ABR traffic
$C_{\mathcal{I}}$	Aggregate capacity usage by bottlenecked connections at a link
$C_{\overline{\mathcal{I}}}$	Aggregate capacity usage by non-bottlenecked connections at a link
$C_{\mathcal{T}}$	Aggregate capacity usage by all connections at a link
$C_{\mathcal{I}}^{MCR}$	Aggregate MCR guarantee of bottlenecked connections at a link
$q(k)$	Queue length at output link buffer
$z(k)$	Binary congestion flag
N_{ABR}	Number of ABR connections at a link
N_{VBR}	Number of non-ABR connections at a link
ω_i	Minimum rate requirement of i-th VBR connection
ψ_i	Minimum rate requirement of i-th ABR connection (i.e., MCR)
Ψ	Minimum rate allocation to aggregate ABR flow (HFQ scheduler)
Θ	Rate allocation to aggregate ABR flow (HFQ scheduler)
Λ	Rate allocation to aggregate VBR flow (HFQ scheduler)
$v(t)$	Global virtual time function of HFQ scheduler (at link server)
$\mathcal{N}$	Set of network switches
$\mathcal{L}$	Set of network links
$G(\mathcal{N}, \mathcal{L})$	Graph representation of network
δ_f^i	Forward propagation delay for connection i
δ_b^i	Backward propagation delay for connection i
δ^i	Roundtrip propagation delay for connection i
T_f^i	Forward notification time for connection i

T_b^i	Backward notification time for connection i
$\mathcal{A}_j$	Set of connections traversing link j
d_j	Propagation delay for link j
$\mathcal{D}_i$	Delay grouping for connection i at a link
$\mathcal{E}_i$	Express grouping for connection i at a link
$\mathcal{B}_k$	Connections in level-k bottleneck
ν_k	Max-min rate allocation for bottleneck level-k ($\nu_k \leq \nu_{k+1}$)
r_i	Ideal max-min fair share allocation for connection i
g_i	Bottleneck grouping level for connection i

Chapter 1

Introduction

In recent years there has been a tremendous growth in the use of computer networks, fueled by rapid advances in the area of fibre optics, VLSI technology, and software development methodologies. As networks get more advanced, service providers are being asked to provide a wider range of services to their end users. *Asynchronous transfer mode* (ATM) networks are very indicative of this trend, being designed to carry a broad spectrum of services, from data to voice/video and multimedia. ATM networks operate on an underlying connection-oriented philosophy and use high-speed packet switching techniques to achieve the data transport function. Traffic streams are packetized into small, fixed-size blocks called *cells*, 53 bytes in length, and this allows for a fine degree of bandwidth granularity. Each network switch has multiple input/output links and routes incoming cells onto output links. A switch's resources are a combination of its output link bandwidth, buffer space, and processing elements [12]. A major advantage of ATM technology is the ability to achieve dynamic bandwidth allocation, which allows for the support of many

diverse traffic types. However, such traffic types usually have widely varying source characteristics, i.e., bandwidth and burstiness. In addition their *quality of service* (QoS) requirements differ significantly, such as cell loss rates, cell delay, and delay jitter. Therefore, proper coexistence among all end user types requires traffic classes to be defined and appropriate transport mechanisms implemented for ATM networks. Both the ITU-T and ATM Forum organizations have formulated several classes to cover the desired ranges of traffic [1],[2]. Providing services support to meet the requirements of all traffic classes is a major challenge to ATM network designers. These classes and their proposed control mechanisms are now briefly reviewed.

1.1 Overview of Traffic Classes

In a broad sense, ATM network traffic can be grouped into two classes [11]: real-time traffic and non-real-time traffic. Some general details on each class are now presented.

1.1.1 Real-time Traffic Classes

Many real-time applications have known bandwidth profiles and cell delay requirements. Examples include voice/video connections, teleconferencing calls, multimedia, virtual reality applications, etc. For such applications, cells cannot be delayed excessively, otherwise they will be of limited or no use at the receiver (i.e., too much distortion in the "playback"). Hence the inter-cell timings must be adequately controlled. In addition, depending upon the nature of the overlying application, the cell

loss requirements may vary for real-time connections. For example, voice streams can tolerate higher loss probabilities without significant degradation in the output quality (i.e., 10^{-4} to 10^{-5}). However, such guarantees may be insufficient for other real-time applications such as compressed video streams [11].

Along these lines, the *constant-bit rate* (CBR) class has been designed for supporting tightly constrained real-time connections (i.e., cell delays and delay variations, [1, p. 6]. For such connections, a "fixed" amount of bandwidth, the *peak cell rate* (PCR), must be reserved by the network. This is the inverse of the minimum inter-cell spacing allowed in the source cell stream. In addition, cell delay tolerances and cell delay variation parameters are also given to bound deviations in cell delivery. Furthermore, another more flexible service type, the *variable bit rate* class, is also defined. VBR is primarily intended for more bursty connections, whose rates are expected to vary with time. In particular, two versions of the VBR traffic class are declared, real-time and non-real-time [1],[2]. For real-time VBR connections, additional parameters need to be specified (beyond those for the CBR service) to best capture the source behaviour. Specifically, an additional rate attribute, called the *sustainable cell rate* (SCR), is defined. This value represents a nominal rate at which connection traffic is to be sustained by the network, and is normally between the mean and peak source rates. Furthermore, a *burst tolerance* (BT) value is also defined to limit the maximum block of back-to-back cells which can be sent at the PCR. Further details on the specifications can be found in [1],[2].

All of the above real-time connection parameters are negotiated with the network at call setup time. This is done via the *call admission control* (CAC) proce-

dure. This function uses a source's requested attributes to determine if an acceptable connection route can be established to the destination. If the request cannot be satisfied, the network can either turn down the connection request or initiate a re-negotiation phase to determine an acceptable set of specifications. For example, the CAC procedure would determine the exact SCR values for VBR connections. If a connection request is granted by the CAC procedure, the agreed upon parameters are subsequently monitored by the policing function (to be discussed subsequently).

1.1.2 Non-Real-time Traffic Classes

In the literature, the term *data* traffic has become synonymous with non-real-time traffic. Examples include LAN (*local area network*) traffic, email messages, file transfer, client-server interactions, facsimilie transmissions, etc. Studies have found that data traffic is typically highly bursty over many time scales [9],[17] and hence good models do not exist for properly characterizing its behaviour. This makes resource reservation-type approaches inflexible. Furthermore, traffic generated by data applications is usually more delay-tolerant than real-time traffic [12] (although the delay sensitivity can vary depending upon the application). As a result, the use of feedback flow control schemes is feasible.

To address the above issue, the ATM Forum has specifically formulated the *available bit rate* (ABR) and *unspecified bit rate* (UBR) traffic classes. The UBR class places very modest demands on network switch equipment and requires the higher layer applications to have adequate response mechanisms to react to cell delays or losses (see [1],[2]). Meanwhile, the ABR traffic class is considerably more

complex, and places more responsibility on ATM networks to ensure reliable data delivery. Hence the overall goal of ABR service is to maximize the goodput of data connections without violating the QoS requirements of real-time traffic. This requires oversubscribing leftover link bandwidth from the CBR and VBR classes: "ABR...should have the goal of providing rapid access to unused network bandwidth...whenever [such] network bandwidth is available," [1, p. 10]. With possible high peak-to-mean ratios (i.e., burstiness) of differing traffic types, significant economies of scale can be realized in operation. In light of migration issues for the large legacy LAN base, the ABR class is crucial to the evolutionary deployment of ATM technology. "The ABR service specification is perhaps the most critical and sensitive specification ever tackled by the ATM Forum..." [16].

Due to the reactive nature, the service offered to an ABR connection can change subsequent to connection establishment [1], depending upon network conditions. Network switches send feedback to ABR sources in order to control their transmission behaviour, and this is done using special ATM *resource management* (RM) control cells. Thus the ABR class uses *closed-loop* control and is the only such class defined in the ATM specifications [1],[2]. The source end-system periodically emits *inband* RM cells, interspersed within the data cell stream, and these are then looped back via the destination. Network switches, meanwhile, insert feedback information in the RM cells to control the sources. The precise method or algorithm used to derive the feedback is termed the *ABR control scheme*. Sources which conform to network feedback can expect to have low cell loss ratios [1],[2]. Conversely, cells from non-compliant connections can be tagged or dropped by the network.

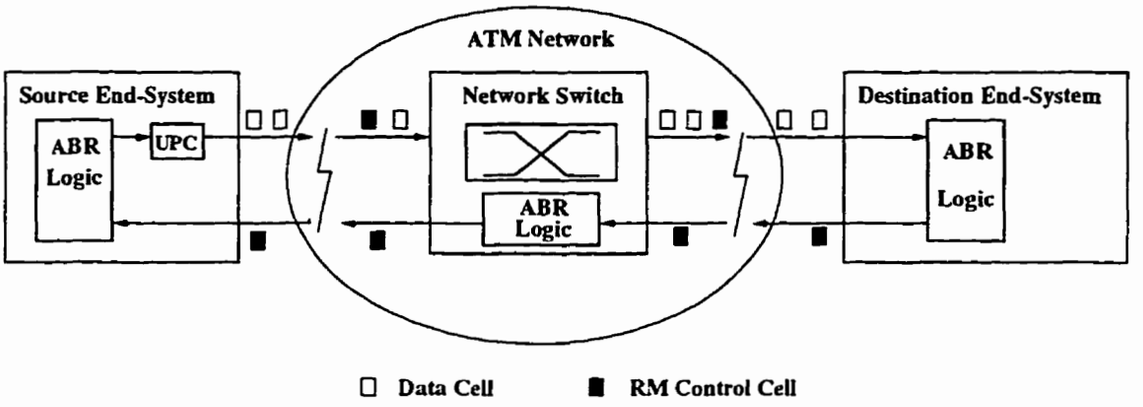


Figure 1.1: Generic overview of ABR feedback control

Although precise traffic requirements are hard to determine at setup time, a data connection is still required to negotiate certain ABR traffic descriptors. These include the *peak cell rate* (PCR), *minimum cell rate* (MCR), and *initial cell rate* (ICR) traffic descriptors. The PCR is defined as the inverse of the minimum spacing between two successive data cells (analogous to the CBR/VBR definition). However, the MCR can be interpretted in several ways. In a strict sense, it can specify the maximum *sustainable* spacing between two successive cells which has to be maintained by the network. More generally, though, it can also represent the minimum *mean* throughput given to the source over a larger time interval. Users are allowed to specify zero MCR values, in which case the service provided is closer to "best-effort." However, among other benefits, non-zero MCR allows for ABR traffic prioritization, quasi-real time applications support, and more flexibility for evolving applications [12]. The ICR is merely the maximum start-up rate, and is usually set to a small fraction of the PCR. Although the actual CAC algorithm is left to vendor discretion, the ATM Forum specifications do require all zero MCR

connection requests to be accepted [1].

Some QoS requirements are also specified for ABR service. The single most important one is the *cell loss ratio* (CLR). Many data applications require error-free transmission, and otherwise retransmit corrupted or lost packets. The relatively large data packet sizes (with respect to the 53-byte ATM cell) can give rise to the *fragmentation* problem, severely degrading throughputs [18]. Hence, stringent cell loss requirements are required to limit the effect of this problem [12]. Delay requirements, meanwhile, are not explicitly given for ABR connections. Nevertheless, it is advisable to avoid large transit delays for data cells in order to prevent higher-layer application timeouts (and retransmissions). Refer to [1],[2] for more details on the ABR specification.

1.2 Overview of Traffic Control Schemes

In ATM networks, traffic has to be carefully controlled in order to prevent network *congestion* and maintain the QoS guarantees of all connections (classes). Congestion, also referred to as overload, occurs when the demand for a resource exceeds its capacity [8]:

$$\sum \text{Demand} > \text{Available Resources.} \tag{1.1}$$

For example, if the incoming cell rate to a link's buffer exceeds it transmission capacity, rate (and possibly buffer) congestion occurs. Congestion can result in undesirable effects, such as excessive cell delays or loss. Typical causes of congestion

include temporary increases in resource utilization, improper network dimensioning, misbehaving users, or even network failures. Since many data applications retransmit delayed/lost packets, if not properly controlled, congestion can worsen to the point where *all* capacity is being used up for retransmission: congestion collapse ([5, p. 390], [8]). Congestion management is the term used in a broad sense to refer to traffic control in networks.

At the highest level, congestion control is done on a long-term basis and modifies the overall network configuration. This has been referred to as *network engineering* [5] or *network resources management* (NRM) [10]. Factors affecting this level include projected user demands, network size, network topology, monetary constraints, etc. Congestion control at this level can involve link sizing and placement, switch size selection, virtual path and traffic class assignments, etc. At a more intermediate level, congestion management is concerned with congestion periods of the order of connection duration time scales. Here, the network performs CAC procedures to appropriately accept or reject connection requests. (Note that for connection-oriented services, routing is also performed to establish a connection, but this is *not* considered as a control scheme.) However, for periods smaller than connection times, separate congestion control schemes have to be devised. Specifically, for real-time traffic, congestion effects can occur at intervals of the order of cell times and thus *open-loop* control schemes are applicable. However, with data traffic, closed-loop schemes are more applicable for congestion periods greater than roundtrip delays. An overview is now presented.

1.2.1 Open-Loop Congestion Control

The overall goal of cell-level open-loop control schemes is to maintain the QoS requirements of connections during periods of congestion, namely throughput, delay, and cell-loss. A variety of mechanisms have been proposed to address these issues. One such example is *usage parameter control* (UPC), which checks to see if connection traffic streams are conforming to their agreed upon traffic descriptors. This is a *preventive* approach [11] and tries to inhibit non-compliant connections from adversely affecting other users. Although conceivably many parameters can be monitored (controlled), a rate-related parameter is usually chosen. For CBR streams this would be the PCR value and for VBR streams the SCR value. Specifically, the *leaky-bucket* algorithm [4] has been proposed to police the given rate and also limit burst sizes, as shown in Fig. 1.2 (also referred to as the *generic rate control algorithm*, GRCA [1],[2]). In this mechanism, data cells can be transmitted if there are available tokens in the token buffer, which are replenished at a fixed rate. If the token pool is empty, then the cells are tagged or dropped by the policer (see [4] for details). Furthermore, the negotiated connection parameters (such as the PCR, SCR, cell delay variation) can be used to determine the leaky-bucket values, namely the token rate, and data/token buffer sizes.

Although UPC can limit ingress traffic, multiplexing effects within the network can still increase stream burstiness. As a result, additional mechanisms must also be derived to maintain QoS guarantees. Along these lines, service *scheduling* schemes have been proposed to ensure that the throughput and delay guarantees are maintained for individual cell streams (see [83]-[85] for complete surveys). These

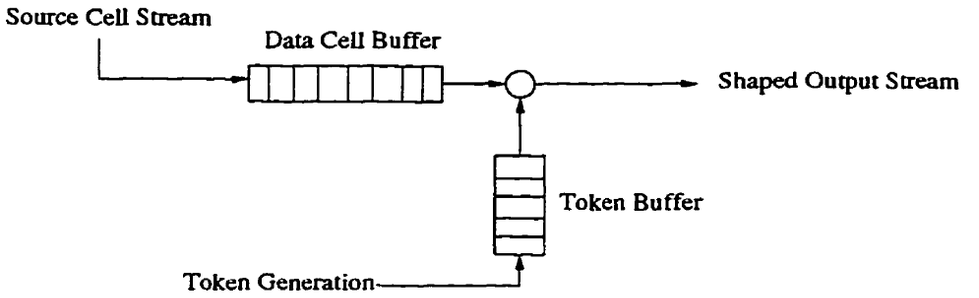


Figure 1.2: Leaky bucket controller

approaches, however, can involve significant switch complexities. Specifically, the cells for each connection (or class of connections) are separately buffered in a FIFO manner, and arbitration logic is used to determine the transmission order during busy (backlog) periods.

1.2.2 Closed-Loop Congestion Control

Closed-loop schemes have typically been termed as *reactive* congestion control mechanisms [11]. In these approaches, the network sends some form of feedback to sources, who in turn modulate their transmission behaviour. Feedback flow control schemes have existed long before the advent of ATM technology and ABR services, with the two major types being credit-based and rate-based approaches.

Credit-based control is a link-by-link mechanism where downstream switches specify how many data cells (packets) a source can transmit. Usually, the switches send permits (i.e., "credits") contingent to their available buffer levels. An upstream source is only allowed to send data if it has available credits. Credit-based schemes can achieve tight queue control and high link utilizations [7],[21]. More importantly, by preventing credit transfers in excess of free buffer space, cell losses

can be eliminated. However, there is significant implementation complexity with credit-based schemes: credit counters and per-connection queues are required at all switch ingress links. Round-robin service scheduling is also required to achieve fairness between the connections. Depending upon the credit management between competing connections, two types of schemes can be realized, static credit and adaptive credit (refer to [21],[22] for more details).

In rate-based schemes, meanwhile, switches specify the transmission rates for sources. This requires source end-systems to be able to vary their transmission rates and results in increased implementation complexities. However, the network switch complexity with rate-based schemes is typically lower than that with credit-based schemes [12], since per-connection queueing is no longer necessary (see [19] for an excellent discussion). Nevertheless, per-connection accounting is usually required to achieve fairness amongst connections [22], and rate-based schemes cannot guarantee zero loss performance [7],[21]. Rate-based approaches, however, attempt to minimize the chances of buffer overflow by providing large storage buffers. The philosophy here is more to minimize implementation complexity and not be overly concerned with maintaining the tight cell loss requirements.

An extensive study was conducted by the ATM Forum's Traffic Management committee to evaluate flow control strategies for the ABR service. In the fall of 1994, a rate-based approach was overwhelmingly approved. The major reason for this was the large implementation complexities of credit-based schemes, which limited their scalability. Since most current network transport technology is based on *first-in-first-out* (FIFO) queueing, the added per-connection requirements posed

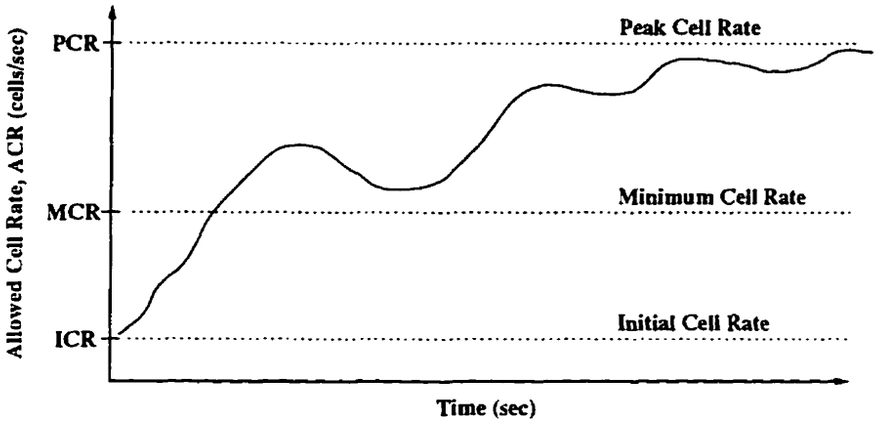


Figure 1.3: General behaviour of ABR source rate, ACR

a significant obstacle to a gradual migratory approach to ABR services support. Among other issues, it is also difficult to guarantee (minimum) cell rates to connections using credit-based schemes [3, Chapter 6]. Also, most credit-based schemes use estimates of the roundtrip delays to derive buffer allocations, etc., (see [22]). Obtaining reliable, accurate measurements for these quantities may require using robust estimation techniques, and this poses increased costs in practical networks with varying queue sizes and service rates.

The adoption of a rate-based ABR standard defines the notion of a varying connection *allowed cell rate* (ACR), which is the current maximum rate at which

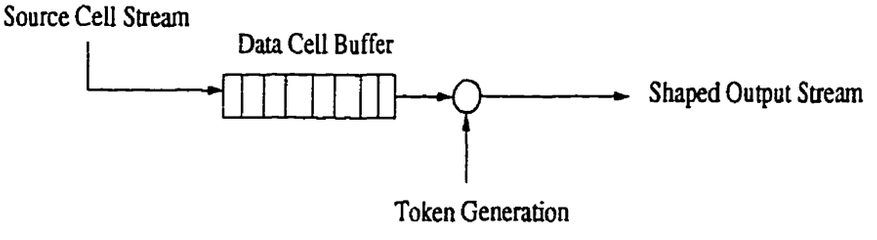


Figure 1.4: Spacer controller

cells can be transmitted. This value is measured at the output of the *user network interface* (UNI) at the source end-system, and is thus the actual input rate into the network, be it the local *customer premises exchange* (CPE) or ATM network. At start-up, the source transmits at its ICR and subsequently continually adjusts its rate as specified by the received feedback. The generic ACR behaviour, depicted in Fig. 1.3, shows that the cell rate should stay within the MCR-PCR window after ramp-up. Note that users have no obligation to send at *at least* their ACR (or MCR) values. UPC of ABR sources can be achieved by using a *simple* leaky bucket controller [6, p. 255-257] as shown in Fig. 1.4. This is also known as a *spacer controller*, and basically ensures that the inverse of the minimum spacing between any two consecutive cells does not exceed the current ACR. In essence, this is similar to a leaky bucket controller without a token buffer and an *adaptive* token rate.

1.3 Issues with ABR Services Support

Given the proposed rate-based framework for ABR services, there are many issues and challenges involved in implementing acceptable flow control schemes. Some are not new to this field and have been treated in the past, whereas others are more specific to the ABR specification. Some of these concerns are now detailed.

1.3.1 Bandwidth-Delay Problem

Due to increasing network link rates, the ratio between propagation delays and cell transmission times has become very large. As transmission rates increase, more

and more cells can be pumped into the network in a given amount of time. This poses a problem for feedback schemes due to the inherent "fixed" propagation delay between the source and destination endpoints. Further increases in transmission link speeds will only exacerbate this problem. For example, in the time it takes a "decrease" message from a network switch to reach the source, a large amount of data may already have been inserted onto the line. In addition, control information arriving at the source is more dated as the bandwidth delay increases. So even if the user adjusts correctly to network feedback, network resources may still have to be reserved to enable low cell loss operation [2, p. 33]. The delay problem cannot be removed in that it arises from the limit of the speed of light. However what can be done is to design schemes which minimize the chances of buffer overflow with delayed feedback information. The solution requires accurate tracking of the resource usage levels at switches and advance prediction of possible congestion. The effects of long propagation delays can also be lessened by segmenting a longer end-to-end control loop into smaller ones, making feedback tighter (as specified in the basic ABR specification [1],[7],[12]).

1.3.2 Network Dynamics

In order to properly allocate resources to competing ABR connections, network switches need to know how much available resources they have beforehand. This is complicated by the fact that connection usages (both ABR and non-ABR) can fluctuate considerably. For example, as VBR users come and go and change their activity levels (i.e., bursts), the leftover bandwidth can vary from the order of mil-

liseconds to seconds [23]. As a result ABR control schemes can be sensitive to the resource usage of other service classes and proper mechanisms must be designed to properly isolate the operation of the various traffic classes. In addition to being robust, resource estimation algorithms must be independent of the underlying switch architectures (i.e., such as scheduling strategies, buffer partitioning schemes, etc). For example, some advanced scheduling disciplines can provide information on per-class bandwidth and buffer usages, but such functionality cannot be assumed. Errors in improper tracking can be detrimental to the performance. If available capacity is over-estimated, then queue lengths can increase and cause cell losses. Similarly, under-estimating capacity can result in reduced throughputs and prolonged overloads. Efficiency (and utilization) indicate how well the resource tracking (and rate allocation) algorithms are "packing" the available bandwidth for ABR users.

1.3.3 Fairness

A heavily debated topic in the ATM Forum has been that of *fair* resource allocation between competing ABR connections. Since cell delays are not a major concern and connection *rates* have to be assigned, ABR-related discussions have focused exclusively on *bandwidth* fairness between users. If the available bandwidth at a given link is enough to support the requests of all traversing connections then the fairness issue does not arise. Bandwidth allocation amongst users is only required when the aggregate demand is greater than the available capacity. At an intuitive level a bandwidth allocation strategy is fair if it does not offer a different treatment

to connections, either based on the time orders in which they make their requests, or on the the particular location of their source and destination endpoints [12]. The ATM Forum has quantified this thinking into a generic fairness index but does not indicate how to actually *compute* the actual allocations (see [1] for details).

However, the ATM Forum's traffic management committee has accepted the *max-min* bandwidth fairness criterion [1],[3] as a desirable objective. Basically, this implies that a bottlenecked connection at a given link will receive a rate allocation which is *at least* as large as that of any other connection also bottlenecked at the same link (see Section D.2.1 for a brief review of max-min fairness). Max-min fairness applies unambiguously only if all connections have zero MCR requirements, and the more general case of non-zero MCR guarantees is left to vendor discretion. Various authors have proposed several extensions for this case [13],[25],[26]. A major challenge in ABR flow control is ensuring that the desired fairness criterion is implemented to an acceptable degree.

1.3.4 Scalability

ATM networks are expected to cover a wide range of configurations varying in the number of switches, transmission link speeds, geographic distances, user populations, behaviour, etc. Therefore it is essential that an ABR scheme not be limited to a particular type of network (e.g., LAN, ring, bus) or source behaviour (persistent, bursty). Many schemes work well in certain environments but poorly in others. For example, end-to-end flow control or aggressive source policies are typically more effective in smaller size networks such as LAN's as opposed to WAN's

(*wide area network*), mainly due to shorter feedback times. However, in WAN networks, these approaches can result in huge queue buildups, possible cell losses, and reduced bandwidth utilizations. Algorithm scalability is therefore a very important issue, especially in light of the the current proliferation of data services in networks.

1.3.5 Robustness, Complexity, Overhead

An underlying concern in evaluating an ABR algorithm is its practicality: is the scheme feasible from a technological and economic point of view? With commercial link rates running in the hundreds of Mbps, and quickly approaching the Gbps mark, highly complex control logic may not be possible at all [23]. The complexity of a scheme can be assessed from its control parameters and required logic. Ideally, the number of control parameters should be minimized so as to improve portability and robustness. Reduced parameter counts improve adaptability and such transparent "plug-and-play" functionality is crucial for a smooth, gradual transition into ABR service usage [16]. Robustness is another feature closely related to complexity. At the very least, any scheme should be relatively immune to minor errors in control parameter settings or loss/delay of control information. If slight variations can result in drastic changes in performance, the scheme may be unacceptable. It is likely that minor parameter errors and RM cell loss will occur, and their consequences should be minimized.

Overhead and cost are two other related issues. Overhead usually refers to the network resources taken up for ABR operation, i.e., resources used beyond the transport of data cells. This includes control cell bandwidth and buffers, param-

eter storage requirements, and processing overheads. These factors must be kept low in order to have more resources left for transporting user data. Cost is more an economic measure of implementing the control functionality. This comprises fabrication costs for the hardware logic, storage memories, and related software development expenses. Inevitably, since ABR is significantly more complex than the data service of current networks, it is expected that the complexities at both the end-systems and network switches will increase.

1.4 Motivation and Goals

The major goal of the research is to design a comprehensive strategy for supporting the ABR service class in ATM networks. The approaches to be considered are strictly at the cell-level. As such, call-level functions such as CAC and routing schemes are not addressed. A robust, distributed feedback control scheme, one which achieves good queue control and fair bandwidth allocation amongst connections, is required. The proposed scheme should fit into the specified *explicit-rate* (ER) guidelines set by the ATM Forum [1],[12], and must have significant advantages over existing rate-based proposals in the ATM Forum (i.e., related to performance issues such as queue control, bandwidth fairness, scalability, etc.). Furthermore, the algorithm should be of acceptable implementation complexity and be able to function well in a wide range of network conditions. Due to the extremely complex nature of the problem, discrete-event simulation techniques should be used to evaluate the design and compare its performance against some other schemes.

The integration of ABR flow control algorithms with transport mechanisms for

real-time CBR and VBR traffic is also a relatively important area. Currently, no detailed algorithms have been presented to properly isolate ABR and VBR traffic streams at the *lowest* cell-level. Specifically, service scheduling schemes which meet the requirements of all traffic classes are necessary. The schedulers should maintain real-time connection delays and also provide protection for guaranteed portions of the data throughputs. In light of the delay-insensitivity of data traffic, also, there should not be a significant increase in switch complexity over current schemes.

1.5 Thesis Organization

The main focus of the thesis is on ABR service management. However, the provision of efficient and robust ABR flow control strategies also entails the coexistence of ABR and non-ABR services. The main text is devoted to ABR flow control while details are relegated to the appendices. The thesis is organized in the following manner.

In Chapter 2 the motivations for the research are presented. A brief overview of the major existing ABR proposals is given and shortcomings revealed. Furthermore, simulation results are also presented to identify additional deficiencies. From this, the major goals of the research are formulated.

In Chapter 3 the proposed ABR flow control scheme is presented. The algorithm is based upon maintaining bandwidth fairness between competing connections, as defined in its rate allocation phase. The capacity tracking and congestion detection features are also introduced to handle the effects of network dynamics.

In Chapter 4, simulation results are presented for a broad range of network

scenarios and comparisons made with other existing proposals. Initial tests verify performance in single link networks. Subsequently, results are also presented with more elaborate multiple-link scenarios. Concluding remarks and recommendations for future research are given in Chapter 5.

Appendix A presents detailed descriptions of the major existing ATM Forum rate-based schemes. This complements the brief overviews in Section 2.1.

Appendix B considers the issue of scheduling ABR traffic along with real-time traffic inside an ATM network switch. An overall hierarchical strategy is proposed to meet the requirements of all service types. Detailed scheduling algorithms are also presented to describe the cell reception and transmission functions. Simulation results are presented and comparisons made with other scheduling methods. In addition, Appendix C looks at some issues related to the proposed scheduler.

Appendix D discusses the convergence properties of the basic rate allocation algorithm in a static environment, and includes a brief overview of max-min bandwidth fairness. The treatment considers both single and multiple link cases, and addresses some practical concerns.

Appendix E also derives a relatively stringent condition for non-oscillatory operation of the scheme in static environments. Although some simplifying assumptions are fairly tight, results confirm that the condition holds in a wide range of networks.

Worst case transient queueing bounds are derived for a simplified version of the scheme in Appendices F (single link case) and G (multiple link case). The bounds are based on various pessimistic assumptions, and their overall accuracy is judged via a simulation study.

Finally, a brief overview of the discrete event simulator developed for evaluating the flow control and scheduling strategies is presented in Appendix H. Here, descriptions of the software classes and their main functions are given.

Chapter 2

Motivations and Goals

Many proposals for ABR flow control have been presented recently. The schemes vary over a wide range, differing in their complexities and performance behaviour. In order to properly rank the relative strengths and weaknesses of the various algorithms a complete survey is necessary. Certain algorithms have well-documented benefits and shortcomings, but other performance aspects need further investigation. This mandates detailed testing of certain schemes to gauge performance criteria such as efficiency, bandwidth fairness, robustness, etc. Only then can the critical areas be clearly identified and adequate solutions proposed and evaluated. This is the focus of this chapter. Some results of the initial work, from which the main motivations for the research are derived, are now presented.

2.1 Overview of ATM Forum Schemes

Early ABR proposals were simple binary, non-selective schemes [14]. In such schemes, source rates are varied up or down by pre-determined amounts, and per-connection behaviour is not taken into account. Subsequently, as shortcomings were found, more advanced selective, *explicit-rate* (ER) schemes were proposed. In these schemes, network switches compute explicit rate updates for each connection and attempt to achieve fair bandwidth distribution. The major proposals are briefly reviewed, and more detailed descriptions can also be found in Appendix A. Furthermore, the references [12],[14] contain excellent descriptions of the ABR service class and surveys of all the major proposals.

2.1.1 Binary Feedback Schemes

Backward explicit congestion notification (BECN) is an early scheme proposed to improve responsiveness for multiple-hop connections [9],[27]. Using queue thresholds, network switches send backward decrease RM cells to all sources whenever the settings are exceeded. The sources, meanwhile, use timeout intervals to increase their rates. If no backward RM cells are received within the timeout interval, fixed rate increments are permitted. However, such *negative* feedback is usually risky in case of RM cell losses or delays [12]. *Forward explicit congestion notification* (FECN) is an end-to-end binary scheme which uses EFCI-bit (*explicit forward congestion indication*) marking. Congested switches mark the EFCI bits in data cells and these are filtered by the destination end-systems to send backward RM increase cells to the source (positive feedback). The congestion indicators are queue

thresholds, and an additive-increase/multiplicative decrease policy is used for rate adjustment (proven fair for single-hop, zero-delay case [66]). The *proportional rate control algorithm* (PRCA) is an improvement over the FECN scheme, using the same additive-increase/multiplicative decrease mechanism. Here, sources exponentially decay their rates between cell transmissions and perform additive increments upon receipt of backward RM increase cells. An improved hysterisis-type queue threshold mechanism is also added to reduce oscillations. All of the above three schemes require minimal switch complexity.

The biggest drawbacks with non-selective schemes are their bandwidth fairness and queue oscillation problems. In congested states, connections traversing more switches have a higher probability of having their rates being reduced than those traversing fewer switches [7]. This arises due to the non-selective nature of the feedback indication. In addition, upon congestion abatement, longer-hop connections also have increased ramp-up times. Both of these effects degrade the throughput performance of longer-hop connections (more pronounced in end-to-end schemes, FECN and PRCA). This is termed the *beat-down* problem [12] and can only be adequately resolved by employing selective feedback techniques (i.e., per-connection accounting). Another problem with binary schemes is the relatively large oscillations in the queue length (and source rates). This gives reduced efficiency and can even result in increased cell loss if the congestion thresholds are improperly set or queue sizes are too small. Furthermore, the responsiveness of binary schemes is poor. In extreme congestion, the network cannot quickly quench source rates, and instead must rely on the exponential rate decay or timeout mechanisms. Source

ramp-up times are governed by queue thresholds and pre-set rate increments, giving sluggish responses to increases in spare capacity levels. The schemes also exhibit significant sensitivity to parameter settings (i.e., thresholds, rate decay factors, rate increments), reducing their autoconfigurability and hindering scalability to general networks. The authors in [74] also expose further weaknesses with the pure FECN and BECN algorithms.

2.1.2 Explicit Rate Feedback Schemes

Explicit rate schemes have been proposed to resolve the bandwidth fairness and oscillation problems of the binary schemes. Selective feedback is used to improve fairness by limiting rate decreases (increases) to connections receiving relatively more (less) throughput than others. The definition of relative throughput is an open issue, although most algorithms attempt to achieve the *max-min* bandwidth fairness criterion [3]. This leads to a rich flavour of explicit rate algorithms, most of which can generally be classified as one of two types, approximate fair and exact fair schemes (as proposed in [14]).

The *enhanced proportional rate control algorithm* (EPRCA) [28] is an approximate fair rate scheme, developed from the *intelligent congestion control* (ICC) proposal [29]. The scheme keeps a running average of the source rates to approximate the fair share, and uses two queue thresholds (regular, heavy). When the queue length exceeds the regular threshold, then only connections whose rates are above the tracked mean rate are throttled (i.e., selective feedback). If the queue length exceeds the higher threshold, all connection rates are reduced. To enforce

convergence to fair rates, several multiplier factors are used. EPRCA has relatively low implementation complexity and does not require per-connection accounting.

However, a problem with EPRCA is its large number of control parameters, and sensitivity to their settings. Improper tuning of these parameters can result in large oscillations and reduced bandwidth fairness [14]. This reduces the scheme's autoconfigurability and scalability to different networks [13]. In addition, if extreme congestion persists, the fairness of the scheme is compromised significantly [32]. Although various enhancements have been proposed to overcome some of these limitations, outstanding issues still remain (see [14]). For example, *adaptive proportional rate control* (APRC) is a modification of EPRCA which attempts to improve responsiveness and fairness by using faster rate overload congestion indicators [32]. However, since no explicit queue information is used, the queue length can still exceed the high threshold and degrade fairness (see [53]). Very recently, there have been additional approximate fair rate proposals, most notably the *max-min rate control* (MMRCA) [47] and *dynamic max rate control* (DMRCA) schemes [48]. These are not evaluated here, although in [14] is it stated that the algorithms require small rate increments to function effectively.

Unlike the approximate fair rate algorithms, a set of *exact* fair rate computation schemes have also been proposed. These algorithms require network switches to know the explicit *available capacity* levels at the links, and attempt to adjust sources directly to their fair rates. An early such scheme was presented in [33], and relied on synchronous network-wide operation to achieve max-min bandwidth fairness. *Distributed explicit rate allocation* algorithms [38] are also another set of exact fair

share algorithms. An example of one such scheme is the MIT-scheme proposed by Charny [35], which is similar in spirit to the earlier, more generalized scheme in [34]. Sources intermittently send control cells containing their current rate and *desired* rate values. Switches monitor the rates and compute a fair share value using the max-min principle [12]. Connections with rates below the fair share are allowed to ramp-up to their desired rates, whereas others are reduced to the fair share. Although response times are fast, there can be significant transient queue buildups with this scheme. Furthermore, the scheme has a considerable increase in switch complexity, most importantly an $O(n)$ computation per control cell (n being the number of connections). To reduce the implementation complexity, recently, an *efficient rate allocation algorithm* (ERAA) [42] has been proposed. The scheme has $O(1)$ computational complexity and can also achieve max-min fairness in a general network setting.

Jain et al. also present two similar end-to-end exact fair rate computation schemes, the *Ohio State University* (OSU) scheme [43] and *explicit rate indication for congestion avoidance* (ERICA) [44]. The switches measure the aggregate traffic input rate over a fixed interval and compare it against a target rate to compute a load factor. The target rate is chosen based upon a desired link utilization, usually around 90%. Switches also monitor connection activity and compute fair share allocations. The goal is to keep the load factor close to unity and operate in the congestion avoidance region [8]. ERICA provides very fast control (as low as one roundtrip delay) and relatively small queue oscillations. Nevertheless, the scheme has been shown to be unfair in various environments, and for certain settings can

yield divergent queueing behaviour [54],[55]. Very recently, there have been many additional modifications and extensions proposed for the ERICA algorithm in order to address some of these shortcomings, namely the ERICA+ scheme [46].

Overall, most of the proposed exact fair rate computation algorithms do not completely address the requirements of the ABR traffic class. For example, it is unclear how to extend the above algorithms to properly handle MCR guarantees. Simply removing the aggregate guaranteed MCR bandwidth component from the given available capacity levels can give reduced utilizations if connections are idle. Also, provisions are lacking for functionality in dynamic environments, where capacity levels fluctuate due to non-ABR traffic. All the above exact schemes assume a fixed, pre-determined link capacity to be distributed amongst the ABR flows. More importantly, the schemes do not have any provisions for handling transient queue buildups, which occur during either source ramp-ups or under the influence of interfering non-ABR traffic (with the exception of the recent upgrades to the ERICA scheme [46]). As such, they must be operated in the congestion avoidance region [8], erring on the side of conservative bandwidth utilization (i.e., available capacities or target rates set to below full bandwidth utilization). Nevertheless, even this cautious approach cannot prevent divergent queueing behaviour under certain conditions [54].

Note that some authors have also investigated control-theoretic approaches to ABR flow control [58]-[65]. Here, the main goal is to match the (tracked) underlying capacity to the aggregate input rate or maintain the operating queue levels about acceptable threshold values. The formulations assume that the number of active

sources and their propagation delays are known and attempt to tune the transient and steady-state responses using a variety of controllers. Examples include proportional controllers [60], proportional-plus-delay controllers [59], H_2 techniques [62], generalized predictive control [63], Smith predictors [61], etc. Typically, the system is solved by computing appropriate weighting coefficients to ensure proper pole placement (and responses).

Although control-theoretic schemes can draw from rich analytical backgrounds, it is felt that they are difficult to realize in practice. Since significant approximations are made in casting the ABR flow control problem into a linear, control-theoretic form, deviations may cause unexpected behaviour. For example, rapidly changing connection sets may require excessive re-computation of weighting coefficients or result in compromised performance otherwise. In addition, it is assumed that the fixed (propagation) delay component dominates the delays [62],[63], and thus inter-RM cell and queueing delays are not considered. Nevertheless, these delays are dominant in LAN networks and can be significant even in WAN networks. As a result, the above formulations are approximations and may have reduced accuracy. The implementation overheads can also be significant with such schemes. These include computing filter coefficients, storing previous rate/queue values, etc. Most such schemes also assume that the available capacity levels along with the number of "active" sources are known. Determining these values poses additional complexity and inaccuracies in their estimates need to be further investigated. Finally, it is not clear if such schemes can achieve max-min bandwidth fairness without explicit additional provisions (as proposed in [64]).

2.2 Simulation Study

Some performance evaluation results from a preliminary simulation study are now presented for existing ABR control schemes. In particular, sample binary and explicit rate control algorithms are chosen, namely PRCA and EPRCA. The choice of PRCA as the representative binary scheme is justifiable in that it is one of the most advanced such schemes. However, for the explicit rate case, there is a wider range of schemes to choose from. Nevertheless, the EPRCA algorithm is chosen since it is one of the most well-known explicit rate proposals. More importantly, EPRCA addresses all the pertinent issues in ABR services support and is the basic scheme in the ATM Forum's proposal [1]. In particular, this algorithm can function in heterogeneous traffic environments and has provisions for MCR rate guarantees. With most other explicit rate algorithms, these provisions are lacking or unclear, and significant extra extensions are required. A similar choice is also made by the authors in [47],[48]. Although the literature contains several variants of EPRCA, the algorithm specified in [28] is used.

Simulation results are presented for a broad range of network scenarios to investigate the performance of the two algorithms. Most of the control parameter settings are chosen from the pertinent values specified in the ATM Forum [28], and are summarized in Table 2.1 (see Appendix A for descriptions). However, some additional parameter sensitivity results are also presented. Note that for the EPRCA scheme, the high queue threshold, DQT, is set to a slightly higher value of 1000 cells to improve fairness with larger connection delays (as suggested in [48]). All results are gathered over two independent runs of duration one second each.

Table 2.1: PRCA/EPRCA parameter settings

Parameter	Setting
Link rates	150 Mbps
N_{rm}	32
AIR	100 kbps
ICR	200 kbps
MDF	256
QT	100 cells
DQT^*	1000 cells
AV^*	0.25
MRF^*	0.25
ERF^*	0.9375

* *EPRCA parameter only*

2.2.1 Performance Metrics

Considering the various goals of the ABR service schemes, proper metrics must be defined to gauge the relative performance of schemes. These include metrics pertaining to queue control, bandwidth fairness (and efficiency), cell-loss performance, etc. Since one of the major goals of ABR is low cell loss rates, the maximum queue buildups are an important metric. As such, these values can be measured over both transient and steady-state intervals to estimate the required operational queue sizes. However, typically, maximum queue buildups are more appropriate in fixed environments, where link capacities and connection demands are constant (or connections are persistent). In dynamic environments with fluctuating capacity levels, a given observed maximum queue size cannot indicate the *absolute* maximum queue buildups possible. Here, it is more appropriate to use metrics which capture the random nature of the queue size. As a result, the mean and standard deviation

of the queue length are used to gauge queue control. The cell loss behaviour can also be inferred, in a more intuitive manner, from these metrics. Typically, it desirable for schemes to have acceptable mean queue levels and small variances (i.e., tight, responsive control). Because feedback schemes are most effective for time scales greater than the roundtrip delays, it is necessary to measure the random quantities for time periods well beyond this range.

To gauge the bandwidth fairness properties of a scheme, several measures can be used. A very important metric is the mean throughput (goodput) of a connection, measured over a sufficiently long time interval ($\gg$ roundtrip delay):

$$\text{Throughput} = \frac{\text{Cells received at destination}}{\text{Measurement interval}}. \tag{2.1}$$

This value can be compared to the ideal (mean) max-min allocation for the connection and the throughput deviation determined. Throughputs should be measured in steady-state, i.e., after the initial ramp-up stage.

Furthermore, to quantify the *aggregate* fairness over a group of connections, several measures are also possible [7],[56]. One such measure is an index similar to that in [7], but with added modifications for handling MCR requirements. Consider N ABR connections. Let r_i denote the ideal throughput for connection i (as determined by the respective fairness criterion), and $\overline{r_i}$ the actual measured *mean* value. The *modified fairness index* is defined as:

$$f = \frac{(\sum_{i=1}^{N} x_i)^2}{N \sum_{i=1}^{N} x_i^2}, \tag{2.2}$$

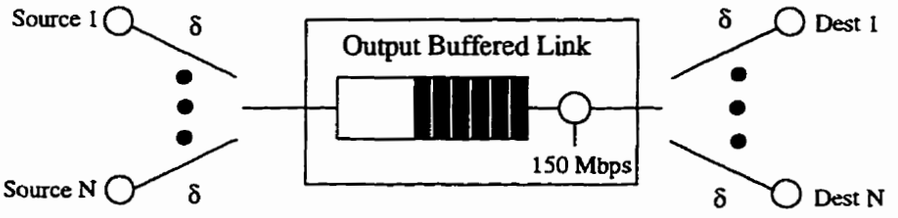


Figure 2.1: Single link delay scenario, all connections using same output link

where x_i is the normalized mean elastic throughput (i.e., beyond MCR):

$$x_i = \frac{\overline{r_i} - MCR^i}{r_i - MCR^i}, \tag{2.3}$$

and MCR reservation is assumed (i.e., $(\overline{r_i} - MCR^i) \geq 0$). Eq. (2.2) reduces to the measure in [7] for the zero-MCR case. Yet another possibility is the *average throughput deviation*, defined as:

$$\text{Throughput deviation} = \frac{1}{N}\sum_{i=1}^{N} |1 - x_i|. \tag{2.4}$$

By removing the guaranteed MCR portions first, Eq. (2.3) ensures that fairness is measured over only the contended bandwidth (i.e., elastic bandwidth, Section 3.3.1). This prevents MCR guarantees from skewing fairness values when connections with larger/smaller MCR values are not being treated fairly. Note that aggregate fairness measures cannot clearly indicate the fairness for a particular group of connections.

2.2.2 Single Link Scenarios

Initially, simpler single link networks are tested to get a relative ranking of the two algorithms. Such test cases can demonstrate the basic characteristics of a scheme without possible multiple link (distortion) effects. Subsequently, more advanced multiple link scenarios are tested to gauge the schemes in a more rigorous manner.

Scalability Scenario

Consider the scalability scenario in Fig. 2.1, with N connections, all sharing the same output link at a switch. Each connection is persistent and has an end-system-to-switch delay of δ sec. (i.e., roundtrip delay of 4δ sec.). The performance of the two schemes is tested for an increasing number of connections and differing link delays, ranging from LAN (microseconds) to WAN (milliseconds). Fig. 2.2 shows the effect of the link delays on the maximum queue size for WAN-type delays, for 10 and 20 connections, respectively. The results confirm the improved queueing performance of the explicit rate EPRCA scheme. The maximum queue size requirement is acceptable and tends to scale linearly with the delay, δ. However, the corresponding throughput performance of both schemes, depicted in Fig. 2.3, shows that the throughput efficiency (i.e., link utilization) declines with increasing propagation delays. Longer delays cause the magnitude and period of the oscillations to increase significantly, resulting in longer queue idle times (i.e., link starvation). For example, for 12 ms roundtrip delays ($\delta = 3$ ms), the link efficiency decreases to about 80% with EPRCA. To show this more clearly, sample queue behaviour for $N = 10$ connections and $\delta = 1$ ms is shown in Fig. 2.4. It can be seen that there is

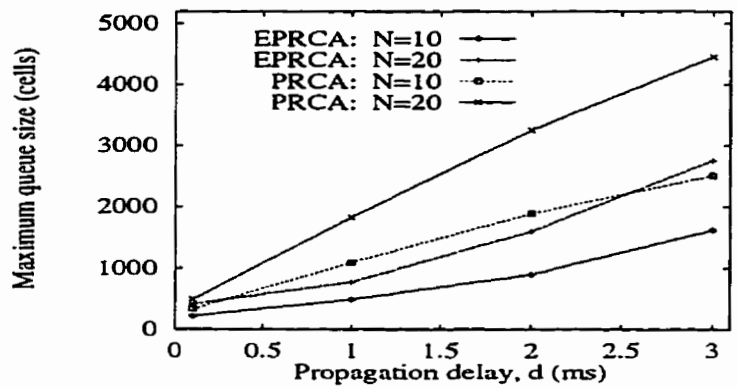


Figure 2.2: Maximum queue sizes in Fig. 2.1

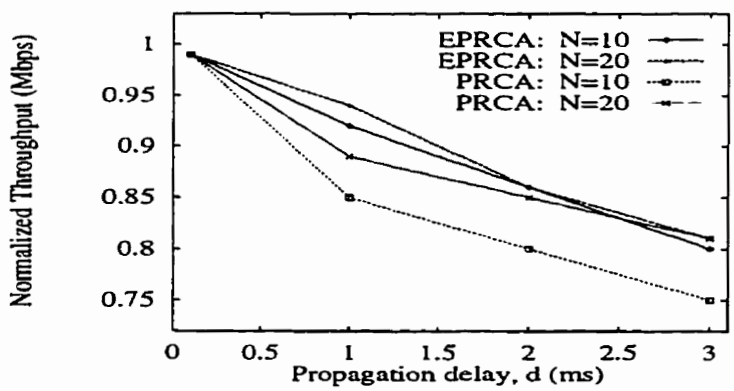


Figure 2.3: Average normalized throughputs in Fig. 2.1

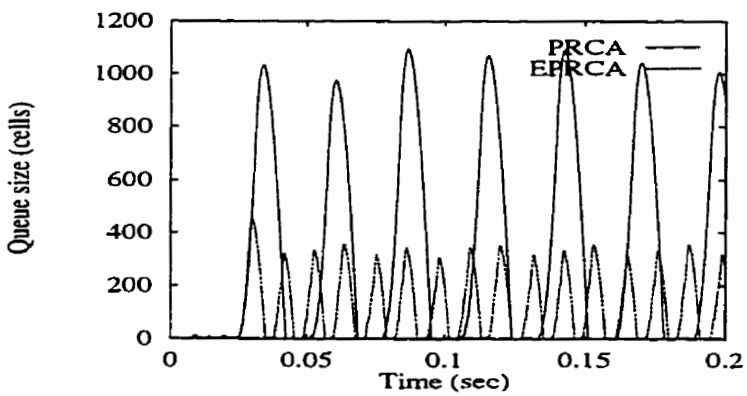


Figure 2.4: Queue behaviour in Fig. 2.1: $N = 10$, δ=1 ms

link starvation with both schemes, although, the EPRCA scheme excercises much tighter queue control about the QT threshold. This indicates that the simpler binary scheme is less scalable to larger delay networks, giving increased oscillations and reduced link utilizations. Additional simulations also show that increasing the QT threshold can improve the throughput performance of the EPRCA scheme. However, this also gives correspondingly larger operating queue sizes.

Bandwidth-Delay Scenario

A more challenging single node scenario, designed to test bandwidth-delay performance, is shown in Fig. 2.5 and Table 2.2. There are three ABR connection groups, each consisting of ten source-destination pairs, competing for bandwidth at link (link propagation delay negligible). To represent switches supporting a broad range of users, the propagation delays for the connection groups are chosen to span several orders of magnitude (from milliseconds to tens of microseconds). Furthermore, ten interfering bursty VBR connections are also added. First, the scenario is tested for the ABR-only case (i.e., VBR connections idle), and the throughput results are given in Table 2.3. It can be seen that the EPRCA scheme provides good performance, with all connection groups getting within 10% of their ideal

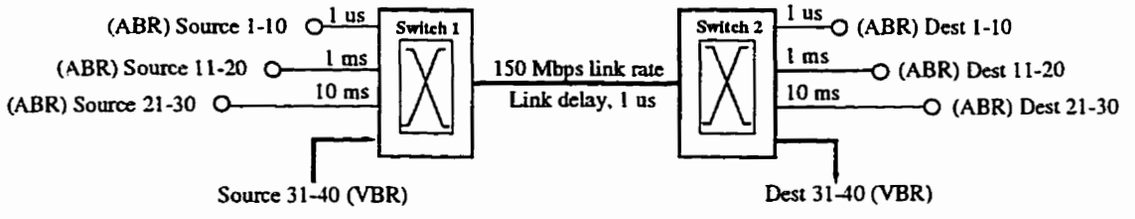


Figure 2.5: Single link bandwidth-delay scenario

Table 2.2: Single link scenario in Fig. 2.5, idle VBR

Connection	Behaviour	MCR/PCR (Mbps)	Throughput (Mbps)
ABR 1-10	Persistent	0/150	5
ABR 11-20	Persistent	0/150	5
ABR 21-30	Persistent	0/150	5
VBR 31-40	Idle	-/-	0

throughput, 5 Mbps (Table 2.2). However, the performance of the PRCA scheme is significantly worse. There is considerable bandwidth unfairness, with the shorter-delay connections getting increased throughputs at the expense of the longer-delay connections (i.e., 6.95 Mbps vs. 2.80 Mbps, Table 2.3). This is mainly due to the fact that PRCA does not track connection rates and hence provides much coarser control. Corresponding sample queue behaviour at the link buffer is also shown in Fig. 2.6, and indicates that both schemes yield roughly the same magnitude of queue oscillations.

Parameter Sensitivity Scenario

Various parameter sensitivity tests are done with the scenario in Fig. 2.5, and sample results for the AIR rate increment parameter are presented. Specifically, sensitivity to the AIR parameter is measured by varying it for the longer delay connections, connections 21-30 (AIR = 100 kbps for all other connections, Table 2.1). This is a difficult scenario since the longer delay connections can now ramp up faster, but still respond more slowly to feedback control. Measured throughputs for each connection group with AIR values ranging from 50 kbps to 200 kbps are

Table 2.3: Bandwidth-delay performance, Fig. 2.5 (throughputs in Mbps)

PRCA				
Connections 1-10 Throughput	Connections 11-20 Throughput	Connections 21-30 Throughput	Bandwidth Utilization	Throughput Deviation
6.95	4.79	2.80	96%	29.1%
EPRCA				
Connections 1-10 Throughput	Connections 11-20 Throughput	Connections 21-30 Throughput	Bandwidth Utilization	Throughput Deviation
4.55	4.64	5.28	96%	7.3%

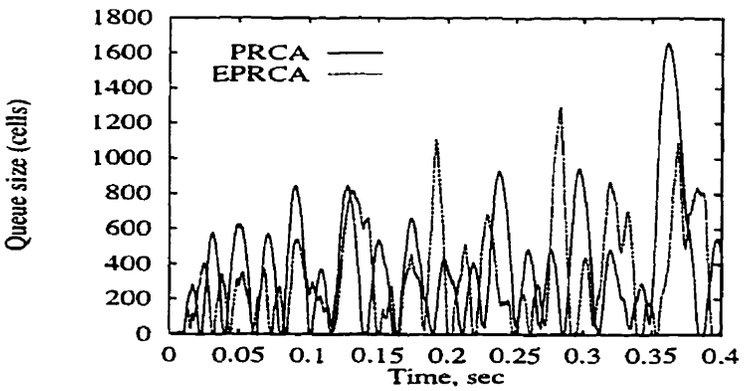


Figure 2.6: Queue behaviour in Fig. 2.5, idle VBR

plotted in Fig. 2.7, PRCA, and Fig. 2.8, EPRCA, (straight lines representing ideal throughputs of 5 Mbps). Clearly, EPRCA does not maintain fair bandwidth distribution when the AIR discrepancy becomes too large. The longer delay connections get markedly better throughputs at the expense of the shorter delay connections. With PRCA, the trend is also similar, but since the scheme already discriminates against longer delay connections (i.e., Table 2.3), increasing their AIR values tends to *improve* the overall bandwidth fairness (i.e., compensatory effect). Nevertheless, both schemes show large sensitivity to the AIR parameter in this scenario. The corresponding maximum queue sizes are shown in Fig. 2.9, and also show an interesting trend. Contrary to some expectations, the PRCA scheme gives relatively smaller buildups with increasing AIR values. This occurs since the longer delay connections get larger throughputs with EPRCA, and their larger bandwidth-delay proximities cause operating queue sizes to increase. Additional tests also reveal large sensitivity to the MDF (rate decay) parameter, specific to the EPRCA and PRCA schemes. The EPRCA scheme, however, is relatively robust with regards to the N_{rm} inter-RM cell parameter.

Bursty VBR Scenario

Since ABR traffic must coexist with real-time traffic inside the network, the scenario in Fig. 2.5 is augmented with bursty VBR sources. Each VBR source is defined as on-off, and the multiple superposition allows for more diversity in the *aggregate* VBR behaviour. The on and off periods are exponentially distributed in an identical manner, and each source generates cells with fixed inter-cell spacings of roughly 35 μs in the on period, i.e., giving an approximate mean usage of 6

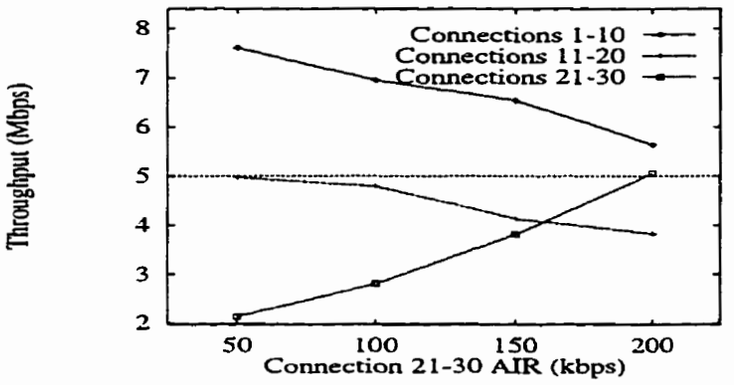


Figure 2.7: PRCA throughput performance in Fig. 2.5, idle VBR

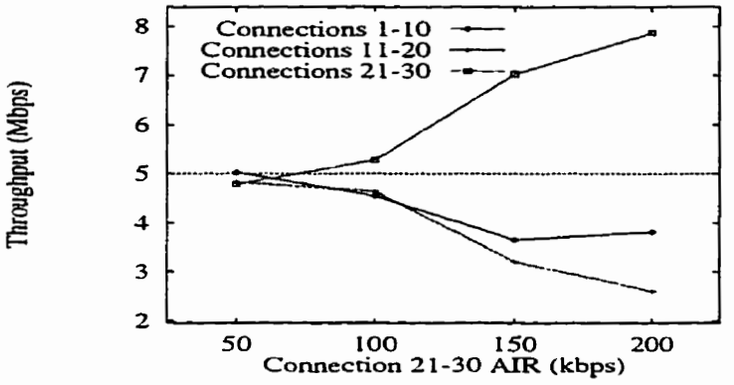


Figure 2.8: EPRCA throughput performance in Fig. 2.5, idle VBR

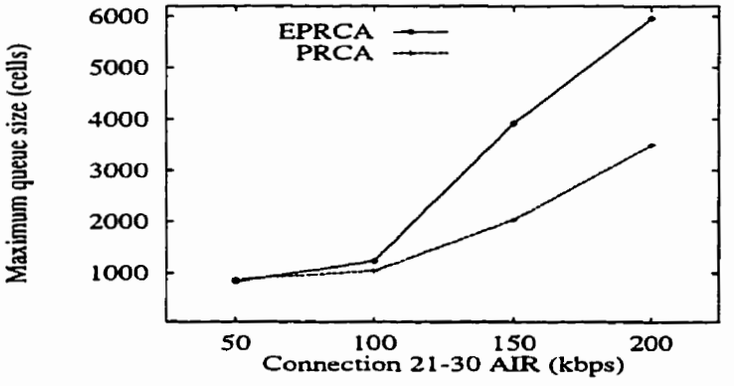


Figure 2.9: Maximum queue sizes at link in Fig. 2.5, idle VBR

Table 2.4: Single link scenario in Fig. 2.5, bursty VBR

Connection	Behaviour	MCR/PCR (Mbps)	Throughput (Mbps)
ABR 1-10	Persistent	0/150	3
ABR 11-20	Persistent	0/150	3
ABR 21-30	Persistent	0/150	3
VBR 31-40	Bursty	-/-	6

Mbps, see Table 2.4. This leaves an average of 90 Mbps to be shared amongst the ABR connections, resulting in ideal *mean* allocations of 3 Mbps each. To test the sensitivity of the flow control schemes to the exact nature of VBR traffic, the mean of the on/off period distribution is varied from 1 ms to 20 ms. For this scenario, results are only presented for the EPRCA scheme since it exhibits considerably superior performance than the PRCA scheme.

Fig. 2.10 plots the measured throughputs for all connection groups in Fig. 2.5, and indicates considerable sensitivity to the burst durations of the background VBR traffic. For increasing on periods durations, since the bursts are more sustained, EPRCA cannot adequately throttle the longer delay sources. Hence, there is reduced bandwidth fairness for longer bursts (i.e., connections 1-10 getting almost 50% above their ideal allocation for 20 ms periods). The corresponding mean and standard deviation of the queue sizes at the link are shown in Fig. 2.11. The plots indicate worsening performance with increasing on and off periods (even though the link utilization, is very high, over 95%). Since the longer delay connections 21-30 are getting larger allocations, their increased delay proximity causes operating queue levels to increase. Note that additional tests with different queue threshold

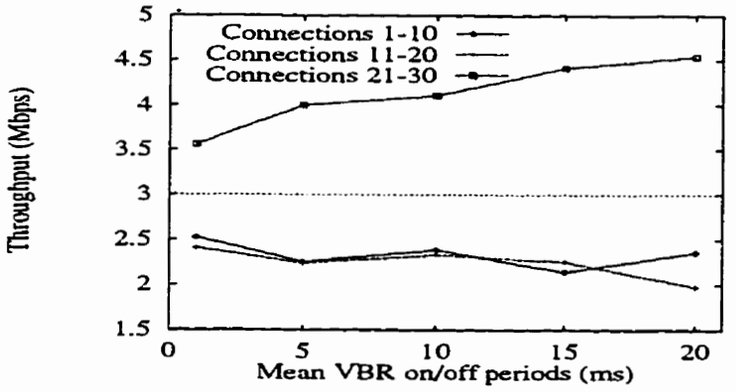


Figure 2.10: EPRCA throughputs in Fig. 2.5, bursty VBR

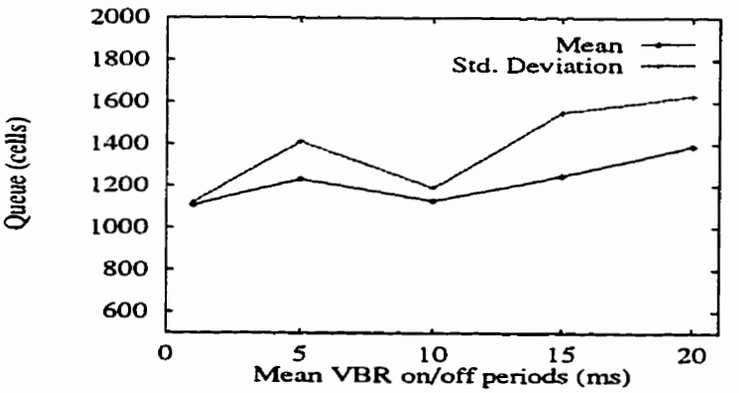


Figure 2.11: EPRCA queue at link in Fig. 2.5, bursty VBR

settings (i.e., $QT = 400, 600$, $DQT = 1500, 2000$) also yield poor fairness and even larger operating queue sizes.

Non-Persistent Source Scenario

In many environments, data sources may not have infinite amounts of data to send at all times (i.e., not persistent over an infinite time horizon). Usually, such sources exhibit highly bursty behaviour [18], and hence can be considered as persistent over shorter time periods. The effect of such *non-persistent* source behaviour on ABR schemes needs investigation. Along these lines, the scenario in Fig. 2.5 is modified by making the shorter delay connections, connections 1-10, non-persistent. Specifically, these sources are bursty, with exponentially distributed on and off periods. In the on period, the cells are generated at a the current ACR (persistent behaviour). Consider the *mean duty cycle* for a source, defined as:

$$\text{Mean duty cycle} = \frac{\text{Mean on period}}{\text{Mean on period} + \text{Mean off period}}. \tag{2.5}$$

Hence, the duty cycle can be varied by keeping the *sum* of the means of the on

Table 2.5: Non-persistent ABR source scenario in Fig. 2.5

Connections 1-10 Duty Cycle	Connections 1-10 Throughput (Mbps)	Connections 11-20 Throughput (Mbps)	Connections 21-30 Throughput (Mbps)
0.2	1.0	7.0	7.0
0.4	2.0	6.5	6.5
0.6	3.0	6.0	6.0
0.8	4.0	5.5	5.5
1.0	5.0	5.0	5.0

and off periods to a constant value, and only varying the mean of the on period. Specifically, the sum is kept fixed at 10 ms and the mean of the on period is varied from 2 ms to 10 ms, using increments of 2 ms. The resulting fair shares are given in Table 2.5 for all connection groups (i.e., *mean* max-min allocations).

For the above scenario, again, only results with the EPRCA scheme are discussed, since PRCA yields very poor throughput and queueing performance. Fig. 2.12 shows the EPRCA throughput performance as a function of the duty cycle of the non-persistent short-delay connections. The straight reference lines represent the ideal max-min throughputs for full link utilization, namely the values from Table 2.5. Two trends are evident here. First of all, the rate-decay mechanism in EPRCA is degrading the throughput of the bursty connections, and the additive ramp-ups exacerbate this problem. Secondly, the longer delay connections are getting a disproportionately large amount of throughput. This indicates that EPRCA is not properly allocating rates according to the *mean* connection usages. Even worse, there is an abrupt increase in the operating queue sizes as the non-persistent loading drops, see Fig. 2.13. Both the mean and standard deviation of the queue length increase significantly for smaller duty cycles (i.e., more than tripling between 100% and 20% duty cycle, Fig. 2.13). The corresponding maximum observed queue buildups are also plotted in Fig. 2.13. Albeit "snapshots" of random quantities, these values also show drastic increases with reduced duty cycles. These trends are due to the fact that the longer delay connections are generally sending at higher rates and have increased control latency.

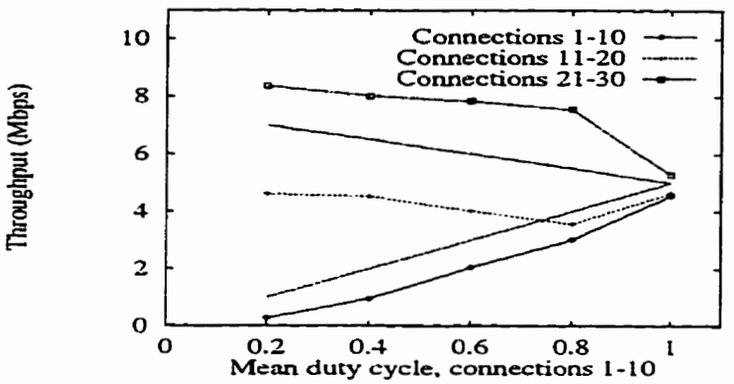


Figure 2.12: EPRCA throughputs in Fig. 2.5, non-persistent connections 1-10

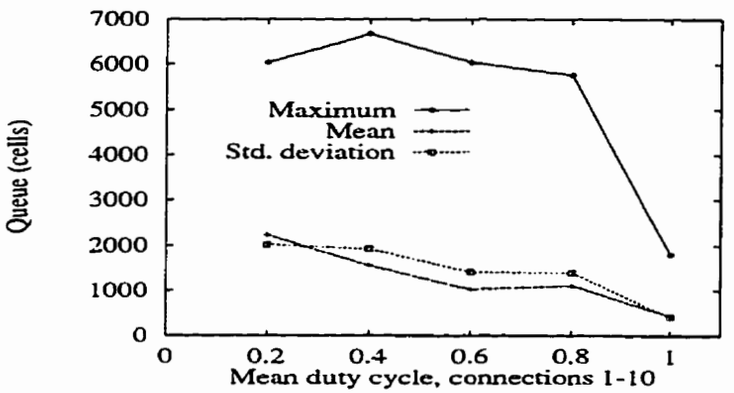


Figure 2.13: EPRCA queue at link in Fig. 2.5, non-persistent connections 1-10

2.2.3 Multiple Link Scenarios

Now some simulation results with larger multiple link (node) networks are presented to observe end-to-end performance issues. One of the major goals is to investigate the bandwidth distribution properties with *differing* MCR guarantees amongst connections. The tests are conducted with large-scale, dynamic networks in order to fully stress the performance of the schemes.

Bandwidth-Fairness Scenario

Consider the sample multiple link network shown in Fig. 2.14. This is similar to the *parking lot* scenarios [7],[12] proposed to test bandwidth fairness. There are four ABR connection groups, each consisting of ten connections, with each group having different bandwidth-delay proximities. The bigger connection sets are chosen to test scalability to large-scale networks. In addition, a set of ten interfering VBR connections is also added to represent heterogeneous environments. The scenario is first tested for the ABR-only case, i.e., all VBR connections idle. Using the *MCR-plus-equal-share criteria* [1],[13], the ideal rate allocations can be computed (see Table 2.6), using a procedure similar to that in [3]. For example, the most

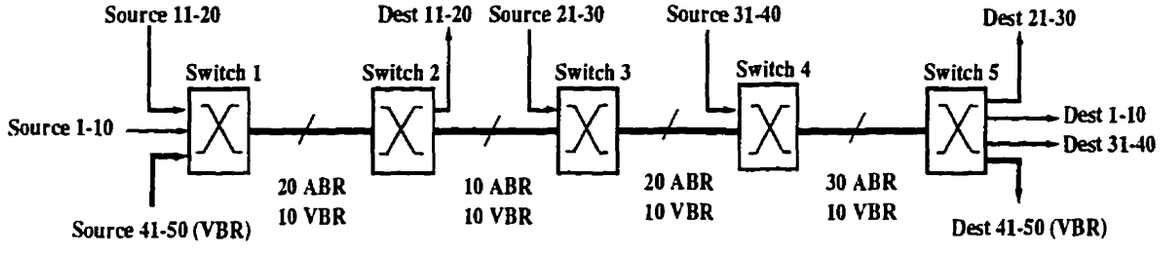


Figure 2.14: Multiple link network with VBR connections

Table 2.6: Multiple link network in Fig. 2.14, idle VBR

Connections	Type	Behaviour	MCR/PCR (Mbps)	Throughput (Mbps)
1-10	ABR	Persistent	0/150	4
11-20	ABR	Persistent	0/150	11
21-30	ABR	Persistent	1/150	5
31-40	ABR	Persistent	2/150	6
41-50	VBR	Idle	-/-	0

bottlenecked link in Fig. 2.14 is link 4-5, supporting 30 connections (connections 1-10 and 21-40). The combined MCR requirement of the bottlenecked connections at this link is 30 Mbps, yielding 120 Mbps to be shared amongst the flows. Hence each bottlenecked connection should get its MCR guarantee plus an additional 4 Mbps from the contendable bandwidth. Connections 11-20 are bottlenecked elsewhere, and hence can receive larger allocations, namely 11 Mbps.

To stress the schemes, the above scenario is run for large, WAN-like link delays, ranging from 1 ms to 5 ms. This is a very challenging, considering that the roundtrip delays for the longer-hop connections 1-10 increase from roughly 8 ms to 40 ms. The steady-state measured throughputs for PRCA and EPRCA are shown in Fig. 2.15 and Fig. 2.16, respectively (straight lines representing the MCR-plus-equal share allocations, Table 2.6). From the plots, the *beat-down* problem [12],[14] with the PRCA scheme is clearly evident. The longer-hop connections are getting under 50% of their ideal allocation (i.e., 1.70 Mbps mean vs. 4 Mbps, Table 2.6), whereas the shorter-hop connections are getting much larger allocations. In addition, there is significant link underutilization with PRCA at the most bottlenecked link for

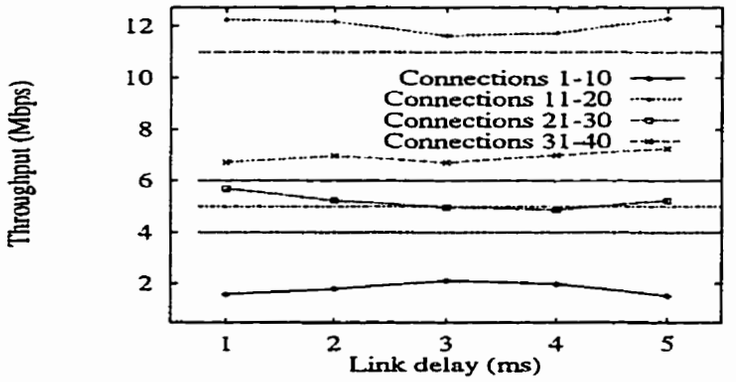


Figure 2.15: PRCA throughputs in Fig. 2.14, idle VBR

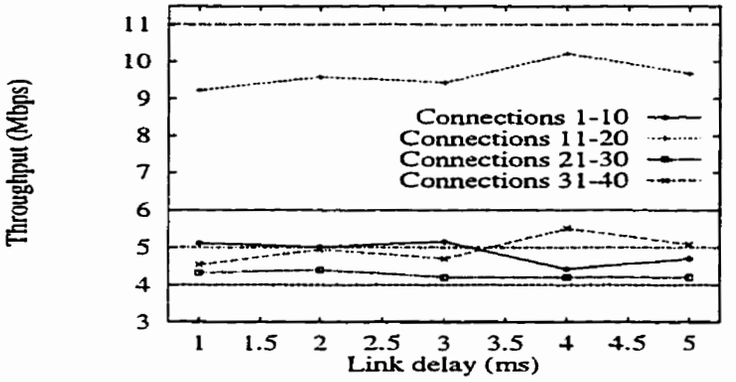


Figure 2.16: EPRCA throughputs in Fig. 2.14, idle VBR

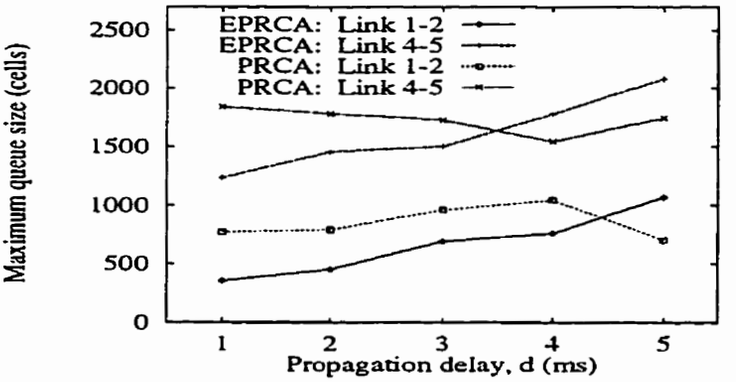


Figure 2.17: Maximum queue sizes in Fig. 2.14, idle VBR

larger delays (i.e., below 80% for 5 ms delays).

Although the fairness improves with the EPRCA scheme, in general the scheme tends to "cluster" the connection rates at the most bottlenecked link about the same average value, i.e., roughly 4.5 Mbps at link 4-5 (see Fig. 2.16). In particular, EPRCA tends to favour the longer-hop connections, giving connections 1-10 on average 20% extra bandwidth beyond their fair allocations. This discrepancy arises since EPRCA estimates the mean cell rate over all connections at a link (i.e., $MACR$, Eq. (A.4), Appendix A). In the case that the estimated rate is close to the ideal fair share, the scheme performs well. However, when certain connections are bottlenecked elsewhere, the mean is generally not an accurate estimate (i.e., underestimate) of the true fair share, and unfairness can result [48]. This is the case at link 1-2, where the longer-hop non-bottlenecked connections, connections 1-10, lower the estimate and degrade the performance of connections 11-20. This problem is even more pronounced in some of the test cases presented in [48] (see reference, Fig. 11) and has also been noted by other authors [53]. In general, the bandwidth distribution properties of EPRCA beyond the MCR guarantees are not very predictable. Fig. 2.17 plots the maximum queuer sizes at the two most bottlenecked links, links 1-2 and link 4-5. In general, the queue sizes are larger at the more bottlenecked link and tend to increase linearly with propagation delays.

Bursty VBR Scenario

The multiple link scenario is now tested for the EPRCA scheme with interfering, bursty VBR traffic. Each VBR connection is on-off with exponentially distributed on and off periods of mean 1 ms each. In the on period, cells are generated with

Table 2.7: Multiple link network in Fig. 2.14, bursty VBR

Connections	Type	Behaviour	MCR/PCR (Mbps)	Mean Throughput (Mbps)
1-10	ABR	Persistent	0/150	3
11-20	ABR	Persistent	0/150	9
21-30	ABR	Persistent	1/150	4
31-40	ABR	Persistent	2/150	5
41-50	VBR	Bursty	-/-	3

exponential inter-arrival times of roughly 71 μs (i.e., mean usage of about 3 Mbps per connection). Subtracting the mean VBR usages from the link capacities and using the fairness criterion, the ideal *mean* ABR throughputs can be computed, and are given in Table 2.7. The mean throughputs observed for each ABR connection group are shown in Fig. 2.18. Again, the fairness of the scheme is still poor, with the longer-hop connections getting better treatment (similar to idle VBR case, Fig. 2.16). Fig. 2.19 plots the queue length metrics (at the most bottlenecked link) and indicates that the standard deviation more than doubles for larger propagation delays. This indicates wider oscillations and overall higher maximum possible queue buildups.

Link Delay Scenario

Consider the scenario detailed in Fig. 2.20 and Table 2.8, where *all* propagation delays (link and end-system-to-switch) are set to a variable quantity, δ sec. This is a challenging test case, consisting of long-hop connections competing for bandwidth with single-hop connections and interfering VBR traffic. The mean rate of the VBR connections is set to 6 Mbps, giving an average *residual* ABR bandwidth of

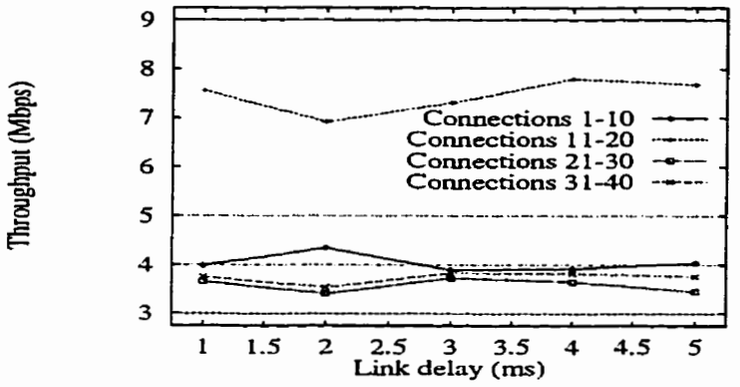


Figure 2.18: EPRCA throughputs in Fig. 2.14, bursty VBR

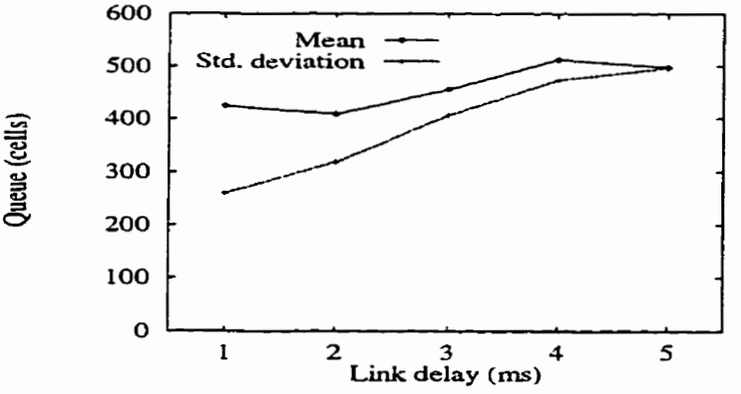
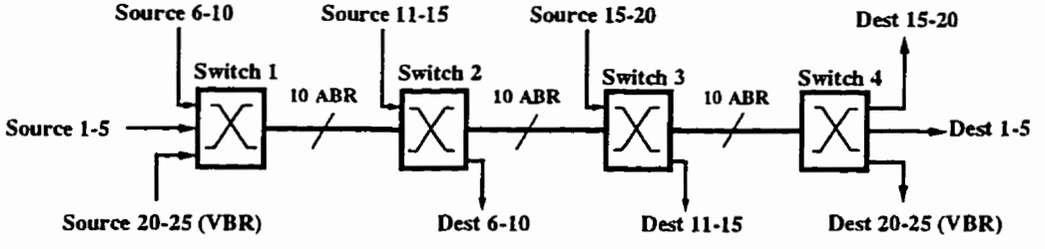


Figure 2.19: EPRCA queues, link 4-5, in Fig. 2.14, bursty VBR



All link and source/destination end-system-to-switch delays, δ

Figure 2.20: Multiple link network with VBR connections (Table 2.8)

Table 2.8: Multiple-node scenario in Fig. 2.20

Connections	Type	Roundtrip delay (ms)	MCR/PCR (Mbps)	Ideal mean throughput (Mbps)
1-5	ABR/Persistent	10δ	0/150	12
6-10	ABR/Persistent	4δ	0/150	12
11-15	ABR/Persistent	4δ	0/150	12
16-20	ABR/Persistent	4δ	0/150	12
20-25	VBR/Bursty	-	-/-	6

120 Mbps on all links. Specifically, each VBR source is bursty with mean on and off periods of 1 ms (exponential), and has a mean Poisson rate of roughly 12 Mbps in the on period (i.e., exponentially distributed inter-arrival times with mean 35.3 μs). Hence, using the max-min criterion [3] each ABR connection should receive a mean throughput of 12 Mbps (Table 2.8).

Since the EPRCA scheme yields considerably better fairness, only its performance results are presented. The average throughput results for each group of connections in Fig. 2.20 are shown in Fig. 2.21. The horizontal reference line represents the ideal throughput of 12 Mbps. The results show that for LAN/MAN delays (i.e., $\delta \leq 100$ μs), the bandwidth fairness is very good, with all groups receiving within 10% of the ideal 12 Mbps. However, for larger WAN delays, there is a re-emergence of the beat-down fairness problem, as noted by the severe degradation of the throughputs for connections 1-5. For example, at 5 ms delays, the longer hop connections average only 4.33 Mbps with the EPRCA scheme, well below 50% of the ideal fair share, Fig. 2.21. Reduced EPRCA fairness is also noted in [48] for various other scenarios.

The mean and standard deviation of the link 1-2 queue length are plotted in Fig. 2.22 (other link queues exhibit slightly smaller operating sizes). The EPRCA scheme is quite sensitive to the delay δ in this particular scenario, as noted by the sizable increase in both quantities. The queue length oscillations increase significantly, and although not shown, the bandwidth efficiency also declines (i.e., only 63% for $\delta = 5$ ms). Additional EPRCA runs are also performed with larger queue threshold settings at WAN delays (i.e., $QT = 500$, $DQT = 2500$). Here, there is a slight improvement in bandwidth fairness (i.e., connections 1-5 now get slightly over 6 Mbps for $\delta = 5$ ms), but the mean queue sizes roughly double. These results indicate that EPRCA can have poor scalability (with regards to bandwidth fairness, queue control, utilization) for large-scale WAN networks.

2.3 Summary and Motivations

Overall, the findings of the simulation study indicate that the binary PRCA scheme has many shortcomings. The oscillations are relatively large and bandwidth fairness is very poor. Furthermore, scalability to larger connection sets, increased link delays, and multiple node networks is very low with PRCA. The results also show that the EPRCA scheme, albeit considerably better than PRCA in most cases, is still deficient in many important areas. EPRCA suffers from relatively large oscillations and exhibits significant parameter sensitivity. Variations in the background real-time VBR traffic or ABR source behaviour can also cause considerable fluctuation (degradation) in the throughput fairness and queueing performance of the scheme. Furthermore, the approximate nature of EPRCA cannot provide very

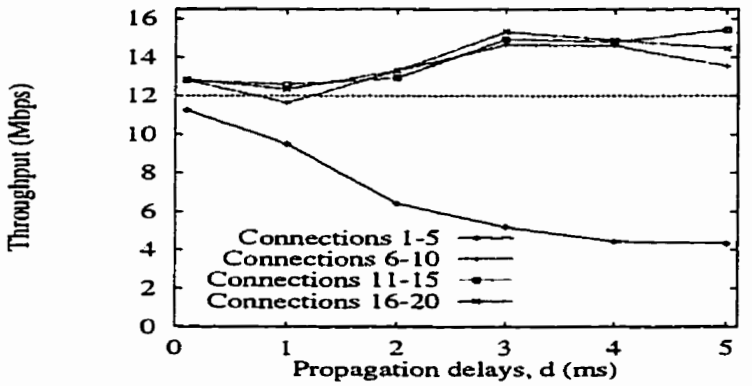


Figure 2.21: EPRCA throughputs in Fig. 2.20

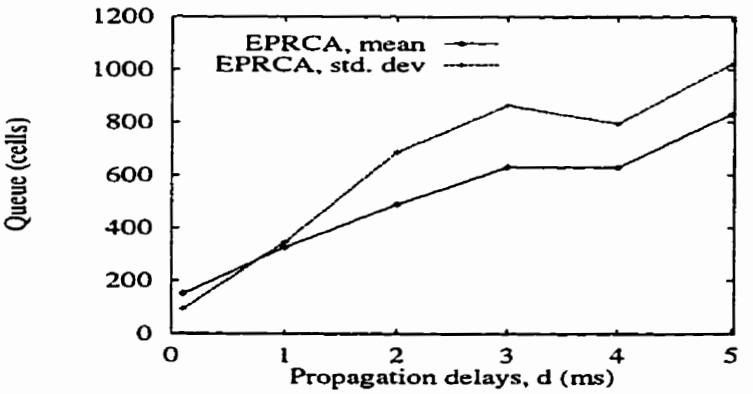


Figure 2.22: EPRCA queue at link 1-2 in Fig. 2.20

precise bandwidth distribution in larger, multiple node networks.

These findings underscore the need to develop improved ABR flow control algorithms to address the above issues. Specifically, the major areas to be investigated are improved queue control and (max-min) bandwidth distribution. The algorithm should explicitly handle per-connection MCR requirements and provide efficient bandwidth distribution. Pertinent traffic measurements must be incorporated into the scheme to properly respond to network dynamics which may arise. Insensitivity to control parameters, especially *source* parameters, is also crucial. Generally, these settings are not under the control of network switch operators, and their effect on performance at network switches should be minimal. It is felt that to achieve the above objectives, *exact* fair share computation [14] coupled with per-connection accounting, is necessary. Approximate algorithms using multiplier factors may not provide a high enough degree of bandwidth resolution in a general, large-scale network setting.

Chapter 3

The EDERA Algorithm

In this chapter an *enhanced distributed explicit rate allocation* (EDERA) algorithm is proposed for ABR services support. The algorithm is an *exact* fair rate computation scheme [14], and implements the MCR-plus-equal-share [1],[13] bandwidth fairness criteria. Congestion indicators are incorporated into the rate allocation phase and provisions made for functionality in dynamic environments with interfering real-time CBR and VBR traffic flows. Since it is a distributed algorithm, EDERA resides at both the source and destination end-systems and each of the network switches. The operation at each of these network elements is now detailed.

3.1 Source End-System Algorithm

The pseudocode for the source end-system algorithm is shown in Fig. 3.1. The two major operations performed at the source end-systems are forward RM cell emission and backward RM cell processing (i.e., rate updates). The algorithm is

relatively simple but requires the end-systems to have countdown timers (increased complexity). Details are now presented.

Sources intermittently emit RM control cells after every block of $(N_{rm} - 1)$ data cells, and these cells contain values which are used by network switches to perform rate allocation. In particular, the emitted RM cells contain the connection's MCR, PCR, current ACR, and SASR (*segmental average source rate*, [51]), as shown in Fig. 3.2. The segmental averaging approach imposes moderate additional functionality at the end-systems, but provides valuable source activity information to the network switches. This approach is well suited for handling bursty ABR sources and does not adversely affect persistent sources. Essentially, the SASR is the *average* bandwidth usage of the connection, measured between successive forward RM cells:

$$SASR = \frac{N_{rm}}{\text{Inter-RM cell time}}. \tag{3.1}$$

The averaging block size of N_{rm} cells is chosen to coincide with the inter-RM cell spacings and thus reduce complexity. Note, however, that it is possible for the computed SASR to be temporarily larger than the current source ACR, i.e., if a decrease RM cell arrives between two consecutive forward RM cells. But since a compliant source can never transmit at a rate above its current ACR, the SASR value in the emitted forward RM cell is set equal to the minimum of the computed SASR and the current ACR (see Fig. 3.1). Note that Eq. (3.1) computes a very simple, short-term average. No doubt, more advanced tracking schemes can also be employed to derive longer-term (more accurate) estimates [15]. The explicit-rate

```
if (data cell to send)
 {
  /* check if RM cell to be sent */
  if (RM_count == N_rm)
   {
    SASR^i = ( N_rm / (current_time - last_RM_time) )
    SASR^i_cell = min(SASR^i, ACR^i)
    Send RM cell (MCR^i, ACR^i, PCR^i, SASR^i_cell)
    RM_count = 0 /* clear count */
    last_RM_time = current_time
   }
  else
   {
    Send data cell
    RM_count = RM_count + 1 /* increment count */
   }
  Set countdown timer to RM_timeout
 }

if (RM_timeout expired) /* check if source is idle */
 {
  /* Set count to full so that next cell is RM */
  RM_count = N_rm
  ACR^i = MCR^i /* reduce ACR */
  Send RM cell (MCR^i, ACR^i, PCR^i, 0)
  last_RM_time = -∞
 }

if (backward RM cell received)
 {
  /* update rate */
  ACR^i = min(ER^i_cell, max(MCR^i, min(PCR^i, ACR^i + N_rm · AIR)))
 }
```

Figure 3.1: Source end-system algorithm, connection i

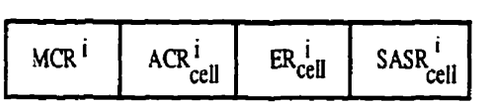


Figure 3.2: Resource management (RM) cell format for connection i

(ER) field in forward RM cells, ER^i_{cell}, field is initially set to the PCR by the source end-system and is reduced on the backward pass by network switches. All other fields are informative and hence static.

Before each data cell transmission a source must check to see if an RM cell needs to be emitted first. This is achieved by using a circular counter, *RM_count* (Fig. 3.1), ranging from 0 to N_{rm}. *RM_count* is incremented after each data cell transmission, and when it reaches $(N_{rm} - 1)$, the latest SASR is computed and an RM cell is emitted, Fig. 3.1. However, to further improve bandwidth utilization inside the network, sources which are idle for "long" intervals should notify the network by emitting appropriate RM cells. This is achieved using a countdown timer and a fixed timeout interval of duration *RM_timeout* sec. Specifically, the countdown timer is reset to *RM_timeout* after each cell transmission. However, if the timer expires, the source must reduce its ACR to its MCR and emit an RM cell with a *zero* SASR field. Subsequently, the first cell emitted after an idle period must be an RM cell. As will be evident later, the above mechanism makes sources relinquish their bandwidth allocations for idle periods greater than the timeout interval. Conversely, for idle periods which are smaller than the timeout interval, rate allocation is done contingent to the "average" usage, the SASR. The *RM_timeout* parameter is an appropriately chosen value, and is similar to the TRM timeout value defined in the ATM Forum ABR specifications [1].

Source-end systems also perform rate updates for arriving *backward* RM cells.

Specifically, the i-th source sets its ACR as

$$ACR^i = \min(ER^i_{cell}, \max(MCR^i, \min(PCR^i, ACR^i + N_{rm} \cdot AIR))), \tag{3.2}$$

where ER^i_{cell} is the value in the returning backward RM cell and AIR is a fixed rate increment (Fig. 3.1). This is an additive increase mechanism which always guarantees a source its MCR allocation, but avoids fast ramp-ups [1]. The maximum rate increase is limited to $N_{rm} \cdot AIR$ cells/second, but rate decreases, however, are not limited in magnitude. This allows for fast ramp-downs during congestion. Note that at startup, the connection's ICR is typically set to a small value, possibly below its MCR guarantee. The above mechanism prevents a new connection from transmitting immediately at its (possibly large) MCR rate and causing large queue buildups inside the network. However, after the initial additive ramp-up stage, the source is always allowed to transmit at its MCR, as guaranteed by the rate allocation algorithm at network switches (Section 3.3).

Additional features can also be added to the source end-systems to improve the cell loss performance of connections. These include rate decays [24],[28], backward RM cell timeouts [24], transient buffer exposures [46], etc. In general, these are conservative policies and result in more complex source end-systems behaviour. Furthermore, such overly cautious approaches can result in reduced bandwidth utilization inside the network. As a result, such these provisions are not considered for the EDERA scheme, and instead, more emphasis is placed on making the network switches guarantee connection goodputs. Therefore, in a sense, the EDERA source

behaviour can be considered as more aggressive.

3.2 Destination End-System Algorithm

The destination end-system algorithm is of minimal complexity, and is presented in Fig. 3.3. The main task performed by destination end-systems is the "echoing" back of forward RM cells to the source end-systems. Here, the incoming cell stream must be be monitored and a filtering function required to detect RM cells. To maintain compatibility with the existing ATM Forum specifications [1], return EFCI-bit marking is also performed for backward RM cells. In particular, the destination end-systems record the latest values of the EFCI-bits in arriving *data* cells. These bits can be set by non-EDERA switches along a connection's path to indicate forward congestion [12]. Hence, a binary flag, $Latest\_EFCI$, is appropriately set to the latest received EFCI-bit value, see Fig. 3.3. This value is then copied to received forward RM cells before they are echoed back. Additional refinements to the basic destination end-system algorithm are also possible, see [24].

3.3 Network Switch Algorithm

The core of the EDERA algorithm lies in each of the network switches where two major operations are performed: rate allocation and traffic measurement/filtering. The rate allocation step is performed upon the reception of forward or backward RM cells, whereas the measurement/filtering step is performed at fixed measurement intervals of duration T_{sw}. In the latter, measured values are used to determine

```
if (data cell received)
  {
   /* check if EFCI bit is set, record status */
   if (EFCI-bit on)
    { Latest_EFCI = 1 }
   else
    { Latest_EFCI = 0 }
  }

if (RM cell received)
  {
   /* Set latest EFCI status, send backward RM cell */
   Set EFCI-bit in RM cell to Latest_EFCI
   Send backward RM cell (MCR^i, ACR^i, PCR^i, SASR^i_cell)
  }
```

Figure 3.3: Destination end-system algorithm, connection i

congestion levels and compute the ABR available bandwidth, $\overline{C_a(k)}$, in a given measurement interval, k. Subsequently, a *usable* ABR capacity level, $C_u(k)$, is computed for the rate allocation phase. The EDERA scheme uses congestion indicators to "modulate" the amount of usable bandwidth, thereby controlling transient effects. Details are now presented.

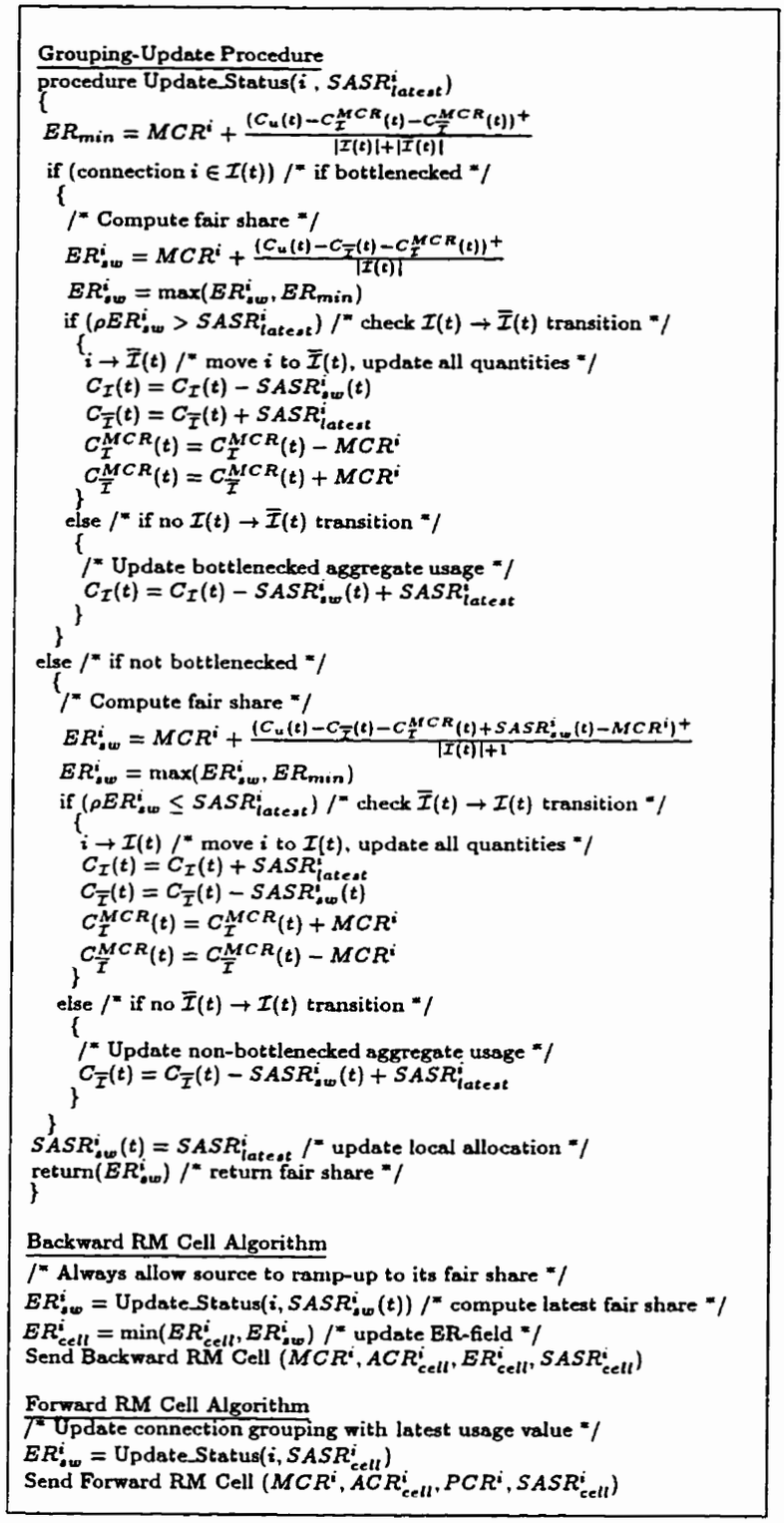
3.3.1 Rate Allocation

The rate allocation phase assumes that there is a given amount of bandwidth to be distributed amongst the ABR flows, the usable capacity $C_u(t)$. This amount is varied contingent to the congestion levels at the link, in order to control transient queue buildups, and its actual computation is done in the traffic measurement/filtering phase (to be described in Section 3.3.2). The rate allocation step proportions the usable capacity in a fair manner and attempts to match the overall ABR ingress (incoming) rate to the usable egress (outgoing) capacity. The algorithm implements MCR-plus-equal-share fairness, and further extensions to other criteria are

also possible (see Section 3.4). This criterion adequately handles a mixture of MCR requirements (i.e., both zero and non-zero) and simplifies to the ubiquitous max-min criterion [3] in the zero-MCR case. Rate allocation is based upon the SASR values in received RM cells (as opposed to the ACR values), and this improves bandwidth utilization with bursty/non-persistent sources. The pseudocode for the rate allocation algorithm is given in Fig. 3.4, and its details are now discussed. Various quantities are necessarily time-varying and explicitly denoted as such.

Each network switch maintains per-connection storages which are updated upon RM cell reception. First, a local copy of the latest usage for a given connection i, $SASR^i_{sw}(t)$, is stored, and this value is updated by the latest SASR values in *forward* RM cells (*Update_Status*$(i, SASR^i_{cell})$, Fig. 3.4). The actual source rate updates, meanwhile, are done for *backward* RM cells, ensuring that the switches use the most current fair share values and provide expedient responses. This is in contrast to doing both usage updates and rate allocations for only forward (backward) RM cells [37]-[42]. In addition, per-connection flags are also maintained for all connections sharing a link, grouping them into one of two disjoint sets: bottlenecked, $\mathcal{I}(t)$, or non-bottlenecked, $\overline{\mathcal{I}}(t)$. Ideally, $\mathcal{I}(t)$ is a subset of active connections, and only these should receive rate increases. Connections bottlenecked elsewhere or exhibiting non-persistent/idle behaviour are considered non-bottlenecked [3], and hence placed in $\overline{\mathcal{I}}(t)$. Although start-up connections can be placed in either grouping, they are put in $\overline{\mathcal{I}}(t)$ to improve transient bandwidth utilization.

Determining the $\mathcal{I}(t)/\overline{\mathcal{I}}(t)$ groupings in a distributed, delayed environment is difficult, but a general rule which can be used is that the bottleneck link will de-



Grouping-Update Procedure

procedure Update_Status(i, $SASR^i_{latest}$)
{
$ER_{min} = MCR^i + \frac{(C_u(t) - C^{MCR}_{\mathcal{I}}(t) - C^{MCR}_{\overline{\mathcal{I}}}(t))^+}{|\mathcal{I}(t)| + |\overline{\mathcal{I}}(t)|}$
if (connection $i \in \mathcal{I}(t)$) /* if bottlenecked */
{
/* Compute fair share */
$ER^i_{sw} = MCR^i + \frac{(C_u(t) - C_{\overline{\mathcal{I}}}(t) - C^{MCR}_{\mathcal{I}}(t))^+}{|\mathcal{I}(t)|}$
$ER^i_{sw} = \max(ER^i_{sw}, ER_{min})$
if ($\rho ER^i_{sw} > SASR^i_{latest}$) /* check $\mathcal{I}(t) \rightarrow \overline{\mathcal{I}}(t)$ transition */
{
$i \rightarrow \overline{\mathcal{I}}(t)$ /* move i to $\overline{\mathcal{I}}(t)$, update all quantities */
$C_{\mathcal{I}}(t) = C_{\mathcal{I}}(t) - SASR^i_{sw}(t)$
$C_{\overline{\mathcal{I}}}(t) = C_{\overline{\mathcal{I}}}(t) + SASR^i_{latest}$
$C^{MCR}_{\mathcal{I}}(t) = C^{MCR}_{\mathcal{I}}(t) - MCR^i$
$C^{MCR}_{\overline{\mathcal{I}}}(t) = C^{MCR}_{\overline{\mathcal{I}}}(t) + MCR^i$
}
else /* if no $\mathcal{I}(t) \rightarrow \overline{\mathcal{I}}(t)$ transition */
{
/* Update bottlenecked aggregate usage */
$C_{\mathcal{I}}(t) = C_{\mathcal{I}}(t) - SASR^i_{sw}(t) + SASR^i_{latest}$
}
}
else /* if not bottlenecked */
{
/* Compute fair share */
$ER^i_{sw} = MCR^i + \frac{(C_u(t) - C_{\overline{\mathcal{I}}}(t) - C^{MCR}_{\mathcal{I}}(t) + SASR^i_{sw}(t) - MCR^i)^+}{|\mathcal{I}(t)| + 1}$
$ER^i_{sw} = \max(ER^i_{sw}, ER_{min})$
if ($\rho ER^i_{sw} \leq SASR^i_{latest}$) /* check $\overline{\mathcal{I}}(t) \rightarrow \mathcal{I}(t)$ transition */
{
$i \rightarrow \mathcal{I}(t)$ /* move i to $\mathcal{I}(t)$, update all quantities */
$C_{\mathcal{I}}(t) = C_{\mathcal{I}}(t) + SASR^i_{latest}$
$C_{\overline{\mathcal{I}}}(t) = C_{\overline{\mathcal{I}}}(t) - SASR^i_{sw}(t)$
$C^{MCR}_{\mathcal{I}}(t) = C^{MCR}_{\mathcal{I}}(t) + MCR^i$
$C^{MCR}_{\overline{\mathcal{I}}}(t) = C^{MCR}_{\overline{\mathcal{I}}}(t) - MCR^i$
}
else /* if no $\overline{\mathcal{I}}(t) \rightarrow \mathcal{I}(t)$ transition */
{
/* Update non-bottlenecked aggregate usage */
$C_{\overline{\mathcal{I}}}(t) = C_{\overline{\mathcal{I}}}(t) - SASR^i_{sw}(t) + SASR^i_{latest}$
}
}
$SASR^i_{sw}(t) = SASR^i_{latest}$ /* update local allocation */
return(ER^i_{sw}) /* return fair share */
}

Backward RM Cell Algorithm

/* Always allow source to ramp-up to its fair share */
ER^i_{sw} = Update_Status($i, SASR^i_{sw}(t)$) /* compute latest fair share */
$ER^i_{cell} = \min(ER^i_{cell}, ER^i_{sw})$ /* update ER-field */
Send Backward RM Cell ($MCR^i, ACR^i_{cell}, ER^i_{cell}, SASR^i_{cell}$)

Forward RM Cell Algorithm

/* Update connection grouping with latest usage value */
ER^i_{sw} = Update_Status($i, SASR^i_{cell}$)
Send Forward RM Cell ($MCR^i, ACR^i_{cell}, PCR^i, SASR^i_{cell}$)

Figure 3.4: Network switch rate allocation algorithm, connection i

termine a connection's maximum rate. Hence, whenever an RM cell is received, the scheme checks if the connection "deserves" to be bottlenecked at the link. This is done by simply computing a fair share value for the connection as if it *were* bottlenecked and comparing the result to the latest usage, regardless of the current grouping. This mechanism maintains bandwidth fairness and also allows non-bottlenecked connections to increase their rate allocations. Note that the connection groupings are updated on *both* forward and backward RM cell passes, Fig. 3.4. Results show that this greatly improves responsiveness, justifying the added computational overhead. Details on the fair share computation are now presented.

Consider the following aggregate bandwidth quantities:

$$C_{\mathcal{I}}(t) = \sum_{i \in \mathcal{I}(t)} SASR^i_{sw}(t), \tag{3.3}$$

$$C_{\overline{\mathcal{I}}}(t) = \sum_{i \in \overline{\mathcal{I}}(t)} SASR^i_{sw}(t), \tag{3.4}$$

$$C^{MCR}_{\mathcal{I}}(t) = \sum_{i \in \mathcal{I}(t)} MCR^i, \tag{3.5}$$

and

$$C^{MCR}_{\overline{\mathcal{I}}}(t) = \sum_{i \in \overline{\mathcal{I}}(t)} MCR^i, \tag{3.6}$$

where $C_{\mathcal{I}}(t)$ and $C_{\overline{\mathcal{I}}}(t)$ are the total usages by bottlenecked and non-bottlenecked

connections, respectively, and $C_{\mathcal{I}}^{MCR}(t)$ and $C_{\overline{\mathcal{I}}}^{MCR}(t)$ are the total MCR guarantees of the bottlenecked and non-bottlenecked connections, respectively. The *elastic* bandwidth for bottlenecked connections is defined as the usable bandwidth *beyond* the aggregate MCR commitment to $\mathcal{I}(t)$ and aggregate bandwidth usage by $\overline{\mathcal{I}}(t)$, i.e., $(C_u(t) - C_{\overline{\mathcal{I}}}(t) - C_{\mathcal{I}}^{MCR}(t))$. To maintain MCR-plus-equal-share bandwidth fairness, it is required that this elastic bandwidth be distributed equally amongst the bottlenecked connections. Thus the fair share for connection i, ER_{sw}^i, is determined as the source's MCR guarantee plus an average portion of the elastic bandwidth:

$$ER_{sw}^i = \begin{cases} MCR^i + \left(\frac{C_u(t) - C_{\overline{\mathcal{I}}}(t) - C_{\mathcal{I}}^{MCR}(t)}{|\mathcal{I}(t)|}\right)^+, & i \in \mathcal{I}(t) \\ MCR^i + \left(\frac{C_u(t) - C_{\overline{\mathcal{I}}}(t) - C_{\mathcal{I}}^{MCR}(t) + SASR_{sw}^i(t) - MCR^i}{|\mathcal{I}(t)|+1}\right)^+, & i \in \overline{\mathcal{I}}(t), \end{cases} \tag{3.7}$$

where $|\cdot|$ denotes set cardinality and $(x)^+ = \max(0, x)$. Recall that the fair share of a connection in $\overline{\mathcal{I}}(t)$ is computed as if it belongs to $\mathcal{I}(t)$. Therefore, for connections in $\overline{\mathcal{I}}(t)$, the elastic bandwidth must be adjusted by $(SASR_{sw}^i(t) - MCR^i)$ and the cardinality incremented in the division (Eq. (3.7)). Eq. (3.7) assumes that MCR reservation is performed at connection establishment by the CAC function.

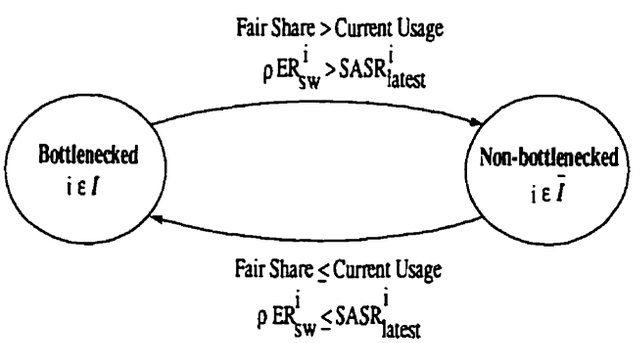


Figure 3.5: Grouping state-transition diagram, connection i

Using the computed ER^i_{sw} fair share, network switches appropriately adjust a connection's grouping as the network dynamics change. Namely, the source's latest SASR value ($SASR^i_{cell}$ on forward pass, $SASR^i_{sw}(t)$ on backward pass, see Fig. 3.4) is compared against a large fraction, ρ, of the computed fair share. If the usage is less than ρER^i_{sw}, the connection stays in or moves to (depending upon current grouping) the non-bottlenecked group, $\overline{\mathcal{I}}(t)$, and vice versa (as shown in Fig. 3.5). This requires a source to "fully" use its current allocation to be considered as bottlenecked. Hence the parameter ρ is basically a utilization threshold. Note that for proper bandwidth accounting, the aggregate bandwidth quantities must also be updated appropriately on both passes, Fig. 3.4. The rate allocation algorithm always attempts to move connections towards their fair share values. Furthermore, in order to give faster convergence, a given connection i is always allowed to send at its minimum "proportional" allocation at the link, ER^i_{min}:

$$ER^i_{min} = MCR^i + \left(\frac{C_u(t) - C^{MCR}_{\mathcal{I}}(t) - C^{MCR}_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)| + |\overline{\mathcal{I}}(t)|} \right)^+ . \tag{3.8}$$

In particular, the ER-field of backward RM cells is adjusted in a *downward* manner (see Fig. 3.4):

$$ER^i_{cell} = \min(ER^i_{cell}, \max(ER^i_{sw}, ER^i_{min})), \tag{3.9}$$

ensuring that returning cells contain the minimum ACR of all path switches, i.e., bottleneck rate. The convergence properties of the algorithm, assuming constant link capacities and persistent/constant sources, are discussed in Appendix D.

The above classification strategy performs bandwidth *overallocation* to improve efficiency: any unused bandwidth is given to the bottlenecked connections, even idle portions of the MCR guarantee for non-bottlenecked connections. The degree of overallocation is controlled by ρ, and larger values give improved link utilizations [51]. For example, smaller values of ρ make it "easier" for connections to enter the bottlenecked state and thus reduce the fair share values (i.e., since division by $|\mathcal{I}(t)|$ in Eq. (3.7)). In certain scenarios, this can result in bandwidth underutilization since non-bottlenecked connections can get (mis)classified as bottlenecked. As an example, consider the case where ρ is set to 0.5 at a link, but due to downstream bottlenecks, a certain group of connections can only use 75% of their fair share allocations. These connections will still be classified as bottlenecked and the unused 25% portion of their fair shares cannot be assigned to those connections which are *truly* bottlenecked at the link. Hence, to maximize bandwidth utilization, larger ρ values are necessary, e.g., 0.95. However, such settings can result in larger operating queue sizes.

3.3.2 Traffic Measurement/Filtering

This step uses traffic measurements to compute usable capacity levels (for rate allocation) and monitor congestion statuses. The operations are performed at the end of each measurement interval and thus all quantities change in a step-wise manner (i.e., at $t = kT_{sw}$). The pseudocode is shown in Fig. 3.6 and some details are now discussed.

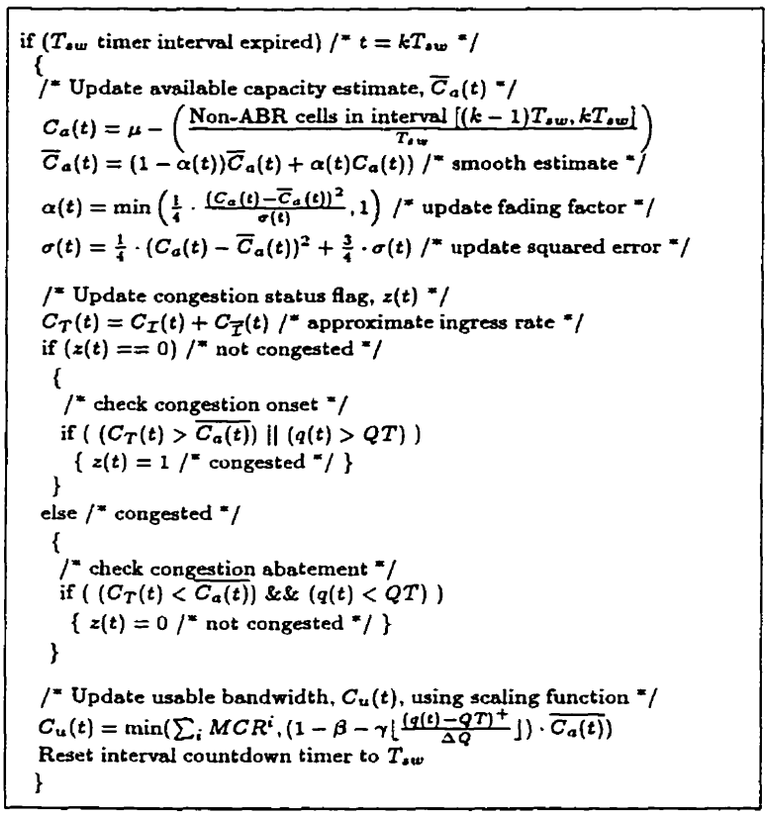


Figure 3.6: Network switch timer-interval (T_{sw}) algorithm

Capacity Tracking

This step tracks the available capacity levels for ABR traffic by measuring the *non*-ABR traffic usage, and is an essential part of the EDERA scheme. It has been shown that ABR control schemes which do not account for VBR traffic can suffer significant performance degradation [51],[54],[62]. Specifically, the number of non-ABR cells departing the link-buffer in the last measurement interval are counted, and from this an instantaneous estimate of the available ABR bandwidth value, $C_a(t)$, is computed (assuming output buffering switches):

$$C_a(t) = \mu - \left(\frac{\text{Non-ABR cells in } [(k-1)T_{sw}, kT_{sw}]}{T_{sw}} \right)^+, \quad kT_{sw} \leq t < (k+1)T_{sw}, \tag{3.10}$$

where μ is the link speed in cells/sec. Depending upon the background non-ABR traffic behaviour, the estimate of Eq. (3.10) can vary significantly due to short-term traffic burstiness (i.e., order of T_{sw} or less). Therefore, to reduce sensitivity to such effects, a weighted moving-average of $C_a(t)$, $\overline{C_a(t)}$, is computed using an approach similar to [49]:

$$\begin{aligned} \overline{C_a(t)} = (1 - \alpha((k-1)T_{sw})) \cdot \overline{C_a((k-1)T_{sw})} \\ + \ \alpha((k-1)T_{sw}) \cdot C_a(kT_{sw}), \quad kT_{sw} \leq t < (k+1)T_{sw}, \end{aligned} \tag{3.11}$$

where $\alpha(t)$ is a fading factor given by:

$$\alpha(t) = \min \left(\frac{1}{4} \cdot \frac{(C_a(kT_{sw}) - \overline{C_a(kT_{sw})})^2}{\sigma(kT_{sw})}, 1 \right), \quad kT_{sw} \leq t < (k+1)T_{sw}, \tag{3.12}$$

and $\sigma(t)$ is the weighted squared estimation error (see Fig. 3.6):

$$\sigma(t) = \tfrac{1}{4} \cdot (C_a(kT_{sw}) - \overline{C_a(kT_{sw})})^2 + \tfrac{3}{4} \cdot \sigma((k-1)T_{sw}), \quad kT_{sw} \leq t < (k+1)T_{sw}. \tag{3.13}$$

The adaptive fading factor $\alpha(t)$, as given by Eq. (3.12), can handle a wide range of fluctuations in background VBR traffic. When the VBR usages are slowly varying, $\alpha(t)$ will be small and Eq. (3.11) will yield smoother estimates. Conversely, if there is an abrupt change in traffic behaviour, $\alpha(t)$ will increase and track faster than a fixed (small) value. Nevertheless, very large short-term fluctuations in the VBR traffic patterns will still give rapidly changing estimates. In such highly bursty environments, however, the overall applicability of feedback flow control schemes is questionable. Typically, such schemes are most effective when traffic patterns vary on time scales *larger* than the roundtrip delays [8].

Congestion Detection

Subsequent to capacity tracking, congestion statuses are updated and the ABR usable bandwidth, $C_u(t)$ (Section 3.3.1), is computed. This is done by appropriately lowering the bandwidth utilization in response to the congestion levels at the link. Although several different types of congestion indicators can be used [12], it is generally known that rate-based indicators are significantly better than simple queue thresholds in detecting *impending* congestion, i.e., potential queue buildups [13],[32]. Rate-based overload indicators basically signal congestion whenever the ingress rate to a transmission link exceeds its available capacity, $\overline{C_a(t)}$. However,

results show that such indicators alone are inadequate in dynamic environments with interfering VBR traffic [50],[54],[62]. Due to feedback delays, inevitably, there will always be queue buildups. Furthermore, the transient capacity overallocation feature of the algorithm (Section 3.3.1) can add to the problem. Therefore the situation can arise where queue levels gradually increase over time if solely rate overload indicators ares used. This positive queue "drift" phenomenon has also been noted in [54] with the ERICA scheme, which uses a rate overload congestion indicator (i.e., with interfering VBR traffic, see [54], Fig. 9). Hence *both* rate and queue congestion indicators must be incorporated in the EDERA scheme.

The pseudocode for the congestion detection algorithm is shown in Fig. 3.6. For rate-overload indication, the ABR ingress rate is simply approximated by the total usage by all traversing ABR connections, $C_T(t)$:

$$C_T(t) = \sum_i SASR^i_{sw}(kT_{sw}) = C_{\mathcal{I}}(kT_{sw}) + C_{\overline{\mathcal{I}}}(kT_{sw}), \qquad kT_{sw} \leq t < (k+1)T_{sw}. \tag{3.14}$$

It should be noted that due to inter-RM cell times, Eq. (3.14) gives slightly longer delays in detecting rate changes. Furthermore, since rate behaviour is bound to change as traffic propagates through the network, $C_T(t)$ may not be a very accurate estimate of the true ingress rates. Hence it may be better to explicitly measure ingress rates if improved, more accurate rate estimates are required. For queue threshold indication, let $q(t)$ be the queue length (or a smoothed estimate thereof), and QT be a queue threshold. The rate and queue congestion indicators are combined, signalling congestion whenever total capacity usage $C_T(t)$ exceeds the

available capacity or $q(t)$ exceeds the threshold QT. To facilitate this description, consider:

$$z(t) = \begin{cases} 1 & \text{if } q(kT_{sw}) \geq QT \text{ or } C_T(kT_{sw}) \geq \overline{C_a(kT_{sw})} \\ 0 & \text{otherwise} \end{cases}, \quad kT_{sw} \leq t < (k+1)T_{sw}, \tag{3.15}$$

be a combined congestion indicator function. Using $z(t)$, the amount of usable ABR bandwidth $C_u(t)$ is varied as (see Fig. 3.6):

$$C_u(t) = \max\left(\sum_i MCR^i \ , \ \left(1 - \beta \cdot z(kT_{sw}) - \gamma \left\lfloor \frac{(q(kT_{sw}) - QT)^+}{\Delta Q} \right\rfloor \right) \cdot \overline{C_a(kT_{sw})} \right), \quad kT_{sw} \leq t < (k+1)T_{sw}, \tag{3.16}$$

where β, γ, and ΔQ are appropriately chosen constants. The inner-bracketed term in Eq. (3.16) represents a step-wise, linear capacity scaling function, as shown in Fig. 3.7. Here, the usable capacity is further reduced by a fraction γ, beyond the initial β, for every block of ΔQ cells beyond QT. A similar function is also used in [48], and other non-linear types have also been proposed [46]. The congestion detection step attempts to keep operating queue levels in a zone about the threshold value, thereby sacrificing connection delays for improved utilizations. As such, this strategy operates beyond the *knee* point (i.e., pure congestion avoidance) of the throughput-load curve [8].

A tradeoff arises in choosing the bandwidth scaling parameters, β and γ. Although larger values are preferred to quickly reduce queue buildups, smaller values are required for improved bandwidth utilization and reduced oscillations. Simulations also show that in some networks, $\gamma = 0$ can be used, given that β is set

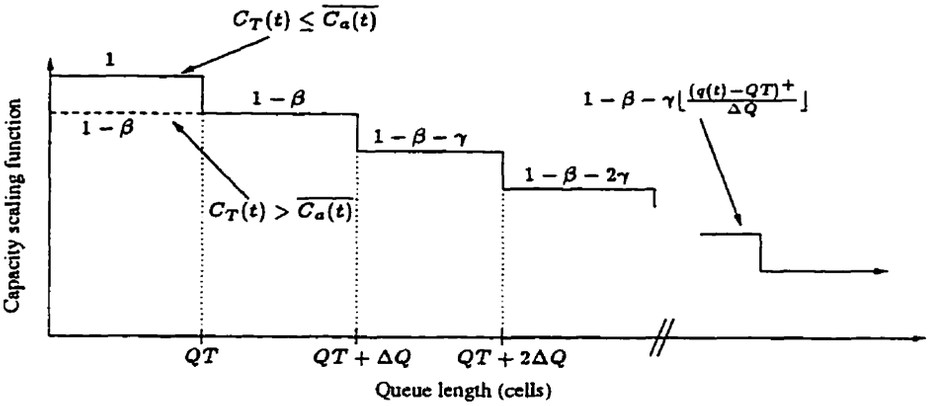


Figure 3.7: Linear step-wise queue length scaling function

to a large enough value. In other words, *fixed* capacity scaling can be done if the queue drain rate is set to a sufficiently large value. Nevertheless, it is difficult to determine beforehand what this value should be for an arbitrary network. Overly conservative settings (i.e., large β) can cause significant oscillations and resource underutilization. Also note that if $\beta, \gamma = 0$ (i.e., congestion detection disabled), the algorithm simplifies to a strict capacity allocation scheme (as presented in [51]).

Non-Oscillatory Condition

For the special case of *constant* capacity links and fixed bandwidth scaling during congestion (i.e., $\gamma = 0$), the scheme can give non-oscillatory steady-state behaviour for the proper choice of β. It is assumed that all sources are either persistent or constant and that the connection sets are fixed and/or varying over sufficiently large time intervals (i.e., greater than transient queue buildup and reduction times).

Consider the network modelled as a graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of

switches and $\mathcal{L}$ the set of inter-connecting *directional* links (as per the notation defined in Section D.2). Link $j \in \mathcal{L}$ has capacity μ_j and supports a set of traversing connections, $\mathcal{A}_j$. This set comprises bottlenecked, $\mathcal{I}_j$, and non-bottlenecked connections, $\overline{\mathcal{I}}_j$, as given by the bandwidth fairness criterion (i.e., $\mathcal{A}_j = \mathcal{I}_j \cup \overline{\mathcal{I}}_j$). Assuming that all connection rates are relatively close to their ideal, fair values, no *steady-state* rate oscillations occur if $\sum_{i \in \mathcal{A}_j} MCR^i < \mu_j \ \forall j \in \mathcal{L}$ and $\beta \leq \beta^*$:

$$\beta^* = \min_{j \in \mathcal{L}} \left\{ \min_{i \in \mathcal{I}_j} \left\{ (1-\rho)\left(1 - \frac{C_{\overline{\mathcal{I}}_j} + C_{\mathcal{I}_j}^{MCR} - |\mathcal{I}_j| MCR^i}{\mu_j}\right) \right\} \right\}, \tag{3.17}$$

where $\beta^* \leq (1-\rho)$ and $C_{\overline{\mathcal{I}}_j}$ and $C_{\mathcal{I}_j}^{MCR}$ are the ideal steady-state values of the aggregate non-bottlenecked usages and aggregate bottlenecked MCR guarantees, respectively, at link j (and hence no dependence on time is shown). The derivation of the above condition is presented in Appendix E.

Intuitively, $\beta \leq \beta^*$ prevents $\mathcal{I}_j \rightarrow \overline{\mathcal{I}}_j$ transitions, thereby eliminating subsequent rate overshoots (i.e., $|\mathcal{I}|$ stays the same in Eq. (3.7)). Hence $\beta^* \mu_j$ cells/sec. is the maximum rate at which the link j queue can be drained to prevent oscillations. Since $\beta^* \leq (1-\rho)$, clearly there is a tradeoff here. Typically, larger values of ρ are required for improved utilization, but larger values of β are also desired to improve responsiveness. Therefore the price paid for non-oscillatory performance is longer transient times for queue reduction. Similar tradeoffs have also been observed with schemes using differential-equation modelling [80] and control-theoretic formulations [60]. Note, though, that the applicability of the above result in dynamic network environments is limited.

3.4 Algorithm Properties

Although the EDERA algorithm is of the same flavour as the basic distributed explicit rate allocation schemes [38],[42],[51], it has several major enhancements. First of all, EDERA has explicit provisions for functioning in heterogeneous networks carrying non-ABR traffic (i.e., capacity tracking step, Section 3.3.2). In addition, added provisions for controlling queue sizes during transience are given. Most other exact ER schemes do not *directly* address this issue [38],[42],[43],[50]. Although using *delayed* rate updates to eliminate queue buildups has been proposed [38], such an approach is not entirely effective. There are added per-connection overheads, and reliable delay estimates (propagation, processing, buffering) are required. Furthermore, this approach is not effective in dynamic environments with fluctuating capacity levels.

The EDERA scheme also explicitly handles MCR guarantees, since the MCR-plus-equal-share criterion is inherent in all the rate update computations. In particular, appropriate MCR usages are accounted for (i.e., Eq. (3.5)), as opposed to simply removing the aggregate MCR guarantees from the available capacity beforehand. Hence this approach is more efficient, detecting idle portions of the MCR guarantee for non-bottlenecked connections and allocating them to bottlenecked connections (i.e., since Eq. (3.7) removes $C_{\overline{I}}(t)$ as opposed to $C_{\overline{I}}^{MCR}(t)$). In addition, it should be mentioned that extensions to several other bandwidth fairness criteria are straightforward. For example, consider the more general *MCR linear* criterion [25], where there is a weight associated with each connection i, w^i. This can be implemented by modifying Eq. (3.7) to scale the elastic bandwidth by the

relative amount:

$$ER_{sw}^{i} = \begin{cases} MCR^{i} + \frac{w^{i}(C_u(t) - C_{\overline{\mathcal{I}}}(t) - C_{\mathcal{I}}^{MCR}(t))^{+}}{\sum_{k \in \mathcal{I}(t)} w^{k}}, & i \in \mathcal{I}(t) \\ MCR^{i} + \frac{w^{i}(C_u(t) - C_{\overline{\mathcal{I}}}(t) - C_{\mathcal{I}}^{MCR}(t) + SASR_{sw}^{i}(t) - MCR^{i})^{+}}{\sum_{k \in \mathcal{I}(t)} w^{k} + w^{i}}, & i \in \overline{\mathcal{I}}(t). \end{cases} \quad (3.18)$$

For the special case of $w^i = MCR^i \ \forall i$, the above gives the *MCR-proportional criterion* [25]. However, the latter may be considered as unfair for zero-MCR connections. This is one of the reasons for choosing the simpler MCR-plus-equal share criterion. In addition, for zero MCR connections, Eq. (3.7) reduces to the *weighted* criterion [25],[26]. A slightly different, hybrid criterion, the *maximum-of-MCR-or-fair-share* has also been proposed [25]. This criterion can be a straightforward method of extending some of the previous exact computation schemes which do not have explicit provisions for MCR guarantees. However, such a paradigm can unnecessarily restrict larger MCR connections from achieving additional throughput gains. Considering that such users will usually have paid a premium for their service, this bias can be unacceptable.

The control scheme has $O(n)$ storage complexity, where n is the number of ABR connections at the link. Such storage overhead, though, is typical of most advanced explicit rate schemes [38],[42],[46]. Furthermore, it is felt that the storage requirements are justified considering the complexity of the ABR flow control problem and the fact that hardware memories are relatively inexpensive. The rate update algorithm itself is of $O(1)$ computational complexity, similar to the recent ERAA algorithm in [42]. From an implementation point of view, this is an improvement over other distributed explicit rate allocation schemes which are up to $O(n)$

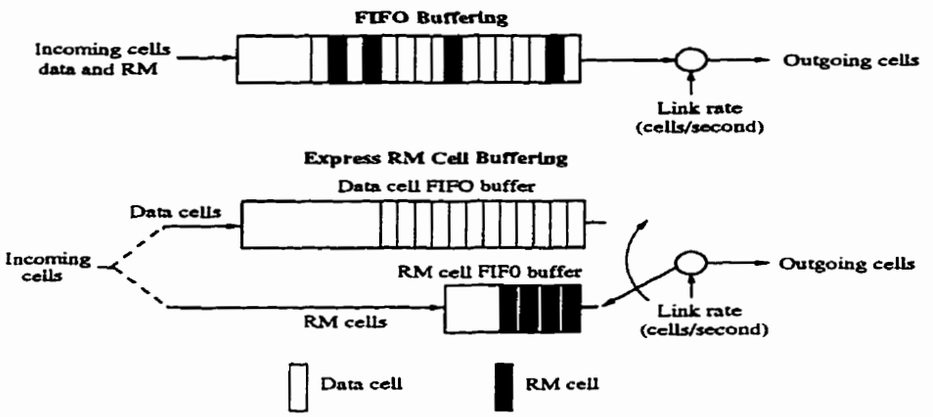


Figure 3.8: ABR buffering schemes: FIFO and express RM cell queueing

computational complexity [38]. Nevertheless, since division operations are still required, the implementation complexity can be significant with such schemes. The algorithm in Section 3.3.1 basically attempts to maintain correct (i.e., *consistent*) connection groupings as well as reduce the computational requirements. This is in contrast to the more computationally-intensive algorithms in [37],[40], where *all* connection groupings are updated upon RM cell arrival, i.e., connection groupings are *always* consistent. The EDERA rate allocation algorithm does not have this property, and therefore bottlenecked connections can be temporarily (mis)classified as non-bottlenecked. This incremental updating of connection groupings has a subtle impact on the convergence properties of the scheme and can result in increased transient oscillations (see Appendix D for convergence analysis).

Finally, to improve the responsiveness of the feedback, it is suggested that RM cells be separately buffered and given non-preemptive service priority over data cells: *express RM cell queueing*, Fig. 3.8. This approach does not violate the ATM Forum specifications, since it maintains continuity within ABR data and RM

cell streams, but not necessarily *between* them [1]. Simulation results show that the queueing performance is greatly improved via this enhancement, especially in WAN networks, since data cells do not delay rate reduction RM cells in periods of congestion. Note that this enhancement has also been applied to other ABR schemes [13].

Chapter 4

Performance Evaluation

In this chapter, the performance of the proposed EDERA flow control algorithm is evaluated. It is desired to investigate real-world network settings over a wide range of operating environments. Typically, however, for ABR services, the non-linear nature of many of the control schemes coupled with asynchronous, delayed feedback environments pose many problems for analytical treatments. In most cases, the analysis is intractable beyond simple, single link networks. Although some worst case transient analysis can be done to bound queue buildups for the EDERA algorithm, several major assumptions are necessary to simplify the development (see Appendices F,G). These include persistent sources, direct source ramp-ups (as opposed to additive ramp-ups, Section 3.1), and strictly capacity allocation at network switches (i.e., β, $\gamma = 0$, Section 3.3.2). Hence, the practical applicability of such analytical results may be very limited. As a result, the use of simulation techniques to realistically investigate the performance of feedback control schemes is very well justified. In fact, this approach is the method of choice amongst most

performance evaluation studies on ABR flow control [14],[28],[35]-[41],[47],[48],[55].

In this chapter, an extensive simulation study of the EDERA scheme is presented, with both single and multiple link scenarios being studied. In particular, many of the scenarios chosen are very similar to those in Chapter 2, in order to maximize the comparison with the EPRCA scheme. However, additional results are also presented, where applicable, to show specific properties of the EDERA scheme. In addition, a brief overview of the software simulator is presented in Appendix H.

4.1 Single Link Scenarios

Initially, simpler, single link environments are tested to verify basic properties of the algorithm (i.e., queue control, fairness, etc.). All results are gathered over two independent runs of one second duration each, and the main parameters settings are summarized in Table 4.1. Here, the bandwidth utilization threshold is set to a high value (95%) and the desired queue thresholds to nominal values. Furthermore, the source end-system timeout value, *RM_timeout*, is set to a relatively large value

Table 4.1: EDERA parameter settings

Parameter	Setting
Link rates	150 Mbps
N_{rm}	32 cells
T_{sw}	0.09 ms (32 cell times)
ρ	0.95
AIR, ICR	100 kbps, 200 kbps
QT, ΔQ	200 cells, 200 cells
RM_timeout	100 ms

(i.e., 100 ms, as suggested in [1] for the TRM parameter). Larger values allow bursty ABR sources to maintain higher rates, and place more burden on the scheme to function with possibly larger loadings and less accurate source averages. For comparison purposes, all EPRCA settings are unchanged from those in Table 2.1.

4.1.1 Scalability Scenario

Consider the scalability scenario presented in Section 2.2.2, Fig. 2.1. This network is tested using moderate bandwidth scaling factors, namely $\beta, \gamma = 0.05$. The maximum queue buildups with the EDERA scheme are plotted in Fig. 4.1 for $N = 10$ and $N = 20$ connections, along with those from the EPRCA scheme (i.e., Fig. 2.2). The plot indicates that the maximum queue buildups with both schemes are almost the same for this scenario and indicate a roughly linear dependence on the propagation delay, δ. Furthermore, the corresponding link utilizations are also plotted in Fig. 4.2. Here, the performance of the two schemes differs, though, with EDERA showing considerably better link utilization at larger WAN-like delays. To show this more clearly, sample queueing behaviour for a WAN-delay scenario ($N = 10$, $\delta = 3$ ms) is shown in Fig. 4.3. The EPRCA scheme yields sizable queue empty periods, induced by the oscillatory behaviour. The corresponding oscillations in the EDERA scheme are slightly smaller, but no bandwidth underutilization occurs (i.e., no queue empty periods). Although larger delays generally increase the magnitude of the oscillations in both schemes, maintaining high link utilizations is still desirable. The EPRCA bandwidth utilization can be improved by increasing the queue threshold, QT, to roughly 500 cells. However, this gives significantly larger

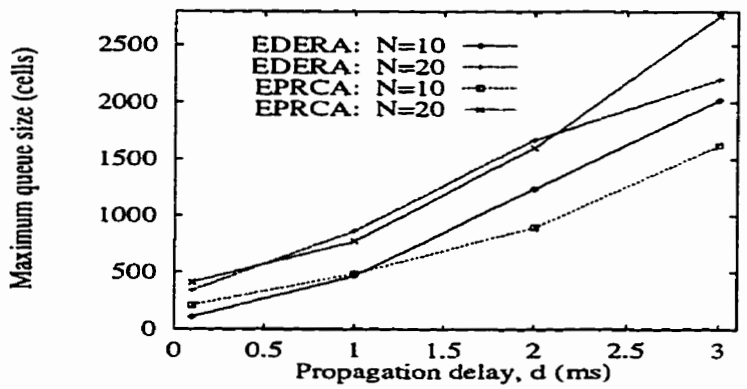


Figure 4.1: Maximum queue sizes in Fig. 2.1

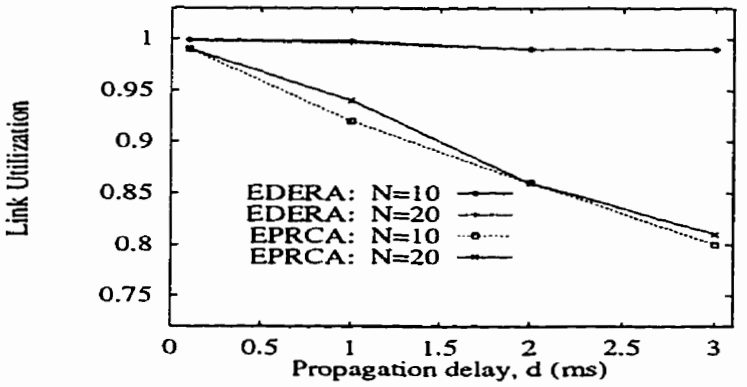


Figure 4.2: Average normalized throughputs in Fig. 2.1

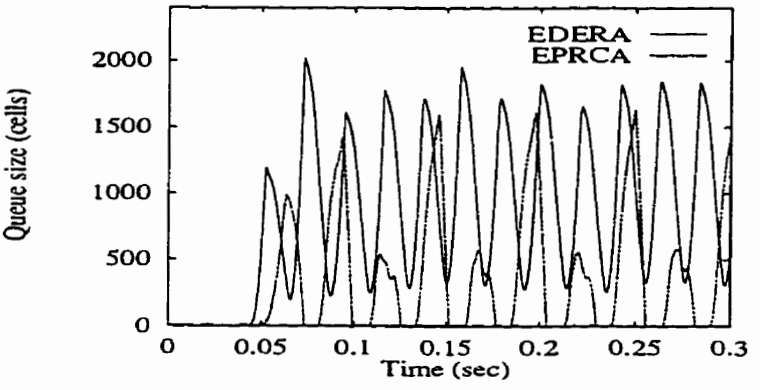


Figure 4.3: Queue behaviour in Fig. 2.1, $N = 10$, $\delta = 3$ ms

operating queue sizes, i.e., over 50% increase. The improved EDERA queue control is a result of the combined congestion indication mechanism (Section 3.3.2).

4.1.2 Bandwidth-Delay Scenario

Now consider the more advanced bandwidth delay scenario in Fig. 2.5. First the network is tested for the ABR-only case, where all connections should ideally receive 5 Mbps (i.e., VBR connections idle, Table 2.2, and $\beta, \gamma = 0.05$). The measured steady-state throughputs are shown in Table 4.2 and indicate very fair bandwidth distribution, even better than EPRCA (see Table 2.3). Note that the longer delay connections, connections 21-30, get slightly more throughput since their increased delay proximities make them more difficult to control. The corresponding queueing behaviour at the link is also graphed in Fig. 4.4 against that yielded by the EPRCA scheme. In this scenario, it is clear that the EDERA scheme provides better queueing performance. Not only are the buildups smaller, but the magnitude of the oscillations is much smaller.

Table 4.2: EDERA bandwidth-delay performance, Fig. 2.5 (Mbps)

Connections 1-10 Throughput	Connections 11-20 Throughput	Connections 21-30 Throughput	Bandwidth Utilization	Throughput Deviation
4.91	4.99	5.07	99%	1.1%

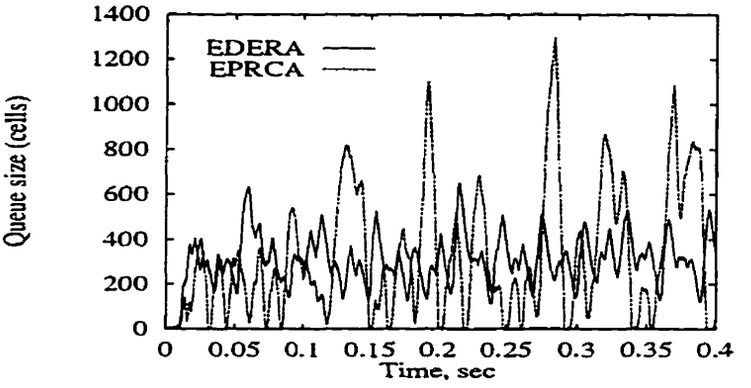


Figure 4.4: Queue behaviour in Fig. 2.5, idle VBR

4.1.3 Parameter Sensitivity Scenarios

Still considering the network in Fig. 2.5, some parameter sensitivity tests are conducted. Clearly, many EDERA (and ABR) parameters can effect the performance of the scheme (i.e., queue thresholds, bandwidth scaling constants, utilization thresholds, etc.). However, the bandwidth scaling constants β and γ (or equivalently ΔQ) have the most direct effect on the transient performance since they explicitly control the amount of bandwidth given to ABR sources (via. Eq. (3.16)). Along these lines, the network is tested for sensitivity to these parameters. Fig. 4.5 plots the throughput deviation, Eq. (2.4), measured over all *groups*, for differing β and γ settings. The results indicate that smaller β and γ values, between 0.05 and 0.1, give considerably better fairness. For example, with $\beta = 0.1$ and $\gamma = 0.05$, the throughput deviation is below 0.1 Mbps, indicating that on average, connections are getting within 2% of their ideal allocation of 5 Mbps. The mean and standard deviation of the queue length are shown in Fig. 4.6 and Fig. 4.7, respectively. The mean value is higher for smaller β, and decreases slightly for increasing

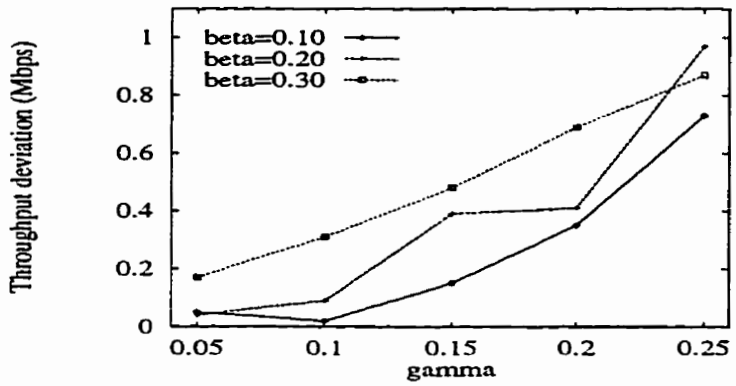


Figure 4.5: Throughput deviation in Fig. 2.5

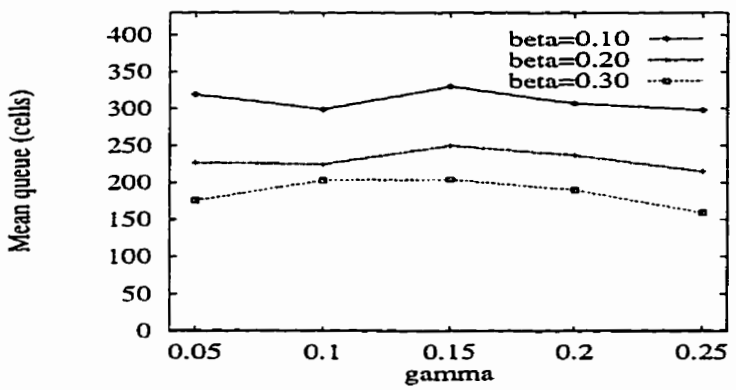


Figure 4.6: Mean of queue sizes in Fig. 2.5

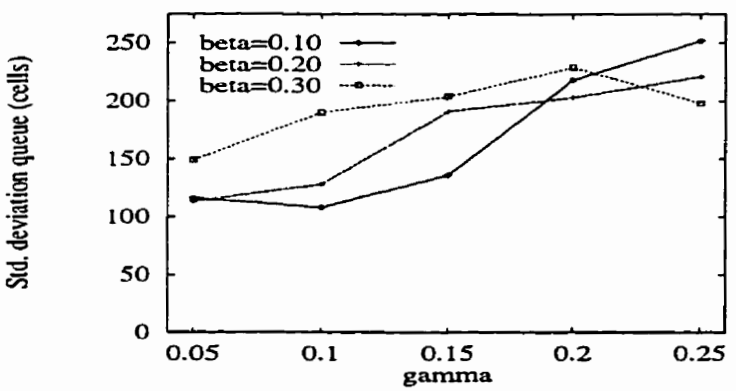


Figure 4.7: Standard deviation of queue sizes in Fig. 2.5

γ values. However, the oscillations increase significantly with larger β and γ values, and this can result in larger possible *maximum* buildups. Intuitively, it may be expected that larger values of γ will give smaller queue buildups. However, in reality, sharper ramp-downs increase the magnitude of subsequent rate oscillations, and this can lead to increased queue buildups. Although not shown, larger values of γ also result in reduced link utilizations (i.e., roughly 90% for $\gamma = 0.30$ vs. 98% for $\gamma \leq 0.10$). In general, the results show that larger β values give increased queue oscillations. Now since the β parameter merely removes a "fixed" portion of the availably capacity, it alone can only give two different capacity levels (i.e., similar to bang-bang approach [57]). Therefore, in practice, it is best to fix β to a nominal value (e.g., 0.05 or 0.10) and only adjust γ for a given network.

The AIR parameter sensitivity tests conducted for the EPRCA scheme using the scenario in Fig. 2.5 (Section 2.2.2) are also repeated with the EDERA scheme (for $\beta, \gamma = 0.05$). Specifically, the AIR values of the longer delay connections, connections 21-30, are varied. The average throughputs for all connection groups are graphed in Fig. 4.8, and show relatively small sensitivity to discrepancies in the AIR values. For example, even for $AIR = 200$ kbps, connections 21-30 only get on average 12% above their ideal mean fair share of 5 Mbps. Furthermore, comparison with the EPRCA throughputs, also plotted in Fig. 4.8 (and even PRCA throughputs, Fig. 2.7), indicates much less sensitivity in the EDERA scheme. The exact rate computations performed by the scheme prevent the larger AIR connections from adversely affecting the others. For completeness, the queue buildups are also compared in Fig. 4.9. By preventing larger rate increments for the longer delay

connections, EDERA also reduces the operating queue sizes significantly.

4.1.4 Non-Oscillatory Condition

The non-oscillatory condition of Section 3.3.2, is verified for the bandwidth-delay network scenario in Fig. 2.5. Since there are no MCR guarantees and all connections are bottlenecked at the link, the β^* value for this network is equal to 0.05 (i.e., $\beta^* = (1 - \rho)$, Eq. (3.17)). Hence the test case is run with $\beta = 0.05$, $\gamma = 0$, and sample rates from each connection group are shown in Fig. 4.10. In addition, the resulting queueing behaviour at the link is shown in Fig. 4.11 (along with that for the suggested settings of $\beta, \gamma = 0.05$, i.e., oscillatory choice). The plots indicate that there is an initial transient phase as the longer delay connections ramp-up and queue buildups are drained. This lasts for about 150 ms, after which the queue length falls below the QT threshold of 200 cells. Subsequently, the connections ramp up for the remaining bandwidth, $\beta^*\mu$ cells/second, without inducing further oscillations, and stabilize at equal allocations of 5 Mbps. Fig. 4.11 confirms that the non-oscillatory parameter settings give a much longer transient queue reduction phase. Furthermore, since the drain rate is considerably smaller, the resulting transient queue buildups are much larger than those with the "faster-response" oscillatory settings (i.e., 1114 cells versus 650 cells, Fig. 4.11).

4.1.5 Bursty VBR Scenario

The effect of bursty VBR traffic on the performance of the EDERA algorithm is also investigated. Consider the network of Fig. 2.5 with each VBR source now defined

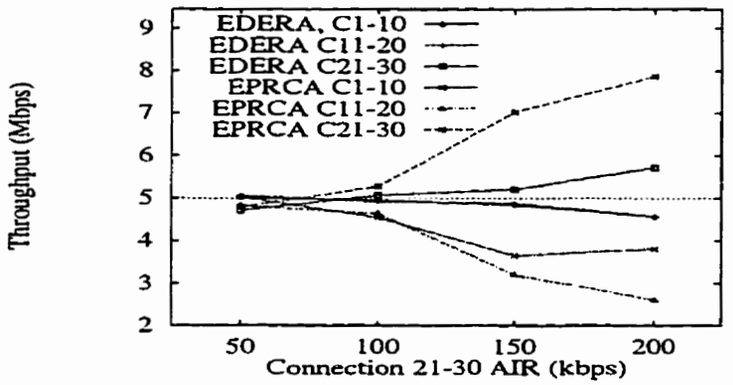


Figure 4.8: Throughput performance in Fig. 2.5, idle VBR

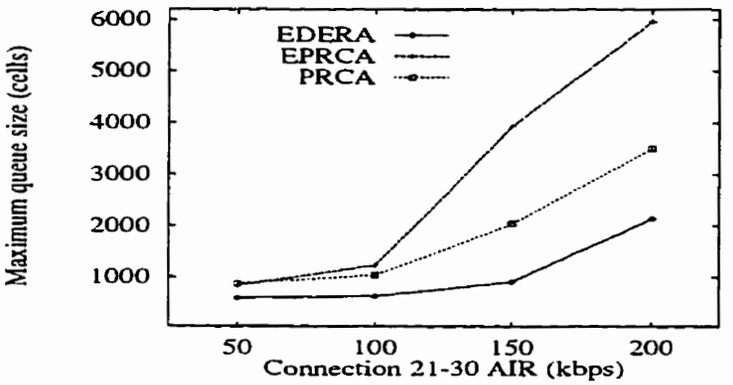


Figure 4.9: Maximum queue sizes at link in Fig. 2.5, idle VBR

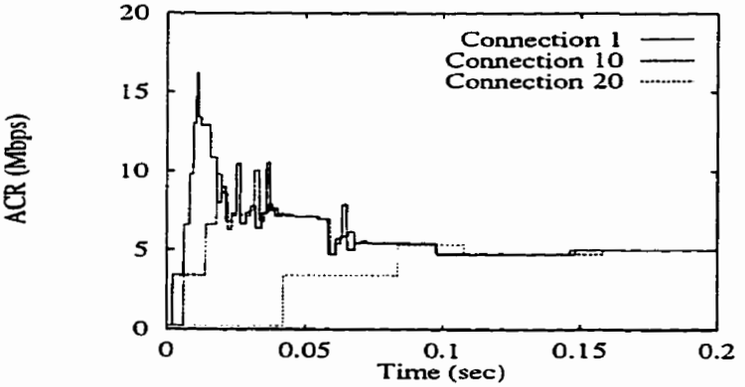


Figure 4.10: Sample connection ACR values, Fig. 2.5 ($\beta = 0.05$, $\gamma = 0$)

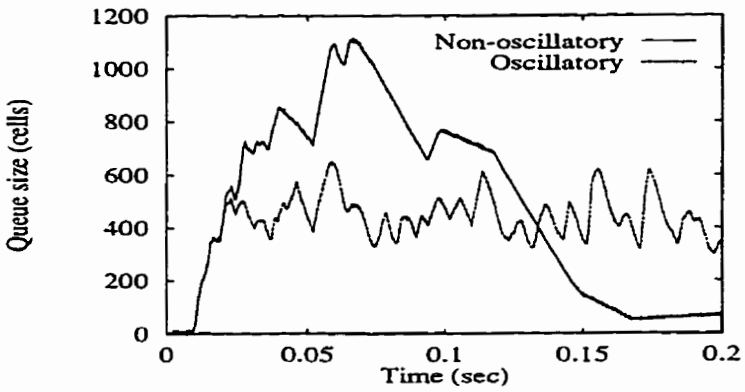


Figure 4.11: Link queue sizes in Fig. 2.5

as bursty on-off. The on and off periods are identically distributed with exponential means, and each source generates cells with fixed inter-cell arrival times of 35 μs in the on period, i.e., approximately 6 Mbps mean usage. This gives ideal mean ABR throughputs of 3 Mbps each. First, some additional parameter sensitivity tests are done with the γ parameter. For this test case the mean on and off periods of the VBR sources are fixed set to 1 ms each.

Fig. 4.12 shows the measured throughputs for each connection group, with the horizontal line representing the ideal mean goodput of 3 Mbps ($\beta = 0.10$). The plot shows that increased values of γ degrade the bandwidth fairness properties of the scheme, as was observed in the idle VBR case (Fig. 4.5). For example, for $\gamma = 0.25$, the shorter delay connections, connections 1-10, receive under 90% of their fair share allocation. Although not shown, however, the link utilization is still very high for all γ settings in this scenario. The mean and standard deviation of the link queue length are shown in Fig. 4.13. Here, increasing γ reduces the overall mean queue length, but causes a slight increase in its variance (i.e., more oscillations with larger capacity ramp-downs). Overall, the results confirm the earlier-specified

heuristic of employing smaller bandwidth scaling parameters to improve bandwidth fairness performance.

To measure the effect of differing VBR behaviour, the bursty VBR scenario presented in Section 2.2.2 for the EPRCA scheme is evaluated. Specifically, still considering Fig. 2.5, the means of the on and off periods of the VBR sources are varied, keeping the average usage fixed at 6 Mbps. This is done to gauge the sensitivity of the scheme to the burst durations of background traffic. The resultant EDERA throughputs are shown in Fig. 4.14 for $\beta, \gamma = 0.05$, alongside with those with EPRCA (Fig. 2.10). These results indicate that the capacity tracking mechanism (Section 3.3.2) performs well, with the rate allocations showing little sensitivity to the VBR burst durations. All connections get within 0.3 Mbps of the ideal 3 Mbps (i.e., throughput deviation under 10%), and the longer delay connections receive slightly larger allocations. Comparison with the EPRCA throughputs shows a clear improvement in the bandwidth fairness. In addition, the corresponding queue length metrics are plotted in Fig. 4.15. Both quantities are well below those obtained with EPRCA, and only increase marginally for larger burst durations (under 20%). These gains are a direct result of the capacity tracking function and fast congestion indicator mechanisms of the EDERA scheme.

4.1.6 Non-Persistent Source Scenario

As demonstrated in Section 2.2.2 (*Non-Persistent Source Scenario*), non-persistent, bursty ABR sources can be problematic for the EPRCA control scheme. The approximate nature of the scheme gives sluggish responses to changes in connection

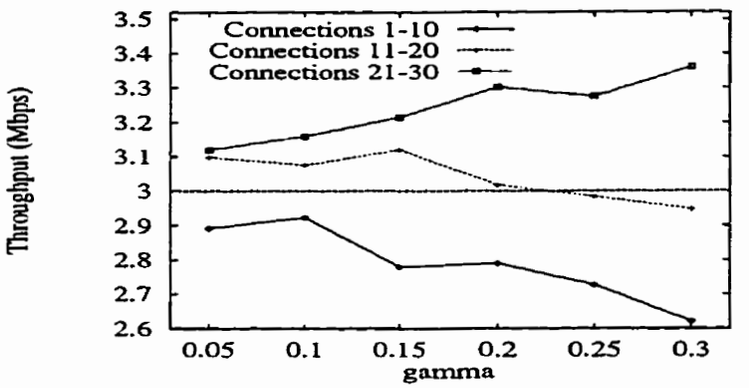


Figure 4.12: Throughputs in Fig. 2.5, bursty VBR ($\beta = 0.10$)

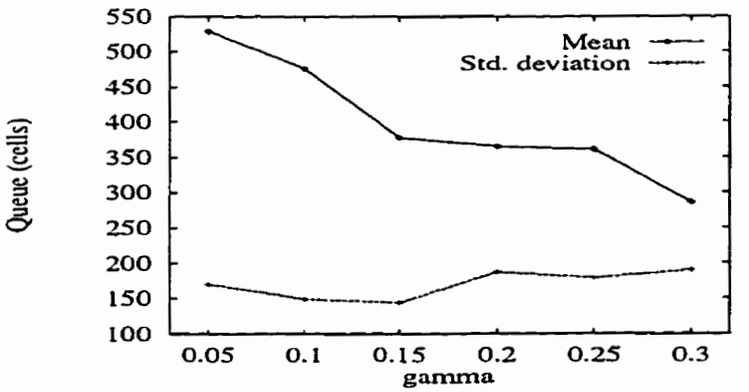


Figure 4.13: Link queue in Fig. 2.5, bursty VBR ($\beta = 0.10$)

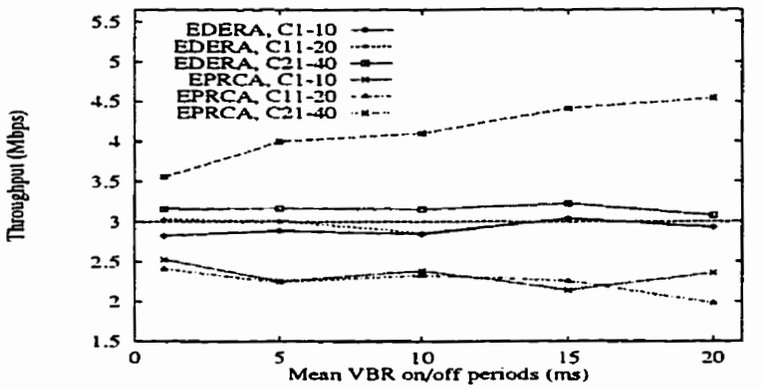


Figure 4.14: EDERA throughputs in Fig. 2.5, bursty VBR

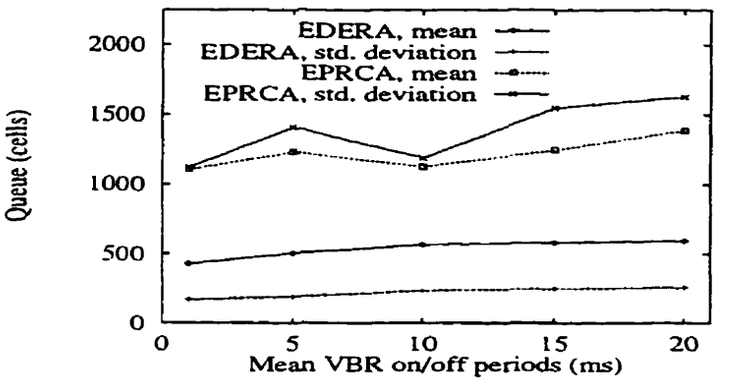


Figure 4.15: Link queue in Fig. 2.5, bursty VBR

usages and prevents bursty sources from ramping up quickly. This reduces fairness and can also lead to larger queue buildups. To this end, segmental averaging is proposed at the EDERA source end-systems, Section 3.1. Consider the single link network of Fig. 2.5, where connections 1-10 are non-persistent with varying duty cycles (Eq. (2.5)). Fig. 4.16 plots the EDERA throughputs for different duty cycles, with the two straight lines representing the ideal fair share allocations for each group of connections (i.e., same as Table 2.5). The bandwidth scaling constants, β and γ, are both set to 0.05. The plot shows that all connection groups get very close to their mean max-min allocations, in contrast to the EPRCA scheme (Fig. 2.12). Furthermore, the mean and standard deviation of the queue sizes at the link are shown in Fig. 4.17 for both the EDERA and EPRCA schemes. The graph indicates that the queueing behaviour of the two schemes is vastly different. The EDERA queue length tends to decrease marginally for smaller duty cycles (i.e., due to more sporadic source behaviour), and is not overly sensitive to the mean duty cycle. Conversely, both the mean and standard deviation of the EPRCA queue length increase abruptly for smaller duty cycles. This trend is indicative of large

oscillations and poor link utilization.

To compare the throughput for bursty ABR sources more clearly, consider the number of cells received at the destinations for each connection group in Fig. 2.5. In particular, the results for a 20% duty cycle are shown in Fig. 4.18, EDERA, and Fig. 4.19, EPRCA (i.e., exponential on/off periods, mean on period of 2 ms, mean off period of 8 ms). From the first plot, the EDERA scheme gives very good steady-state throughput allocation to all connection groups. For example, the persistent connections 11-20 (4 ms roundtrip delays) and connections 21-30 (40 ms roundtrip delays) get near identical goodput, as evidenced by their slopes. This is not the case with EPRCA, where the longer delay connections are getting significantly more throughput. Now consider a given bursty ABR connection, one of connections 1-10. If this connection were persistent (i.e, non-bursty), then its ideal allocation would be 5 Mbps. However, since it has a mean duty cycle of only 20%, it should now receive roughly 1 Mbps throughput. Multiplying this value by ten for all such connections gives an aggregate usage of 10 Mbps, or roughly 23,585 cells/second. From the above figures, clearly the EDERA count is very close to this value (about 27,000 cells), whereas the EPRCA count is much lower (about 9,000 cells). Note that for the above tests, the means of the on and off periods are well below the *RM_timeout* value (100 ms). Further investigation is required to determine the effects of the timeout value on throughput performance with bursty ABR connections.

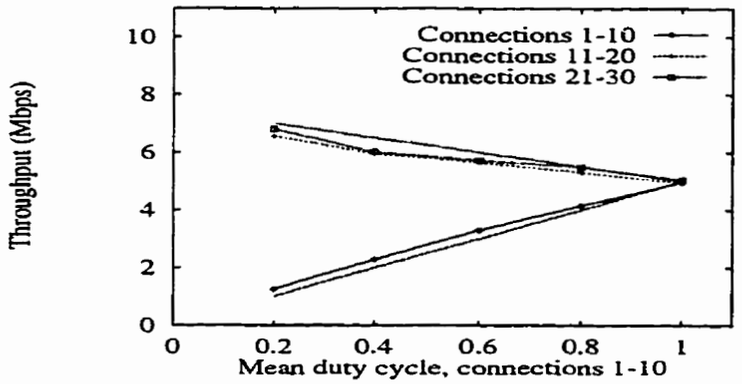


Figure 4.16: Throughputs in Fig. 2.5, non-persistent scenario

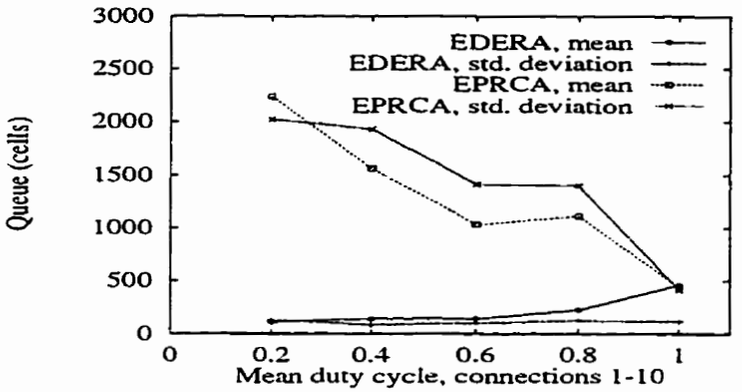


Figure 4.17: Link queue in Fig. 2.5, non-persistent scenario

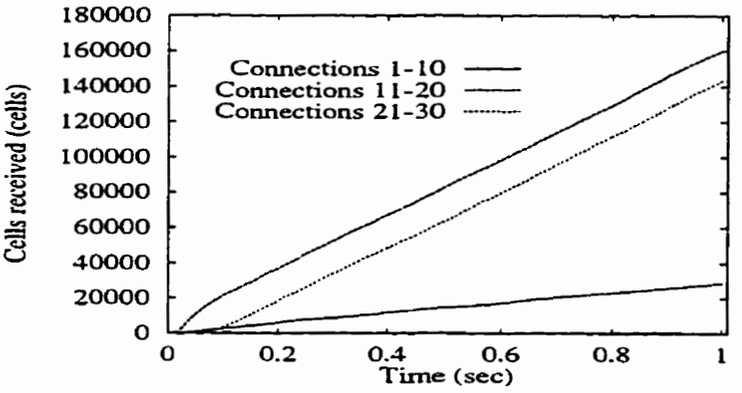


Figure 4.18: EDERA destinations in Fig. 2.5, non-persistent scenario

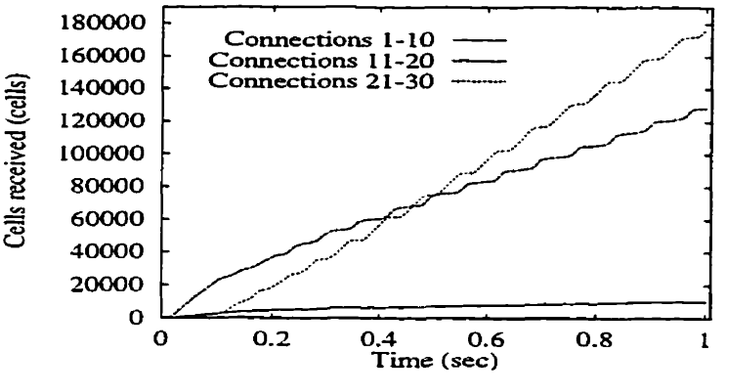


Figure 4.19: EPRCA destinations in Fig. 2.5, non-persistent scenario

4.2 Multiple Link Scenarios

To investigate the performance of the EDERA scheme in more realistic environments, multiple link scenarios are now tested. The experiments aim at stressing the queue control and bandwidth fairness properties of the algorithm for large-scale, dynamic networks. To gauge the autoconfigurability of the scheme, the bandwidth scaling constants are unchanged from their suggested settings of $\beta, \gamma = 0.05$. All results are averaged over two independent runs of one second duration each.

4.2.1 Bandwidth-Delay Scenario

First, consider the multiple link *parking-lot* scenario in Fig. 2.14, with differing connection requirements and idle VBR connections (i.e., Table 2.6). This scenario is tested with increasing link delays, and the mean steady-state throughputs for all connection groups are shown in Fig. 4.20 (horizontal lines representing the ideal MCR-plus-equal-share allocations in Table 2.6). The results show that the exact rate computations being done in the EDERA scheme give very precise (fair) band-

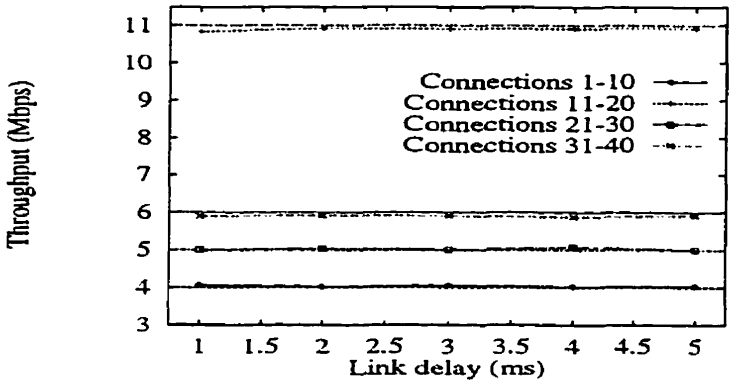


Figure 4.20: EDERA throughputs in in Fig. 2.14, idle VBR

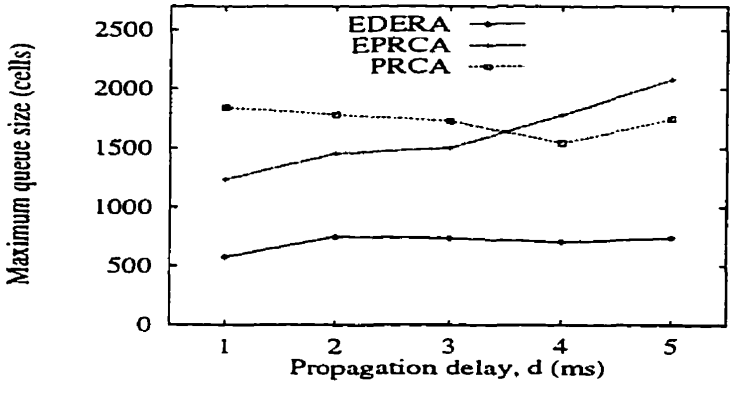


Figure 4.21: Maximum queue sizes at link 4-5 in Fig. 2.14, idle VBR

width allocation to all connection groups. There is no visible beat-down fairness problem (as in PRCA, Fig. 2.15) or rate clustering (as in EPRCA, Fig. 2.16). The corresponding maximum queue sizes at the most bottlenecked link, link 4-5, are also shown in Fig. 4.21. These buildups are much smaller than those with both the EPRCA and PRCA schemes, and are due to the improved congestion indicators.

4.2.2 Bursty VBR Scenario

The multiple link scenario of Fig. 2.14 is now tested with interfering, bursty VBR traffic (see Table 2.7). The scenario is identical to that in Section 2.2.3 (*Bursty VBR Scenario*), with each VBR source defined as on-off bursty and having a mean usage of 3 Mbps. The mean steady-state throughputs for each connection group in Fig. 2.14 are plotted in Fig. 4.22 (horizontal lines indicating mean fair share allocations, Table 2.7). Again, as in the idle VBR scenario, the throughput fairness is very good and does not degrade with increasing link delays. The bandwidth fairness is also significantly improved over that with the EPRCA scheme, i.e., compare with Fig. 2.18. For example, at 1 ms link delays, the throughput deviation for EDERA is only about 2.67% whereas it is 29.67% for EPRCA. Furthermore, the corresponding means and standard deviations of the (most-congested) link 4-5 queue length are graphed in Fig. 4.23. Although the mean values for both EDERA and EPRCA are fairly close, the standard deviation is much less with the EDERA scheme, confirming improved queueing performance. As an example, sample link 4-5 queue lengths are plotted for 2 ms link delays in Fig. 4.24, and verify much tighter control with the EDERA scheme.

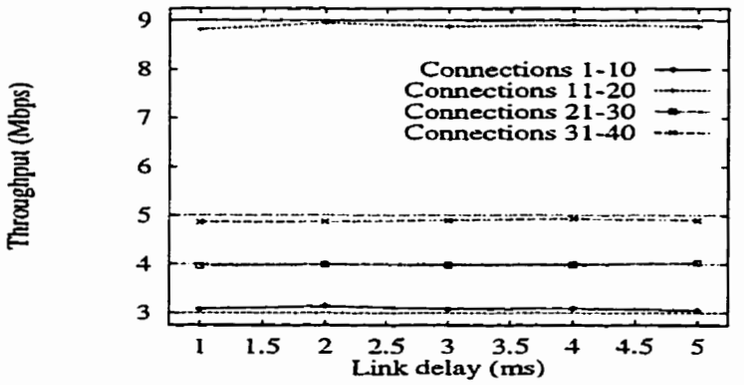


Figure 4.22: EDERA throughputs in Fig. 2.14, bursty VBR

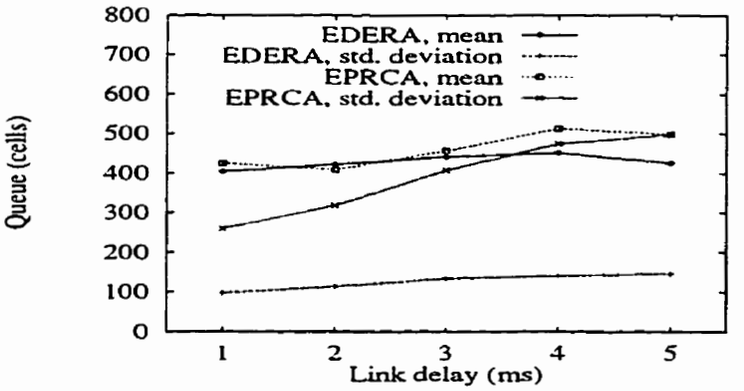


Figure 4.23: Link 4-5 queues in Fig. 2.14, bursty VBR

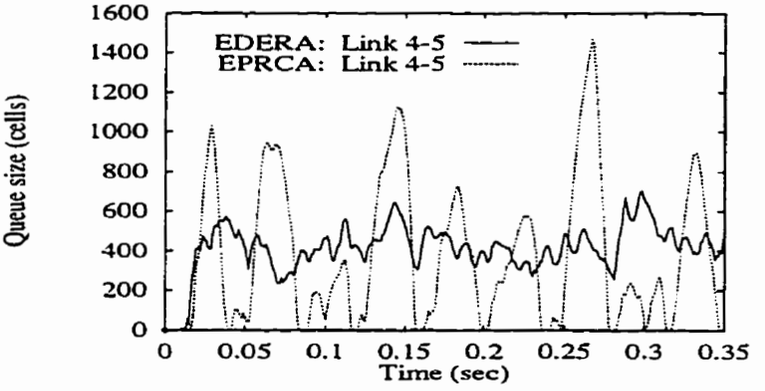


Figure 4.24: Sample link 4-5 queues in Fig. 2.14, 2 ms link delays, bursty VBR

4.2.3 Link Delay Scenario

Consider the multiple link (node) delay scenario presented in Section 2.2.3 (*Link Delay Scenario*). Similar to the EPRCA tests, the δ delay parameter in Fig. 2.20 (representing network link delays and end-system-to-switch delays) is varied from 100 μs to 5 ms. The mean steady-state EDERA throughputs are shown in Fig. 4.25 and indicate very little throughput deviation for LAN delays, i.e., less than 10% throughput deviation for $\delta \leq 100$ μs. However, as the delays increase, there is an increase in the throughput deviation, with the longer hop connections receiving roughly 14% below their mean fair share of 12 Mbps (Table 2.8), about 10.4 Mbps. This throughput beat-down occurs since the longer hop connections are bottlenecked at *all* links along their path, increasing their chances of being reduced. Nevertheless, with respect to the equivalent EPRCA throughputs shown in Fig. 2.21, the EDERA bandwidth fairness is still much higher (i.e., at 5ms, EDERA throughput deviation is 15.5% vs. 35.1% for EPRCA). This improvement is due to the exact rate computations being performed by the rate allocation step.

The mean and standard deviation of the link 1-2 (Fig. 2.20) queue size are plotted in Fig. 4.26 (along with the EPRCA results, Fig. 2.22). From the figure, it is noted that both quantities approximately double between $\delta = 100$ μs and $\delta = 5$ ms. However, these increases are much smaller than those manifested in the EPRCA scheme. For example, sample link 1-2 queue lengths are plotted in Fig. 4.27 for $\delta = 5$ ms. The EDERA scheme has much smaller and more tightly controlled *steady-state* queue lengths, confirming improved queue control. These results are very good, considering that there is no source-rate decay mechanism in

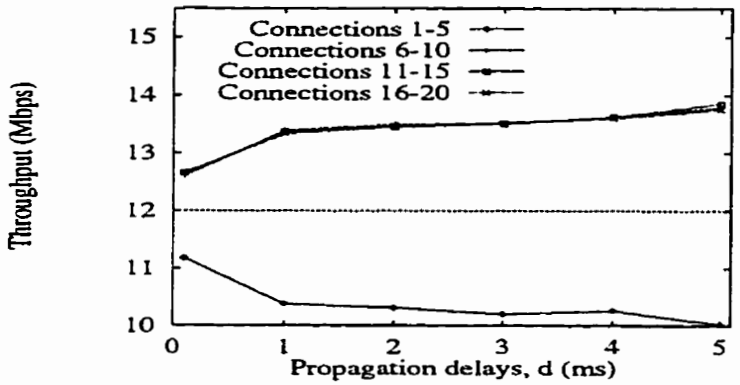


Figure 4.25: EDERA throughputs in Fig. 2.20 ($\beta, \gamma = 0.05$)

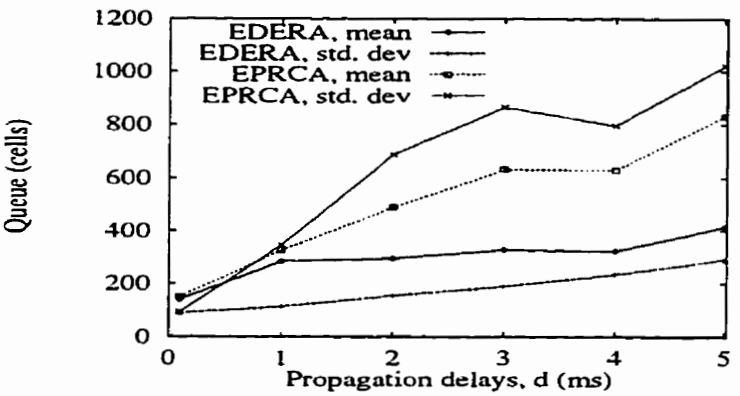


Figure 4.26: Link 1-2 queues in Fig. 2.20

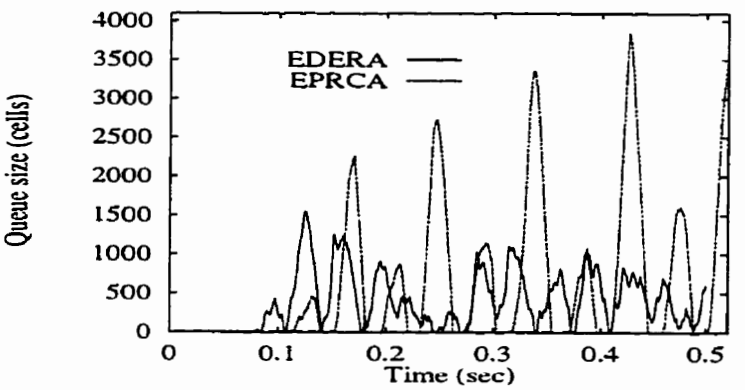


Figure 4.27: Link 1-2 queues in Fig. 2.20, $\delta = 5$ ms

the EDERA scheme, as there is in EPRCA. Furthermore, no real parameter tuning is done either, indicating good autoconfigurability of the suggested parameter set.

4.2.4 Non-Oscillatory Condition

The non-oscillatory requirement for the bandwidth scaling parameters is further tested for the multiple node network in Fig. 2.14. Since constant link capacities are assumed (Section 3.3.2), the condition can only be verified for idle VBR connections (i.e., Table 2.6). To stress the scheme, the link propagation delays are set to large values of 5 ms each. Choosing $\rho = 0.90$ (different from Table 4.1) and computing Eq. (3.17), β^* is evaluated as 0.073 for this network, i.e., at link 1-2 for connections 11-20. The resulting ACR behaviour for connections from each group is shown in Fig. 4.28, for $\beta = 0.05 < \beta^*$. The plot confirms non-oscillatory operation and proper MCR-plus-equal-share rate allocation. The link 4-5 queue buildups are also shown in Fig. 4.29, alongside those with oscillatory parameter settings ($\beta, \gamma = 0.05$), and reveal similarities with the single link scenario in Section 4.1.4. The queue size at the most bottlenecked link (link 4-5) falls below the QT threshold of 200 cells at roughly 170 ms, after which the traversing connections ramp up for the remaining unused bandwidth of $\beta^*\mu$ cells/second (see connections 1, 21, 31, Fig. 4.28). Since connections 11-20 are bottlenecked elsewhere (link 1-2), they must relinquish some bandwidth to the other connections, connections 1-10, after congestion abates at link 4-5. This is evidenced by the slight reduction in the connection 11 ACR at about 195 ms in Fig. 4.28. The initial transient phase is longer and the maximum queue buildups are larger for the non-oscillatory settings. However, in light of

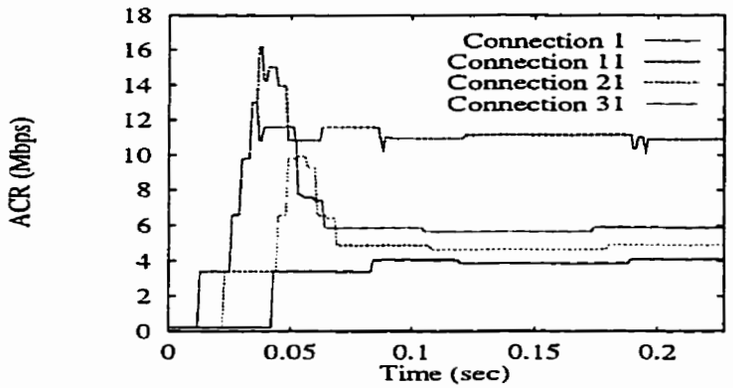


Figure 4.28: Sample connection ACR values, Fig. 2.14 ($\beta = 0.05$, $\gamma = 0$)

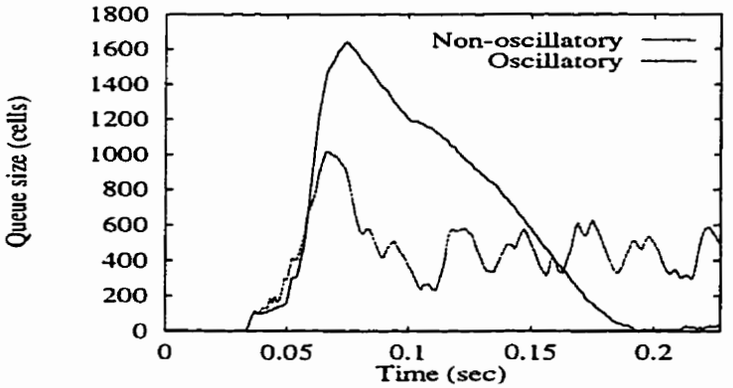


Figure 4.29: Link 4-5 queue sizes in Fig. 2.14

the scale of the network being tested (connection sets, propagation delays), these buildups are still very reasonable. Tests with the multiple link network in Fig. 2.20 also show similar overall behaviour for non-oscillatory parameter sets (i.e., $\beta^* = (1 - \rho)$ since there are no MCR guarantees or non-bottlenecked connections at any of the links).

It should be mentioned that the non-oscillatory condition is derived based upon very tight assumptions on the operating point of the scheme (see Appendix E). The derivation assumes that the network is close to the (MCR-plus-equal-share) optimal operating point at all links, thereby approximating the actual connection rates with their ideal values. However, as is seen above, the condition still holds in many practical settings. In fact, further simulation results indicate that Eq. (3.17) is actually a very stringent condition. In practice, non-oscillatory behaviour can also occur for non-zero γ values in various networks. Specifically, if the transient queue buildups are not large enough to force (any) transition into the non-bottlenecked state upon congestion abatement, rate oscillation will be avoided.

4.3 Summary

The EDERA scheme, tested for a wide range of network scenarios, shows very good overall performance. EDERA outperforms EPRCA for many of the test cases tried, both with respect to queue control, link utilization, and bandwidth fairness. Furthermore, EDERA shows good scalability over a broad range of operating environments, maintaining high performance with a given (suggested) set of control parameters. Sensitivity results also indicate that the scheme functions best with

small bandwidth scaling constants. Larger settings degrade the throughput fairness and cause increased queue length oscillations. The non-oscillatory condition stated in Chapter 3 is also verified for several test networks. In general, it is observed that the resulting transient times and queue buildups increase significantly with this feature.

Chapter 5

Conclusions and Future Work

As a packet-switching technique, ATM has been proposed as the enabling technology for future high speed networks. The desire to support a diverse range of traffic types with this common network infrastructure has led to the development of several user traffic classes. Crucial to the widescale acceptance of this new technology is the definition and services support for a data traffic class. In general, data traffic is highly unpredictable and usually more delay tolerant than other forms of traffic. In light of these requirements, the governing standards bodies have defined the ABR service class to maximize network bandwidth utilization. Because the ABR service class does not have delay specifications, ABR service management admits feedback flow control to modulate data source transmission behaviour. The design of robust ABR schemes, capable of functioning in a wide range of network conditions, is therefore essential.

Although various ABR flow control proposals have appeared in the literature, many important issues remain outstanding. Many of the existing algorithms lack

clear extensions to handle MCR guarantees amongst connections. Others give sluggish behaviour which are overly sensitive to the tuning of various tracking parameters (e.g., EPRCA, APRC), and hence do not scale well to different network scenarios. Although faster, more exact rate allocation algorithms have been proposed (MIT-scheme, ERAA, ERICA), necessary provisions for functionality in dynamic environments are missing. These include features for handling transient queue buildups, bursty sources, and varying background non-ABR traffic levels.

From the outset, the main goal of the work has been to develop an improved, comprehensive strategy for ABR services support in ATM networks. It is desired to adequately address the previous shortcomings with other (end-to-end) flow control schemes and comply with the stipulated standards guidelines. Furthermore, rigorous performance evaluation is required to fully test (and improve) the proposed rate allocation algorithm.

5.1 Conclusions

The *enhanced distributed explicit rate allocation* (EDERA) scheme is introduced to address deficiencies in the existing ABR flow control proposals. The heart of the scheme is a distributed algorithm running at both the end-systems and network switches, which adheres to the ATM Forum guidelines. To enable functioning in more general network settings, additional capacity tracking and congestion detection features are also included. The main rate allocation procedure employs exact rate computations to improve the bandwidth fairness performance, not relying on (approximate) tuning factors to force convergence to fair operation. In

addition, the MCR-plus-equal-share fairness criterion is implemented to properly handle bandwidth partitioning beyond the MCR guarantees. Further extensions to other MCR-related criterion (as suggested in the ATM Forum [1]) are also straightforward. Transient effects are countered using advanced capacity tracking and congestion detection algorithms which closely follow changes in available capacity levels. Specifically, both rate and queue information is used to derive more responsive congestion indicators and modulate usable capacity levels. Source end-systems also measure their usages to improve bandwidth utilization with bursty sources. The scheme has a moderate control parameter count and is of acceptable implementation complexity.

Extensive performance evaluation results (via simulation) show that the EDERA scheme performs very well in many diverse network scenarios. The scheme achieves very high link utilizations along with good, tight queue control. In addition, the bandwidth proportioning occurs in a predictable manner, closely following the chosen fairness criterion in most cases. The scheme is shown to outperform the well-known EPRCA proposal for almost all network scenarios tested. For constant link capacities and more idealized network settings, EDERA can also yield non-oscillatory steady-state performance. Simulation results verify this feature in various test cases, albeit with corresponding larger transient phases.

Since ATM network switches are integrated transport platforms, they must serve the requirements of all traffic classes. Proper resource allocation between the traffic types has to be done by the switches to prevent misbehaving traffic flows from undermining the network's performance. To implement this partitioning, a

hierarchical scheduling strategy, *hierarchical fair queueing* (HFQ), is proposed for ABR and VBR integration at the cell level (see Appendix B). The scheme uses fair queueing techniques and, in general, gives real-time traffic increased priority for link access. However, firewall mechanisms are provided to prevent excessive real-time traffic loadings from degrading the guaranteed portion of the throughput requirements of the ABR flows. Simulation findings indicate that the scheme is significantly better than a strictly VBR priority approach, especially at heavier loads. Most notably, the operating ABR queue sizes are smaller. In addition, HFQ also offers better delay performance for real-time traffic compared to a simpler, non-hierarchical integration strategy.

5.2 Future Work

In this thesis, a comprehensive strategy for ABR services support in ATM networks is presented and evaluated by means of extensive simulation and some analysis. However, there are various important issues relating to the proposed algorithm which deserve further investigation.

The robustness and accuracy of the capacity tracking mechanism needs to be further evaluated with more realistic input traffic patterns. Possibilities include the use of genuine non-data flows, such as video traces or self-similar streams. Also, the effect of capacity fluctuations of relatively shorter time scales (i.e., less than roundtrip delays) needs to be investigated. Although feedback schemes are designed with the assumption that transients occur on relatively large time scales, in practice, this may not be the case. These results can be used to further improve the tracking

mechanism. Closely related to the performance of the capacity tracking mechanism, is the effect of the T_{sw} measurement interval at network switches. This value should depend upon the exact nature of the background non-ABR traffic, if meaningful capacity estimates are desired. In particular, it is felt that T_{sw} should be adjusted (on a longer time frame) using measures related to the capacity tracking phase (i.e., such as [change in] variance of tracked capacity). Along the same lines, adaptive utilization thresholds (i.e., $\rho(t)$ versus ρ, Section 3.3) can also be tested. During periods of high congestion (rate or queue), $\rho(t)$ can be lowered appropriately to reduce the amount of transient overallocation being done.

It may be argued that the division operation required in the rate allocation phase (on a per RM cell basis) is too computationally expensive. Hence, in order to reduce implementation complexities, simplifications to the rate computation phase can be proposed. These include using coarser look-up tables and/or performing the calculations less frequently (i.e., every T_{sw} measurement interval as opposed to bidirectionally for each RM cell). The effects of such approximations on the convergence and steady-state behaviour of the scheme need further attention.

Analytical modelling of the queueing behaviour of the scheme is another important area. This will allow network service providers to properly size link buffers and tune control parameters. Although some initial work has been presented to bound queue buildups for solely the rate allocation algorithm (Appendices F, G), more realistic results are desired. Specifically, the reactive congestion detection mechanism must be modelled along with more realistic additive source rate increments.

The source end-system algorithm can have an important effect on the overall

bandwidth efficiency of the scheme. The current literature contains many additional, conservative features for improving the cell loss performance of ABR connections [24]. The impacts of such functionalities on the bandwidth distribution and queue control performance of the EDERA scheme should be investigated. In addition, extensions to the simple segmental averaging scheme can also be tested to improve bandwidth utilization with bursty sources (i.e., more advanced source usage tracking).

Supporting multicast ABR connections is another topic which needs further attention [16]. Multicast transmission is an important feature of current data networks which must be "ported" to ATM networks for migratory purposes. Furthermore, the interaction of the scheme with higher transport layer protocols (such as TCP/IP, NETBLT, etc.) can affect the throughput fairness seen by end-level users. These effects must be carefully examined and well understood to determine how effective the scheme is as an overall data transport mechanism.

With regards to ABR and VBR integration at the cell level, some initial work has been done with the proposed HFQ scheduling strategy (Appendix B). Overall, simulation findings show significant performance improvements over some other integration strategies, especially at heavier VBR loads. In this area, further work is necessary to extend the integration approach to cover other types of schedulers (i.e., such as round-robin, earliest-due-date, etc.). Pertinent implementation issues also require detailed investigation.

Appendix A

ATM Forum ABR Schemes

Many rate-based control schemes have been tabled in the ATM Forum and some details on the major algorithms are presented here to compliment the overviews in Chapter 2. Interested readers should also check the respective references for full descriptions of the various schemes.

A.1 BECN

Backward explicit congestion notification (BECN) is a very simple technique in which switches directly signal congestion to sources via backward RM cells. Because notification is sent directly from the point of congestion to the source, fast responses are achieved. The congestion indicator is based on a single queue threshold. If a source receives a BECN RM cell, it reduces its transmission rate by half (down to MCR). If no RM cells are received within a certain "recovery" time period, a connection can double its rate (up to PCR). Filtering at the network switches is

suggested to prevent excessive BECN cells being sent to a source. There is no point in sending additional BECN cells before the previous feedback has reached the source. The above feedback mechanism is classified as negative-feedback [12], since RM cells are sent to decrease rates. This technique is risky in case of RM cell losses or delays. To improve fairness Newman suggests that the recovery periods be proportional to the transmission rates. This allows slower rate connections to speed up faster. BECN is a non-selective technique and requires minimal switch complexity. Further details are presented in references [9],[27].

A.2 FECN

Forward explicit congestion notification (FECN) is an end-to-end timer-based scheme using bit marking [32]. An ATM cell has a reserved *explicit forward congestion indication* (EFCI) bit-field which can be set by switches in case of congestion (similar to the earlier equivalent window-based scheme [20]). Switches use queue levels to determine congestion and set the EFCI bits in passing data cells when congested. The destination uses an averaging interval, RMI, to monitor the EFCI bits of incoming cells. At the end of the RMI interval, if no EFCI bits are set, an increase RM cell is sent to the source. The increase allows the source to increase its rate by a fixed additive amount. Similarly, the source has an observation interval, UI, which is used to decrease rates. Sources decrease their rates multiplicatively if no RM cells are received within a UI interval. This is a positive feedback strategy and is generally safer than that used in the BECN scheme. Note that destinations should also perform some filtering on the EFCI bits to prevent an excessive amount

of RM cells being sent to a source. Both BECN and FECN use timing intervals to update rates, be it at the source or destination.

A.3 PRCA

Similar to FECN, the *proportional rate control algorithm* (PRCA) [12] is an end-to-end technique using counters to send RM control cells as opposed to timing intervals. Here a source sends an RM cell after every block of $(N_{rm} - 1)$ data cells, and N_{rm} values ranging from 20 to 32 have been used in LAN simulation studies [30]. This mechanism ensures that the maximum RM cell rate is below $\frac{1}{N_{rm}}$-th of the source rate. Sources keep reducing their rates multiplicatively after each cell transmission time (not necessarily after each cell transmission):

$$ACR = \max(MCR, \min(ACR(1 - \frac{1}{MDF}), PCR)), \tag{A.1}$$

where MDF (*multiplicative decrease factor*) is a relative decrease factor, roughly 16. Switches operate as in FECN, setting EFCI bits in data cells during congested states. Destination end-systems send backward increase RM cells if the last received EFCI status is cleared, indicating no congestion. Increase RM cells at the source allow for a rate increase to compensate for the decreases made since the last increase plus a constant amount:

$$ACR = \max(MCR, \min(PCR, (ACR + N_{rm} \cdot (ADR + AIR))), \tag{A.2}$$

where AIR is a fixed increment, usually a fraction of the PCR, and the ADR (*additive decrement rate*) is a relative decrement:

$$ADR = \frac{ACR}{MDF}. \tag{A.3}$$

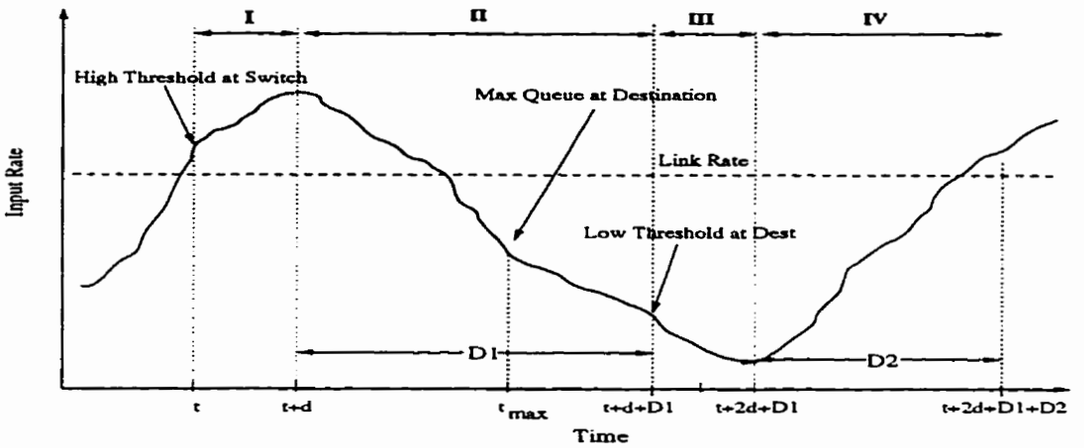
By also allowing switches to remove backward increase RM cells during congestion, a faster BECN-like operation can be achieved. The continual rate decrease policy is a *use-it-or-lose-it* philosophy [12], and prevents an idle high bandwidth source from coming online and causing overload. It also reduces traffic during congestion if feedback cells are delayed. Chiu has shown analytically that additive increase/multiplicative decrease is the best linear policy for bandwidth fairness for the single hop, zero-MCR case [66]. Using this, the switch should converge to equal proportioning among zero-MCR connections.

As previously mentioned, the congestion indicators in BECN, FECN, and PRCA are of the queue-threshold type. Adjusting the threshold values allows throughput to be traded for delay. Thresholds control the mean queue length and thus by Little's Law the mean cell delay. PRCA has a refinement over the single threshold mechanism used in BECN and FECN. This is a hysterisis mechanism with two thresholds to detect congestion onset and abatement (i.e., high and low, respectively). A congested state is entered if the queue length crosses over the high threshold and the previous state is uncongested. Similarly, the uncongested state is entered if the previous state is congested and the queue length falls below the low threshold. Figs. A.1 and A.2 show in detail the source rate and link queue behaviour using the hysterisis mechanism (single source at roundtrip propagation

delay of δ). The illustrations are general, not depicting any specific control scheme, but the overall behaviour is similar. In region I, Fig. A.1, the source increases its rate while the network switch is congested. In region III the source decreases its rate while the switch is non-congested. This graphically depicts the feedback delay problem. The region II and IV are associated with the queue reduction and increase times between the two thresholds. Hysterisis thresholding prevents oscillatory congestion indication when the queue length varies about the high threshold. This filtering reduces the amount of RM cells generated, reducing overhead. The hysterisis mechanism usually results in steady-state rate oscillations about the ideal fair share values. The magnitude and frequency of these oscillations is in part controlled by the difference between the high and low queue thresholds at the switches. If these values are close then the oscillations will be small, but the frequency will be large. The converse holds for larger differences. Various authors have mathematically show this relationship for relatively simple schemes [69],[71],[72].

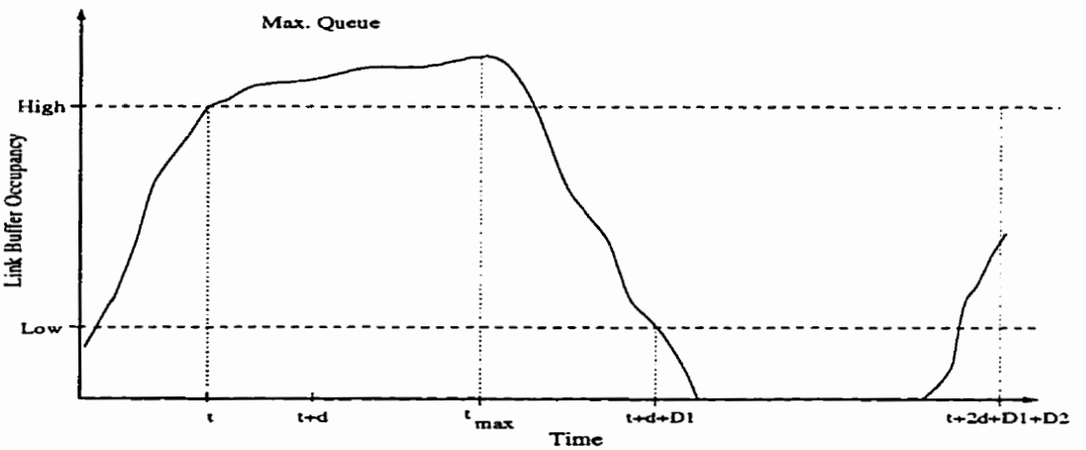
A.4 EBCI

The *explicit backward congestion indication* (EBCI) scheme and the *enhanced proportional rate control algorithm* (EPRCA) are two spinoffs of the basic PRCA scheme. In EBCI the congestion indicator is the same hysterisis queue threshold mechanism of PRCA. Similar to PRCA, a source continually decays its ACR's after every cell transmission. In addition the source emits *forward* RM cells after each fixed group of $(N_{rm} - 1)$ data cells. These forward cells contain stamped values of the source's current cell rate (ACR) and its desired explicit rate, ER (set



t	Congestion detection at switch
t+d	Rate-decrease message arrives at source
t+d+D1	Congestion abatement at switch
t+2d+D1	Rate-Increase message arrives at source
D1	Queue empty time (High-Low)
D2	Queue rise time to High threshold

Figure A.1: Source rate behaviour using hysterisis



t	Congestion detection
t+d	Rate-decrease message arrives at source
t+d+D1	Congestion abatement
t+2d+D1	Rate-Increase message arrives at source
D1	Queue empty time (High-Low)
D2	Queue rise time, minimum to High

Figure A.2: Switch queue occupancy using hysterisis

to PCR). A received RM cell at the destination is echoed back towards the source end. On the forward pass the switches use the ACR and ER values in the RM cells to adjust an exponentially-weighted running average, the *mean allowed cell rate* (MACR). If the switch is uncongested then the mean ACR is adjusted as:

$$MACR = (1 - AV) \cdot MACR + AV \cdot ACR_{cell}, \tag{A.4}$$

where AV is an averaging factor and ACR_{cell} is the rate in the cell [13]. If the switch is congested but the ACR of the cell is *lower* than MACR, then Eq. (A.4) is also applied. Otherwise the MACR is unchanged. This is a heuristic idea, updating in favour of all sources during underload and only underallocated ones during overload.

On the reverse path, the switches compare the stamped ACR to the MACR, and mark the EFCI bit if the ACR exceeds a large fraction (DPF, *down pressure factor*) of the switch's MACR. If a source receives an RM cell with the EFCI-bit set, it does not increase its rate. If the EFCI-bit is clear, then the PRCA update of Eq. (A.2) is used. This is called "selective marking", and in congested states favours connections with lower ACR's. Another enhancement to the scheme is the addition of a second queue threshold to detect extreme congestion, DQT. DQT is set well above the regular threshold, and when exceeded, *all* EFCI-bits in backward RM cells are marked. This reasoning ignores fairness during emergency situations, since bounding cell loss and preventing congestion collapse are more important concerns. Ohsaki et. al also denote EBCI switches as *binary enhanced switches* (BES) [13]. EBCI has per-connection computational overhead but no per-connection storage requirements, since the RM cells contain all required information

to adjust connection rates. BES switch complexity is not high and bandwidth fairness is improved. Nevertheless, responsiveness remains a problem since explicit-rates are not used.

A.5 EPRCA

EPRCA is a further enhancement of EBCI [28],[29]. As in EBCI, sources send out intermittent RM cells with stamped ACR and PCR fields, and network switches use Eq. (A.4) to keep track of the mean ACR, MACR. Destination end-systems perform RM cell echoing, and the major differences over EBCI arise for switch and source end-system behaviour for the backward pass of RM cells. If the switch is uncongested then the RM cell is passed without any changes. An issue only arises during congestion. If a switch is congested and the stamped ACR of a reverse RM cell is greater than a large fraction of the MACR, then the PCR field of the cell is reduced:

$$ER_{cell} = \min(ER_{cell}, ERF \cdot MACR), \tag{A.5}$$

where ER_{cell} is the ER-field in the RM cell and ERF (*explicit reduction factor*) is a large fraction, about $\frac{15}{16}$. Otherwise the cell is passed downstream without any changes. Like EBCI, extreme congestion is also checked using a very high threshold, DQT. If the switch is found to be extremely congested, then all connection rates

are reduced, bypassing Eq. (A.5):

$$ER_{cell} = \min(ER_{cell}, MRF \cdot MACR), \tag{A.6}$$

where MRF (*major reduction factor*) is typically around $\frac{1}{4}$. Sources receive the RM cells and update their ACR's using:

$$ACR = \max(MCR, \min((ACR + N_{rm} \cdot (AIR + ADR), ER_{cell}, PCR))), \tag{A.7}$$

where AIR and ADR are identical to the PRCA scheme. Limiting the increase to the maximum allowed by PRCA (Eq. (A.2)) preserves backwards compatibility and prevents sources from directly ramping up to their PCR in uncongested states. EPRCA switches are also called *explicit down switches* (EDS) [13]. More details on the scheme can be found in [28],[31]. Some analytical modelling of EPRCA has been done using fluid flow models for *persistent* sources [13],[29]. Note that some simulation results for EPRCA are presented in Chapter 2.

A.6 APRC

Siu also presents a modified version of EPRCA, *adaptive proportional rate control* (APRC), which addresses the "extreme-congestion" unfairness problem with the former scheme [13]. APRC performs rate adjustments in an identical manner, but only differs in the regular congestion indicator [32]. Instead of queue thresholds, congestion is now indicated by evaluating the change in queue length over a time

interval. Specifically, if the queue length increases over N cell times then congestion is indicated. The higher (extreme) queue threshold is still kept to handle boundary cases. Responsiveness is dramatically improved and simulations show that resultant queue lengths are also much smaller than those in the original EPRCA [32].

A further advancement of APRC, APRC2, is also presented by Siu [31]. This is done by separating all ABR connections into two groups: those which are bottlenecked at a given switch and those which are bottlenecked at *other* switches. Typically the ACR's of one of these groups is less than the average ACR, MACR, of the switch. The scheme introduces another mean ACR parameter, the UCR, which is basically a running average similar to MACR but only for connections whose rates are above MACR:

$$UCR = (1 - \alpha) \cdot UCR + \alpha \cdot ACR_{cell}, \tag{A.8}$$

for appropriate α. The author claims improved fairness and ramp-up times with the new scheme [13].

A.7 MIT Scheme

Charny has proposed an end-to-end rate allocation scheme called the MIT scheme [36]. Although this scheme was developed before the definition of the ABR service class, it still holds a lot of relevance and can easily be cast into the ABR framework. Sources emit special control cells containing two values, a "reduced" bit and an estimate of their rate/desired rate, the stamped rate. The switches meanwhile

maintain a value called advertised rate which indicates the capacity allowed per connection [36]. On the forward pass all switches on the path compare the stamped rate to their advertised rate. If the former is larger, the stamped rate is reduced to the advertized rate and the reduced bit is set, similar to the EFCI bit in PRCA. Otherwise, the control cell passes the switch unchanged. Hence on arrival at the destination, the control cell contains the minimum of the source rate and allowable rates of all switches along its path. The destination echoes the control cells back to the source which adjusts its rate if the reduced bit is set. A clear reduced bit means that no link along the path is bottlenecked and the source can increase its rate *directly* to its desired rate. So basically this scheme works by a source requesting a rate and the network adjusting it to the maximum allowable. If there is a single bottleneck switch for all connections then one roundtrip delay will allow all sources to adjust to their fair rates. If there are multiple bottlenecks, then more roundtrips are required, since capacity given up by bottlenecks with less spare capacity in the early roundtrips can be used in subsequent roundtrips by bottlenecks with more spare capacity.

The focal point of this scheme is computing the advertized rate, the fair share, between competing connections. Switches must keep a list of all connections traversing a given link and their latest stamped rates. Connections are grouped into two groups, overloaded and underloaded, based on whether their stamped rates are above or below the advertized rate. The advertized rate is computed as:

$$\text{Advertized Rate} = \frac{\text{Link Capacity} - \sum_{i \in Underloaded} \text{Rate}_i}{\text{Total Number of Connections} - \text{Number of Underloaded}}. \tag{A.9}$$

Eq. (A.9) assumes that underloaded sources are bottlenecked elsewhere, and hence their capacity can be allocated to others (i.e., max-min principle [3]). Note that the advertized rate is a running parameter and has to be updated on each RM cell arrival. If the stamped rate in a control cell is less than the advertized rate, then Eq. (A.9) is evaluated to update the advertized rate. Since this new value may cause old groupings to change, the advertized rate has to be recomputed for the latest groupings. It can be shown that the second computation is sufficient to ensure that the groupings will no longer change [36]. This is an $O(n)$ procedure, with n being the number of connections using the link, and the resulting switch complexity is significant. The MIT schemes gives oscillation-free performance in steady-state and very quick response times (as low as one roundtrip delay). However, this scheme has no provisions to curtail transient queue buildups or handle MCR requirements.

A.8 OSU and ERICA Schemes

Jain et al. have presented two schemes, the OSU-scheme and *explicit rate indication for congestion avoidance* (ERICA) [43],[44]. In the earlier OSU-scheme (also called EPRCA+ in [13]), both sources and switches are required to measure their input rates over given measurement intervals. Specifically, sources measure their input rates over an interval, *offered cell rate* (OCR), and also measure the instantaneous cell rate, *transmitted cell rate* (TCR), as the inverse of the latest inter-cell time. RM cells are sent out at the end of each interval and contain the maximum of the OCR and TCR, denoted *current cell rate* (CCR). In addition, an *adjustment factor* field is also defined for RM cells, and this is set to unity on forward emission.

Meanwhile at the switches a load factor, z:

$$z = \frac{\text{Measured Input Rate}}{\text{Target Rate}} = \frac{\text{Cells in Interval}}{\text{Interval Time} \cdot \text{Utilization} \cdot \text{Link Capacity}}, \tag{A.10}$$

and fair share value:

$$\text{Fair Share} = \frac{\text{Target Rate}}{\text{Number of Active Sources}}, \tag{A.11}$$

are computed based upon the desired link utilization and measured input rate [44]. z is the congestion indicator and ideally should be unity. In essence, Eq. (A.10) measures change in queue length and this can give earlier warning of congestion as opposed to queue length alone. Utilization values of about 0.95 are recommended [44]. On the forward pass of an RM cell, network switches just store the CCR value to have a latest estimate of source activity. Also, if the cell is the first from a given connection within the switch interval, the source is marked as active (for later use in Eq. (A.11)). This is per-VC accounting. Destinations simply echo RM cells back and rate adjustment is done by switches on the backward pass. A switch first checks to see if the load factor is within a desired *target utilization band* (TUB) about unity, $[1-\Delta, 1+\Delta]$. If it is not, the adjustment factor field in the RM cell, z_{cell}, is set to the maximum of the load factor in the cell and the switch's load factor, regardless of the cell's CCR:

$$z_{\text{cell}} = \max(z_{\text{cell}}, z). \tag{A.12}$$

Otherwise the measured utilization is within the TUB and the adjustment factor is set contingent on the cell's CCR:

$$z_{\text{cell}} = \begin{cases} \max(z_{\text{cell}}, \frac{z}{1+\Delta}) & \text{if } \text{CCR}_{\text{cell}} < \text{Fair Share} \\ \max(z_{\text{cell}}, \frac{z}{1-\Delta}) & \text{if } \text{CCR}_{\text{cell}} > \text{Fair Share.} \end{cases} \tag{A.13}$$

The goal is to get the *maximum* load factor along the connection's path. Sources adjust their rates based on arriving RM cells:

$$\text{ACR} = \frac{\text{CCR}_{\text{cell}}}{z_{\text{cell}}}. \tag{A.14}$$

During extreme overload ($z \gg 1$) the scheme does not view fairness as a major concern, and a congested switch's overload factor will override those of other path switches by virtue of Eq. (A.12). When all path switches are within the TUB, then fairness is ensured via Eq. (A.13). This mechanism is similar to the dual queue thresholds used in the EBCI and EPRCA schemes.

ERICA is similar to the OSU-scheme and differs mainly in the switch operation for backward RM cells. The TUB is no longer used in ERICA. Instead the ER field of RM cells is modified by network switches as:

$$\text{ER}_{\text{cell}} = \min(\text{ER}_{\text{cell}}, \max(\text{Fair Share}, \frac{\text{CCR}_{\text{cell}}}{z})), \tag{A.15}$$

where the fair share and z are given by Eq. (A.11) and Eq. (A.10), respectively. Sources just set their rates to the ER values in returning RM cells. The logic is very simple, and works by determining the minimum allowable rate of all path

switches. By virtue of Eq. (A.15), during extreme congestion ($z \gg 1$) a connection is still guaranteed its fair share at congested network switches. During extreme underload ($z \ll 1$), a connection can be allocated more than its fair share. This allows for very fast ramp-up/down times and is a very elegant mechanism. This is not the case in the original OSU-scheme (and even EPRCA), where fairness is ignored during extreme congestion. In addition, recently, there have been many notable additions and extensions to the ERICA scheme in order to eliminate some inherent shortcomings (see [46] for complete details). However, the basic issue of MCR support is still not addressed in this enhanced proposal.

Appendix B

Scheduling Issues

Data services in ATM networks must share link bandwidth with real-time services. As such, provisions for satisfying the QoS requirements of both traffic types must be provided. Since non-ABR traffic can influence the performance of ABR flow control schemes, the coexistence of ABR flow control algorithms with real-time scheduling schemes [83] requires further investigation. Most studies on real-time traffic scheduling [83]-[92] have not explicitly addressed the requirements of the ABR service class (MCR guarantees, cell loss). Rather, the focus has been on maintaining delay and throughput requirements for real-time flows, assuming that data traffic is merely buffered and served only during real-time traffic idle periods [83]. In fact, this is the very approach adopted in the simulation studies in the previous chapters, Chapter 2 and Chapter 4, where CBR/VBR cells are separately buffered and given non-preemptive service priority over ABR data cells.

However, under heavy load conditions, if the throughput guarantees for ABR flows are not enforced, there can be significant queue buildups (and cells losses)

for data traffic. Here, it is important to properly integrate the proposed cell-level scheduling algorithms [83] with ABR flow control schemes in order to ensure that the requirements of all traffic types are met. An overall hierarchical scheduling strategy for handling real-time and data traffic in ATM networks is now proposed. The design uses known rate-based *fair queueing* [94] algorithms to give (approximate) fair sharing of idle bandwidth. The hierarchical link sharing concept is first introduced followed by a detailed design description. Some simulation results are also presented to evaluate the performance of the strategy.

B.1 Hierarchical Link Sharing

Consider an output port of an output-buffered switch supporting two classes of traffic with service capacity μ cells/second, as shown in Fig. B.1. The first traffic class is comprised of N_{VBR} real-time connections (either CBR or VBR) with individual rate guarantees given by ω_i cells/sec. This value is determined by the CAC function, and usually represents the *sustainable cell rate* (SCR) [1] of the connection. The second traffic class is comprised of N_{ABR} ABR (data) connections with individual MCR [1],[2] guarantees of ψ_i cells/sec. Note that Fig. B.1 does not make any assumption on the exact buffer sharing strategy (i.e., complete sharing, complete partition, pushout etc.). To ensure stable queueing behaviour at the link, it is assumed that the following rate inequality is satisfied:

$$\sum_{i=1}^{N_{VBR}} \omega_i + \sum_{i=1}^{N_{ABR}} \psi_i \leq \mu, \tag{B.1}$$

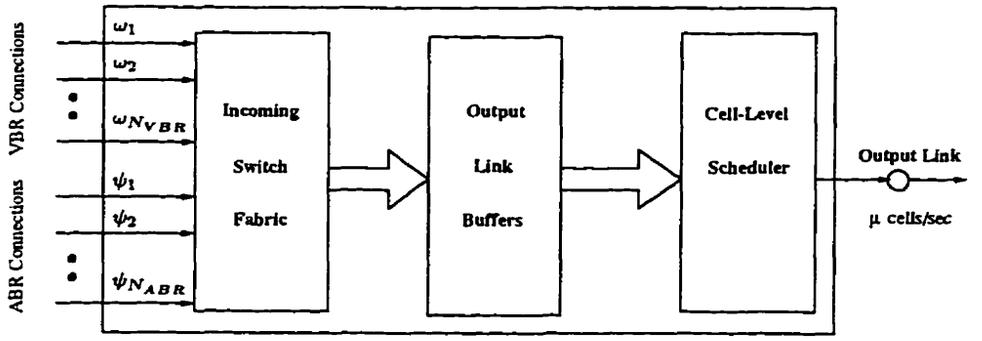


Figure B.1: Overview of an output-buffered network switch

(i.e., no capacity over-booking). Hence, the ABR connections require a minimal, *non-zero* bandwidth commitment, Ψ cells/second, equal to the aggregate MCR requirement of all ABR connections:

$$\Psi = \sum_{i=1}^{N_{ABR}} \psi_i. \tag{B.2}$$

The *actual* service rate extended to ABR traffic, Θ cells/second, however, can be greater than minimum aggregate guarantee, i.e., $\Theta \geq \Psi$. Nevertheless, determining an appropriate setting for Θ is more of a network services provisioning issue and is not considered further here.

First, consider a *work-conserving* server serving both the real-time and data traffic classes in an an integrated services environment. Conceptually this can be represented as a *one-level* (flat) service tree using the link sharing approach presented in [81], see Fig. B.2. Each leaf node in the tree represents a queue and the branch weight, its rate allocation. To maintain delay guarantees, per-connection queueing is required for real-time connections [82] i.e., a leaf node for each CBR/VBR con-

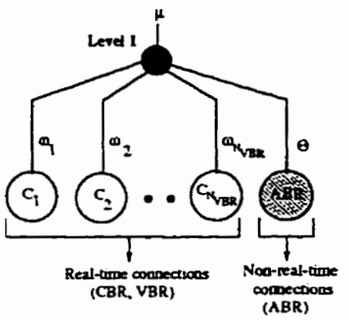


Figure B.2: One-level work-conserving model

nection. However, since ABR traffic is more delay-tolerant [1],[2], data connections can be aggregated into a single queue (i.e., shaded leaf node in Fig. B.2). To ensure that the ABR flows receive their minimum guarantees, the ABR buffer is allocated its required rate, Θ cells/second. Nevertheless, this straightforward "flat" hierarchy may cause problems for delay-sensitive traffic, since all *queues* in Fig. B.2 can get equal relative shares of any idle bandwidth. Specifically, data traffic can receive large amounts of idle bandwidth when real-time connections are backlogged (depending upon the service discipline). Thus a better strategy is to serve ABR traffic at its guaranteed rate, Θ cells/second, and only allocate it extra bandwidth if *no* real-time connections are backlogged. This objective can be achieved using a *hierarchical* link sharing approach [81].

Consider the two-level service hierarchical tree shown in Fig. B.3, where there are two work-conserving servers (represented by dark circles). The Level 1 server for real-time traffic has a service capacity lower-bounded by Λ, where:

$$\sum_{i=1}^{N_{VBR}} \omega_i \leq \Lambda \leq \mu - \Psi. \tag{B.3}$$

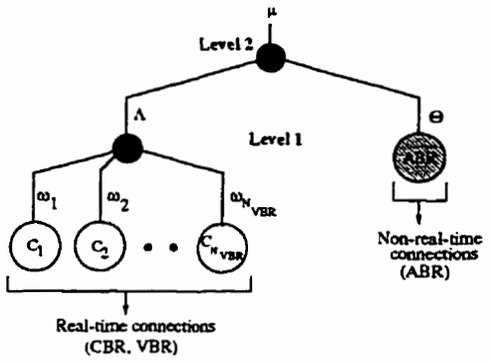


Figure B.3: Hierarchical work-conserving model

The subsequent Level 2 server handles the two traffic classes, real-time and data, and allocates the ABR traffic a minimum service rate of Θ cells/sec. The real-time flows are usually given a higher priority for the remaining available link capacity, namely $\Lambda = (\mu - \Theta)$ cells/second. Hence the strategy favours real-time connections by regulating the ABR flow rates. A practical realization of this model is now presented.

B.2 Hierarchical Fair Queueing

The proposed hierarchical service model is shown in Fig. B.4. This is a two-level service model consisting of three servers: a real-time traffic (VBR) server, a data traffic (ABR) server, and a link server. The link server handles the output from both of the individual class servers. The i-th real-time connection is denoted as VC^i, *virtual connection* (Fig. B.4), and ABR RM control cells are separately buffered at the ABR server, i.e., express RM cell queueing, Section 3.4. Both the VBR and link servers are of the fair queueing type [85]. The ABR server, however,

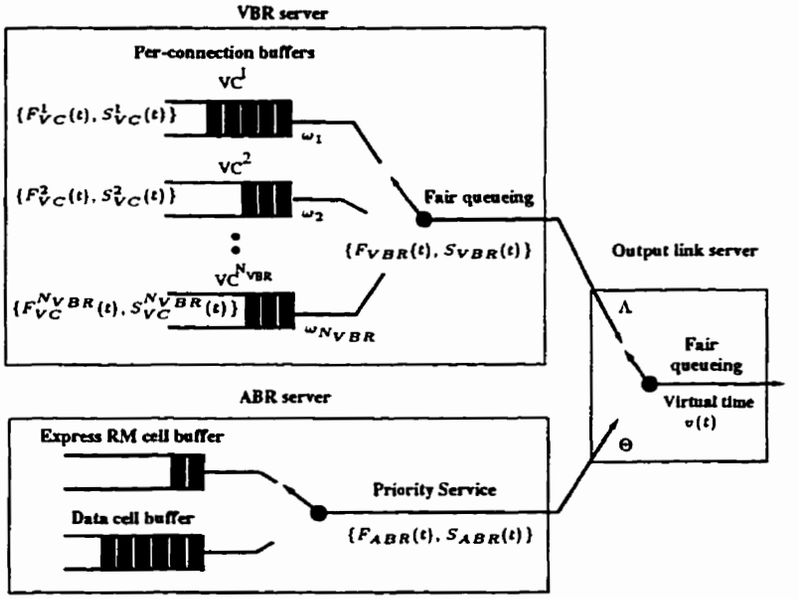


Figure B.4: Hierarchical Fair Queuing (HFQ) server for ABR/VBR traffic

gives service priority to RM cells in order to minimize feedback delays. The above server model of Fig. B.4 is referred to as *hierarchical fair queueing* (HFQ).

Many fair queueing scheduling algorithms have been proposed in the literature and can be employed in the HFQ approach [83],[84]. Since the major concern here is providing integrated services support and not designing new scheduling algorithms, it is best to draw from the existing body of work and choose acceptable schemes. In particular, a modified version of the *start-time fair queueing* (SFQ) algorithm [94] is selected to implement the fair queueing servers in Fig. B.4. In the original SFQ algorithm [94], *virtual starting times* [88] are associated with each cell in the queue, requiring cell headers to be time stamped upon arrival. For relatively small cell sizes and high link rates, this operation may be expensive. In the modified version, stamps are only maintained for the the *head-of-line* (HOL) position of the

queues, thereby reducing implementation complexity (somewhat similar to [90],[93] for SCFQ implementation). It can be shown that this simplification still renders an exact implementation of the original SFQ algorithm (see Appendix C).

There are two values associated with each pertinent queue in Fig. B.4, namely, its virtual time function and *head-of-line* (HOL) stamp value. Hence "local" virtual time functions are defined for both the ABR and VBR servers, $F_{ABR}(t)$ and $F_{VBR}(t)$, respectively, and each real-time connection VC^i, $F^i_{VC}(t)$ (see Fig. B.4). The corresponding HOL stamp values are $S_{ABR}(t)$ (ABR server), $S_{VBR}(t)$ (VBR server), and $S^i_{VC}(t)$ (each VC^i). In addition, there is a global virtual time function defined for the output link server, $v(t)$, which is set to the HOL stamp value of the current cell in service (or the last cell served in the case of an idle link). Hence the design has a simple $O(1)$ virtual time function and overall $O(log(N_{VBR}))$ computational complexity (since queues must be sorted by their HOL values).

B.2.1 Cell Arrival Algorithm

The pseudocode for the cell arrival algorithm is given in Fig. B.5, and assumes a slotted-time, *non-cut-through* implementation [84]. Although extensions to a cut-through version are straightforward, these are omitted for clarity. Whenever a cell reaches the HOL position of its respective queue, the virtual clock updates its virtual time. First, consider a cell entering the HOL position of the VC^i buffer (Fig. B.4) at time t. This can occur either due to a prior cell transmission from this buffer or an arrival at an empty buffer. The HOL stamp value and virtual time

```
if (VBR cell) /* VBR cell */
{
 if ($VC^i$ buffer empty) /* check if buffer empty */
  {
   /* Update $VC^i$ HOL and virtual time values */
   $S^i_{VC}(t) = \max\{F^i_{VC}(t), F_{VBR}(t)\}$
   $F^i_{VC}(t) = S^i_{VC}(t) + \frac{1}{\omega_i}$
   if( $VC^j$ buffer empty $\forall j$ ) /* check if all VBR empty */
    {
     /* Update aggregate VBR HOL and virtual time values */
     $S_{VBR}(t) = \max\{F_{VBR}(t), v(t)\}$
     $F_{VBR}(t) = S_{VBR}(t) + \frac{1}{\Lambda}$
    }
  }
  Enqueue cell in $VC^i$ buffer
}
else /* ABR cell */
{
 /* Check if both ABR buffers empty */
 if ( (express RM buffer empty) && (data buffer empty) )
  {
   /* Update aggregate ABR HOL and virtual time values */
   $S_{ABR}(t) = \max\{F_{ABR}(t), v(t)\}$
   $F_{ABR}(t) = S_{ABR}(t) + \frac{1}{\Theta}$
  }
 /* Enqueue cell appropriately */
 if (RM cell)
  { Enqueue cell in express RM buffer } /* RM cell */
 else
  { Enqueue cell in data buffer } /* data cell */
}
```

Figure B.5: Cell arrival algorithm

function for this connection are updated in the following order:

$$S^i_{VC}(t) = \max\{F^i_{VC}(t), F_{VBR}(t)\}, \tag{B.4}$$

$$F^i_{VC}(t) = S^i_{VC}(t) + \frac{1}{\omega_i}. \tag{B.5}$$

Note that the above updates are performed relative to the virtual time function at the *VBR server*, $F_{VBR}(t)$, rather than at the link server, $v(t)$, due to service hierarchy in Fig. B.4. Also, the HOL value $S^i_{VC}(t)$ represents the (virtual) starting time [94] as opposed to finishing time [90],[91],[93] of the arriving cell. If all the other VBR buffers are empty (i.e., the VC^i HOL cell is the only cell in the VBR server), then the VBR server values must also be updated appropriately:

$$S_{VBR}(t) = \max\{F_{VBR}(t), v(t)\}, \tag{B.6}$$

$$F_{VBR}(t) = S_{VBR}(t) + \frac{1}{\Lambda}. \tag{B.7}$$

For the ABR server, because RM cells have service priority, updates are done in a different manner. If an ABR cell arrives at the HOL position of a given ABR buffer, (either the express RM or data buffer) the ABR virtual time function and HOL stamp values are updated *only* if the other ABR buffer does not already contain a HOL cell. In other words, the arriving HOL cell must be the only HOL cell amongst *both* the express RM cell and data buffers. In this case, the updates

are done in the following order:

$$S_{ABR}(t) = \max\{F_{ABR}(t), v(t)\}, \tag{B.8}$$

$$F_{ABR}(t) = S_{ABR}(t) + \frac{1}{\Theta}. \tag{B.9}$$

In addition, no updates are required for cells arriving at non-empty buffers (both VBR and ABR), since only HOL cells are considered for transmission.

B.2.2 Cell Departure Algorithm

The cell departure algorithm, shown in Fig. B.6, is executed upon cell transmission (or arrival at an empty server). If there are cells buffered for transmission, the link server picks the class with the smaller HOL value, with all ties being resolved in favour of VBR traffic. If the VBR server has the smaller HOL value, then the HOL cell is extracted from the VC^i with the minimum HOL value and the global virtual time function, $v(t)$, is updated with its stamp value. Subsequently, the respective values for the connection must be updated. The virtual time function, $F^i_{VC}(t)$, is updated in the same manner as in the cell arrival case (i.e., with respect to $F_{VBR}(t)$, Eq. (B.5)). However, the HOL value is updated slightly differently. If there are additional cells in the VC^i buffer, the update is the same as in the cell arrival case, Eq. (B.4). Otherwise, the buffer is now empty, and the HOL value is set to a large value, thereby "removing" VC^i from contention (i.e., $S^i_{VC}(t) = \infty$, Fig. B.6). Similarly, updates are also done for the aggregate VBR server at the

```
/* Check if there are any cells for transmission */
if ( (VBR queue not empty) || (ABR queue not empty) )
{
 if (S_VBR(t) ≤ S_ABR(t)) /* check cell type with minimum HOL */
  {
   /* Find and extract cell from VC^i with minimum HOL value */
   i = {j s.t. S^j_VC(t) < S^k_VC(t) ∀k ≠ j}

   Dequeue head-of-line cell in VC^i buffer
   /* Update global virtual time with start time of cell in service */
   v(t) = S_VBR(t)

   /* Update VC^i HOL and virtual time values */
   if (VC^i buffer empty)
    { S^i_VC(t) = ∞ /* clear HOL if empty */ }
   else
    { S^i_VC(t) = max{F^i_VC(t), F_VBR(t)} /* update HOL */ }
   F^i_VC(t) = max{F^i_VC(t), F_VBR(t)} + 1/ω_i /* update virtual time */

   /* Update aggregate VBR HOL and virtual time values */
   if (VC^j buffer empty ∀j)
    { S_VBR(t) = ∞ /* clear HOL if all VBR empty */ }
   else
    { S_VBR(t) = max{F_VBR(t), v(t)} /* update HOL */ }
   F_VBR(t) = max{F_VBR(t), v(t)} + 1/Λ /* update virtual time */

   Transmit dequeued cell
  }
 else /* send ABR cell */
  {
   /* Implement express RM cell priority queueing */
   if (express RM buffer empty)
    { Dequeue head-of-line cell in data buffer }
   else
    { Dequeue head-of-line cell in express RM buffer }

   /* Update global virtual time with start time of cell in service */
   v(t) = S_ABR(t)

   /* Update aggregate ABR HOL and virtual time values */
   if ( (express RM buffer empty) && (data buffer empty) )
    { S_ABR(t) = ∞ /* clear HOL if all ABR empty */ }
   else
    { S_ABR(t) = max{F_ABR(t), v(t)} /* update HOL */ }
   F_ABR(t) = max{F_ABR(t), v(t)} + 1/Θ /* update virtual time */

   Transmit dequeued cell
  }
}
```

Figure B.6: Cell departure algorithm

link server, (i.e., $F_{VBR}(t)$ and $S_{VBR}(t)$).

At the ABR server, the express RM cell priority mechanism requires some additional considerations. First, since the RM express buffer has priority over the data buffer, the next cell is extracted accordingly. The global virtual time function, $v(t)$, is then set to the HOL stamp value of the aggregate ABR server, $S_{ABR}(t)$. The subsequent updates to the ABR virtual time and HOL stamp values are done in much the same manner as for the cell arrival case. The only major difference is that *both* ABR buffers must be empty if the ABR server is to be "removed" from contention at the link server (i.e., $S_{ABR}(t) = \infty$ iff both RM express and data buffers are empty, Fig. B.6).

B.2.3 Impact on VBR Traffic

Since the link server only serves two (aggregate traffic) queues, the maximum service latency for either traffic class is $O(1)$, even though a simple SFQ scheduler is being used (i.e., VBR server is a *fluctuation constrained* server [95], with parameters $(\Lambda, \frac{2\Lambda}{\mu} + 1)$, and likewise for the ABR server with $(\Theta, \frac{2\Theta}{\mu} + 1)$, see [85]). In other words, the maximum delay deviation relative to an ideal class server running at the dedicated rate (Θ or Λ cells/second) is of order one cell transmission time. Therefore, the gains achieved by using more advanced virtual time functions/schedulers at the *link* server may be minimal.

Even though the SFQ scheduler at the VBR server (Fig. B.4) has relatively large *delay guarantee*, $O(N_{VBR})$ [85],[94], it has been shown analytically and experimentally in [94] that for relatively low throughput applications (i.e., $\omega_i \ll \mu$), the

delay bound is actually *less* than that with more complex schemes [89]. Regardless, the design is flexible and if desired, can be easily modified to handle different (more advanced) schedulers at the VBR server, and even hierarchies between real-time connections (i.e., separate CBR and VBR servers). The only requirement would be to "chain" the VBR server's virtual time, $F_{VBR}(t)$, to the virtual time at the link server (i.e., $F_{VBR}(t) = \max\{F_{VBR}(t), v(t)\} + \frac{1}{\Lambda}$ and $S_{VBR}(t) = \max\{F_{VBR}(t), v(t)\}$, as is done in Figs. B.5 and B.6).

B.2.4 Impact on ABR Traffic

Apart from bandwidth-delay effects, another major issue which ABR flow control schemes must contend with is network variability. It is generally accepted that end-to-end control schemes are best suited for dynamics of time scales greater than the connection roundtrip delays [12]. Therefore, it is clearly in the best interest of ABR traffic to ensure that the VBR fluctuations are "smooth" and occur on large enough time scales. Along these lines, scheduling algorithms can also be developed to restrict the short-term fluctuations of VBR traffic to acceptable time scales (i.e., improve ABR/VBR interaction in favour of ABR traffic). One possible option is to have time-varying quantities $\Lambda(t)$ and $\Theta(t)$. This would allow ABR feedback control schemes to react properly and effectively control queue sizes. The design of such "hybrid" schemes, though open to further research, is beyond the scope of the current work.

B.3 Simulation Results

The performance of the HFQ algorithm is now compared with some other scheduling strategies via simulation. In particular, comparisons are made with two simpler one-level approaches, both using SFQ schedulers also. The first scheme implements the flat one-level hierarchy in Fig. B.2, where the ABR traffic is assigned a portion, Θ cells/second, of the link rate. This is termed the *regular SFQ* approach. The second scheme gives non-preemptive priority to VBR traffic, only serving ABR traffic during VBR idle periods, and is labelled the *VBR-priority* approach. Hence, as far as the ABR traffic is concerned, the VBR-priority approach is closest to the non-preemptive priority approach used in Chapter 4 (with the exception of per-connection queueing for individual VBR connections). Express RM cell queueing is also done with both level-one approaches.

To properly gauge the algorithms, additional performance metrics have to be given for real-time traffic measurements. Specifically, due to its delay-sensitive nature, the most important measure for a real-time stream is its mean cell delay and cell delay variation. For data traffic, however, the previous metrics still apply (i.e., goodput, queue metrics, Section 2.2.1). The EDERA control parameter settings are the same as those in Table 4.1, and the bandwidth scaling constants, β and γ, are fixed at 0.05 each. Furthermore, a relatively simple CAC rule is used for VBR traffic, based upon *mean* bandwidth utilization (i.e., total mean VBR rate less than Λ cells/sec.). In particular, no UPC is done for the VBR connections, allowing them to transmit directly into the network (aggressive sources). Although more detailed, and conservative, admission strategies coupled with leaky-bucket UPC can be used,

these are more CAC issues and are not investigated here. Rather the main goal is to evaluate ABR performance for a wide range of VBR traffic behaviour, not overly relying on the CAC algorithm to ensure ABR goodput. Although the schedulers are tested for various networks, both single and multiple link, only multiple link results are presented. This is because findings indicate that the observed trends are very similar in both cases. Hence multiple link results are presented to better show end-to-end performance. All results are gathered over two independent runs of one second each.

Consider the *parking-lot* scenario detailed in Fig. 2.14 with forty ABR connections and ten VBR connections. The detailed behaviour of all the connections is given in Table B.1, where most notably, all ABR connections have an MCR of 1 Mbps (different from Table 2.7). All link rates are fixed at 353,773 cells/second (i.e., approximately 150 Mbps), link delays to 1 ms, and source/destination end-system delays to 1 μs. The rates assigned to the traffic classes and VBR connections are summarized in Table B.2. Although there is some flexibility in choosing these values, identical settings are chosen at all links, with maximum possible allocations being given to the real-time traffic. Thus, for the HFQ and regular SFQ schedulers, the ABR allocation is set to 20% of the link capacity, ensuring that the MCR guarantees are met at the most bottlenecked link, link 4-5 (i.e., minimum possible network-wide ABR allocation, $\Psi = \Theta = 70,755$ cells/second, roughly 30 Mbps). The remaining 80% of the link bandwidth is assigned to the VBR traffic (i.e., $\Lambda = 283,018$ cells/second, about 120 Mbps), and each VBR connection is allocated an even share of this portion, $\omega_i = 28,301$ cells/second, approximately 12 Mbps. In

Table B.1: Multiple switch network in Fig. 2.14 (all rates in Mbps)

Connections	Type	Description	Throughput
1-10	ABR	Persistent, 1 MCR, 150 PCR	$5 - \frac{5x}{30}$
11-20	ABR	Persistent, 1 MCR, 150 PCR	$10 - \frac{x}{3}$
21-30	ABR	Persistent, 1 MCR, 150 PCR	$5 - \frac{5x}{30}$
31-40	ABR	Persistent, 1 MCR, 150 PCR	$5 - \frac{5x}{30}$
41-50	VBR	Bursty, exponential 5 ms on/off periods $14.4 \leq x \leq 22.8$ Mbps in on period	$\frac{x}{2}$

Total VBR loading=$\frac{5x}{120}$, varies from 60% for $x = 14.4$ Mbps, up to 95% for $x = 22.8$ Mbps

the VBR-priority server, however, there is no allocation made to ABR traffic and thus all VBR connections get equal rate allocations ($\omega_i = 35,377$ cells/second $\forall i$, about 15 Mbps).

There are four groups of ABR connections, with each group containing ten connections. The VBR connections, meanwhile, are bursty with each VBR source having exponentially distributed on and off periods of mean 5 ms. The source rate in the on state, x Mbps (Table B.1), is varied to give *mean* aggregate VBR usages (i.e., loadings) ranging from 60% to 95% of the allocated VBR guarantee, 283,018

Table B.2: Scheduling algorithm allocations (all rates in cells/second)

Scheduler	ABR Rate Θ	VBR Rate Λ	VC^i Rate ω_i $(1 \leq i \leq N_{VBR})$
HFQ	70,755	283,018	28,301
SFQ	70,755	-	28,301
VBR-Priority	-	-	35,377

– denotes not applicable

cells/second (120 Mbps). Since the EDERA algorithm implements the MCR-plus-equal-share bandwidth fairness criterion, the ideal mean ABR throughputs can be computed (see Table B.1). For example, at the most bottlenecked link 4-5, the total mean VBR usage is $(10 - \frac{x}{2}) = 5x$ Mbps, and the total MCR commitment is 30 Mbps. This gives $(150 - 5x - 30)$ Mbps to be divided equally amongst all ABR connection at the link, yielding a fair share value of $(5 - \frac{5x}{30})$ Mbps per connection (i.e., MCR plus equal share). Since connections 11-20 are bottlenecked at link 1-2, it can be similarly shown that their fair allocations are higher, $(10 - \frac{x}{3})$ Mbps.

Fig. B.7 and Fig. B.8 show the mean and standard deviation of the link 4-5 (most bottlenecked) queue length for differing VBR loadings. Here, the queue oscillations increase significantly with the VBR-priority scheduler for increased VBR loads (i.e., both mean and standard deviation more than double for 95% loading). This occurs since the VBR-priority scheme does not provide any form of cell-level rate guarantee to the ABR traffic. Meanwhile, the HFQ and SFQ schedulers perform much better, showing little sensitivity to the VBR load. In most scenarios tested, both HFQ and SFQ tend to give roughly equivalent queue control. The effect of the increased mean and variance can also be seen in Fig. B.9, which plots sample queue behaviour at 95% VBR loading for the HFQ and VBR-priority schemes. The results confirm much larger operating values with the VBR-priority approach. In general, since the ABR source specifications always allow sources to send at their MCR guarantees (or more) [1],[24], it is conjectured that other ABR flow control algorithms will also yield similar queueing behaviour with the VBR-priority scheme.

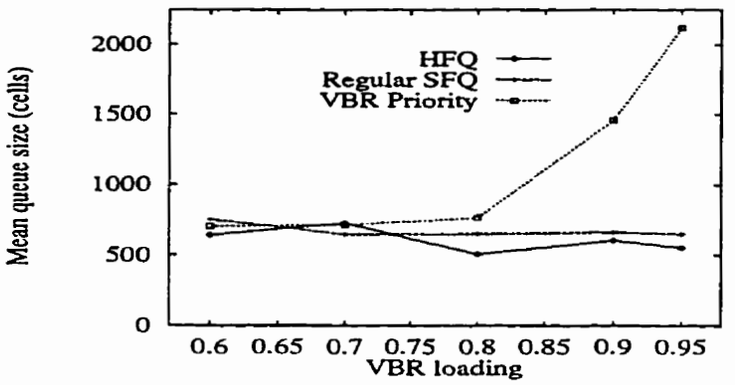


Figure B.7: Link 4-5 mean queue length in Fig. 2.14

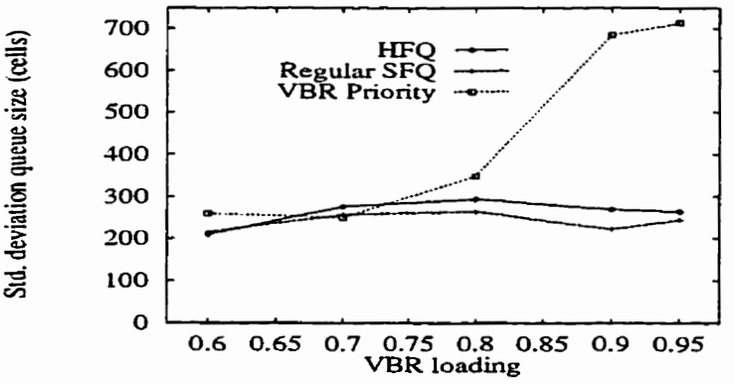


Figure B.8: Link 4-5 standard deviation of queue length in Fig. 2.14

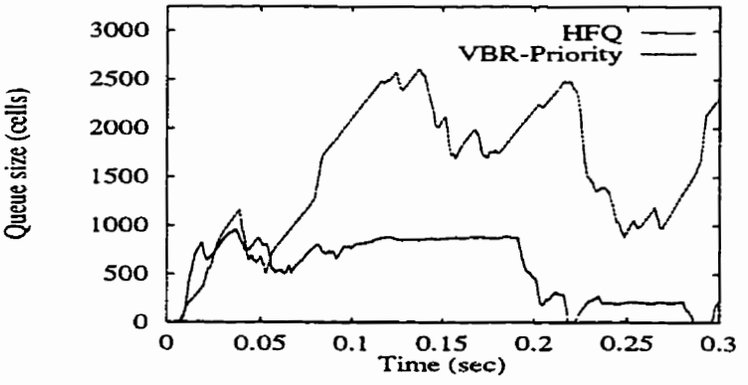


Figure B.9: Link 4-5 sample queue behaviour, 95% VBR utilization in Fig. 2.14

To measure ABR bandwidth fairness in Fig. 2.14, the maximum *deviation* from the ideal mean throughputs (Table B.1) over all connection *groups* is measured. Note that this value is more (or partly) a function of the mean VBR utilization and of the bandwidth distribution properties of the end-to-end ABR flow control scheme, rather than the cell-level scheduling algorithms. The results, shown in Fig. B.10, indicate that the maximum deviation stays under 10% for moderate VBR loads, and increases slightly for heavier loads. Overall, the HFQ scheduler gives slightly better fairness. At higher loads, however, the larger range of VBR fluctuations are more difficult to track with feedback control schemes, and throughputs will suffer. Nevertheless, the EDERA algorithm is still very effective even in relatively hostile environments (under 15% deviation), and allocates bandwidth "fairly" relative to the long-term mean VBR rate. Also note that part of the deviation can be attributed to the diversity in feedback delays for the three connection groups in Fig. 2.14 (i.e., bandwidth-delay effects).

Fig. B.11 and Fig. B.12 plot the mean and standard deviation of the cell delays for the real-time VBR connections in Fig. 2.14. The values represent the statistics for *normalized* delays, adjusted by the fixed delay component at all links, i.e., link propagation delays, cell transmission times. Hence they reflect purely the *buffering* delay statistics. As expected, the VBR priority server gives the lowest delays, since ABR traffic is not served at all when the VBR buffers are busy. Conversely, the one-level regular SFQ server does not adequately "protect" the VBR connections. From Fig. B.11, it is seen that the HFQ scheduler provides considerably smaller delays than the regular SFQ scheduler (by almost an order of magnitude). In addition,

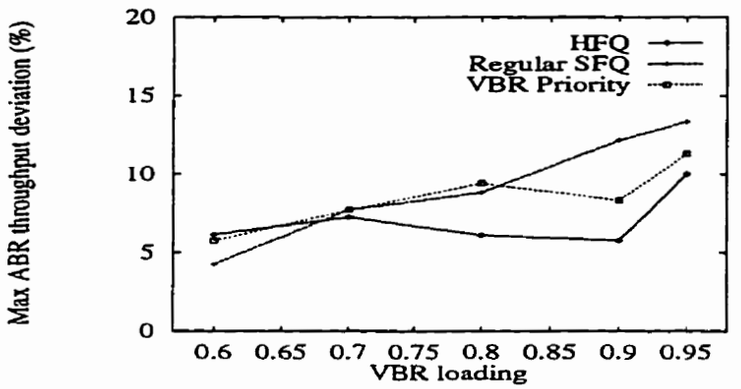


Figure B.10: Maximum ABR throughput deviation in Fig. 2.14

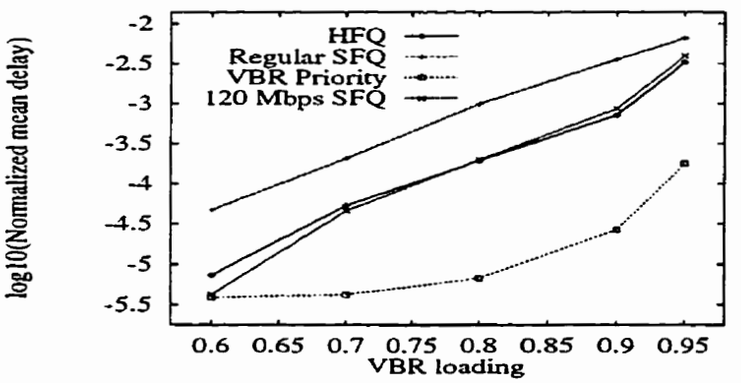


Figure B.11: Mean VBR cell delay in Fig. 2.14

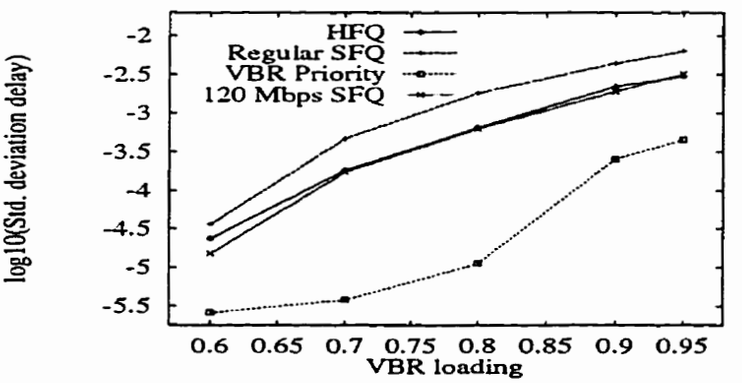


Figure B.12: Standard deviation of VBR cell delays in Fig. 2.14

if there is larger separation between the ω_i values and Θ, the delay discrepancy between the HFQ and SFQ schedulers increases further. However, the HFQ delays are still considerably larger than those with the VBR-priority server. This arises due to the bandwidth "firewall" in the HFQ scheme to protect the ABR minimum throughput guarantee component from misbehaving VBR connections. Thus as an additional comparison, consider the delay performance of the HFQ server compared to that of a dedicated SFQ server, serving *only* the ten VBR connections. Let this server have a reduced link rate of 120 Mbps (i.e., equal to the $\Lambda = 283,018$ cells/second VBR allocation in the HFQ scheduler) and all connections have equal rates ($\omega_i = 28,301$ cells/second $\forall i$, about 12 Mbps). In essence, this is fairer a comparison between the HFQ and VBR priority schedulers, since the VBR traffic should not be using more than its share of the link rate. The delay performance for this server is also plotted in Fig. B.11 and Fig. B.12, labelled as *120 Mbps SFQ*, and is almost identical to that of the HFQ server. This confirms proper bandwidth allocation by the link server in Fig B.3. Overall, the HFQ scheme achieves a good tradeoff between controlling VBR cell delays and maintaining ABR performance, both queue sizes and throughput.

Additional simulation results confirm that the hierarchical scheduling strategy is well-suited for integrating ABR data and real-time VBR services inside ATM networks. The setup can effectively control real-time cell delays and preserve ABR throughputs (i.e., MCR guarantees) in dynamic network environments. The ABR queueing performance is also significantly better than with the the well-known non-preemptive VBR priority approach, preventing large buildups at heavier loads.

Appendix C

SFQ Equivalency

Consider a general *level-one* SFQ link scheduler as defined in [94], serving N connections at the link rate of μ cells/second. Each connection i has an associated link share, ω_i (where $\sum_{i=1}^{N} \omega_i \leq \mu$) and a virtual (finishing) time function, $F_i(t)$. The global time function is given by $v(t)$, and equals the stamp value of the cell currently being served. The SFQ algorithm stamps the j-th cell for connection i, arriving at time t, with its (virtual) starting time value:

$$S_i^j(t) = \max\{F_i^{j-1}(t), v(t)\}, \tag{C.1}$$

and then advances the connection i finishing time (virtual clock) as:

$$F_i^j(t) = S_i^j(t) + \frac{1}{\omega_i}. \tag{C.2}$$

In other words, when cell j reaches the HOL position of the connection i buffer, it will contend with value $S_i^j(t)$. Similarly, denote by SFQ$'$ the modified version of the SFQ algorithm presented in Appendix B, which only stamps cells when they reach the HOL position. Let the respective values for this scheduler be denoted as $F_i'^j(t)$, $S_i'^j(t)$, and $v'(t)$. In particular for the j-th cell for connection i arriving at the HOL position of its buffer at time t, the (virtual) starting time stamp is:

$$S_i'^j(t) = \max\{F_i'^{j-1}(t), v'(t)\}, \tag{C.3}$$

(i.e., similar to Eq. (B.4), where $F_{VBR}(t)$ is now $v'(t)$). The (virtual) finishing time is updated accordingly as:

$$F_i'^j(t) = S_i'^j(t) + \frac{1}{\omega_i}. \tag{C.4}$$

Theorem C.1

Given identical initial settings for all virtual time values (i.e., $F_i(0) = F_i'(0)$, $S_i(0) = S_i'(0)$, $v(0)$, $v'(0)$), the SFQ and SFQ$'$ algorithms produce identical cell orderings for the same set of connection inputs.

Proof

To show this, only the stamp values of the *HOL* cells need to be considered. This is because even for the SFQ scheduler, a given connection's time stamp values are monotonically increasing. Therefore, clearly, a connection's HOL stamp will be the minimum value from its (non-empty) queue which can contend for link access. Therefore SFQ and SFQ$'$ equivalency can be demonstrated by showing that both

algorithms give identical HOL values (for the same input streams during a busy period). In addition, it is not difficult to see that the overall *durations* of the busy periods will be identical in both schedulers. Now consider a cell j arriving at its respective buffer at time s and reaching the HOL position at time t, $s \leq t$. There are two possible ways that this cell can reach the HOL position of its respective queue, and these are considered separately:

Case A: Connection i buffer is empty at time s (i.e., $s = t$)

This case represents the start of a busy period. Here the cell arrives at the HOL position of the connection i buffer, and $v(s) = v'(s)$. In the SFQ server, since the previous cell has already departed the scheduler, the cell is stamped with:

$$S_i^j(s) = \max\{F_i^{j-1}(s), v(s)\} = v(s), \tag{C.5}$$

(i.e., since $v(s) \geq F_i^{j-1}(s)$ and the previous connection i cell has departed the server). The corresponding stamp in the SFQ$'$ server is:

$$S_i'^j(s) = \max\{F_i'^{j-1}(s), v(t)\} = v(t), \tag{C.6}$$

and is identical to Eq. (C.5) since $s = t$.

Case B: Connection i buffer is non-empty at time s (i.e., $s \neq t$)

Here the arriving cell j is placed in the connection i buffer at time s in both schedulers. Now in the the SFQ scheduler, the cell gets stamped with the value $S_i^j(s) = \max\{F_i^{j-1}(s), v(s)\}$, and this value is used for link contention when the cell reaches the HOL position at time t. However, since the queue is backlogged,

the stamp value can be re-written as

$$S_i^j(s) = \max\{F_i^{j-1}(s), v(s)\} = F_i^{j-1}(s). \tag{C.7}$$

In other words, since $v(s)$ is the (starting) time stamp value of current cell in service, and this is the *minimum* over all connections, then $F_i^{j-1}(s) > S_i^{j-1}(s)$ via Eq. (C.2).

Meanwhile, in the SFQ′ scheduler, the arriving cell is only stamped when it reaches the HOL position at time t with value $S_i'^j(t) = \max\{F_i'^{j-1}(t), v(t)\}$. However, since the buffer is backlogged and the cell *ahead* of the reference cell has just departed, then the (initial) virtual time $v(t^-) = S_i'^{j-1}(t) < F_i'^{j-1}(t)$, giving

$$S_i'^j(t) = \max\{F_i'^{j-1}(t), v(t)\} = F_i'^{j-1}(t). \tag{C.8}$$

Note that Eq. (C.7) and Eq. (C.8) are very similar and evolve in the same manner. Furthermore, when the *first* burst of connection i cells arrives at the schedulers, it is known that $F_i^j(s) = F_i'^j(s)$, i.e., both schedulers' virtual clocks are identical (since $S_i^j(s) = S_i'^j(s)$, *Case A* above). Given that the initial virtual starting times are identical and their subsequent evolutions during the busy period are the same, then clearly the HOL stamp values will be identical. ∎

Note that the above proof only shows that cell orderings are preserved in a one-level scheduler. A recursive extension to hierarchical servers is straightforward.

Appendix D

Convergence Analysis

For the case of fixed capacity links and persistent (or constant) connections, under certain appropriate assumptions, it can be shown that the rate allocation procedure of the EDERA scheme (Section 3.3.1) converges to max-min bandwidth fairness [3]. To simplify the derivation, it is assumed that the rate allocation is done according to the actual ACR values in received RM cells (as opposed to the SASR values, Section 3.3.1). All connections have zero MCR requirements and the utilization threshold ρ is set to unity at all links. Both the single and multiple link cases are analysed, and subsequently, various implications on convergence behaviour in more general operating environments are discussed.

D.1 Single Link Case

Consider a network link with capacity μ cells/second, supporting a set of N traversing connections. The connections are classified into the appropriate groupings at

the link, bottlenecked, $\mathcal{I}(t)$, and non-bottlenecked, $\overline{\mathcal{I}}(t)$, respectively. These partitions are updated upon both forward and backward RM cell reception in accordance with the rate allocation algorithm in Fig. 3.4. Clearly, the ideal max-min fair rate for all connections is $\frac{\mu}{N}$ cells/second. However, during transient periods, the *computed* fair share for connection i, $ER^i_{sw}(t)$ (Eq. (3.7)), is generally not equal to the ideal steady-state value. Hence, consider the following:

Definition D.1 (Consistency)

The groupings $\mathcal{I}(t)$ and $\overline{\mathcal{I}}(t)$ are consistent *if:*

$$\begin{aligned} ACR^i_{sw}(t) &< ER^i_{sw}(t) \quad \forall i \in \overline{\mathcal{I}}(t) \\ ACR^i_{sw}(t) &\geq ER^i_{sw}(t) \quad \forall i \in \mathcal{I}(t). \end{aligned} \tag{D.1}$$

In other words, a consistent grouping at time t is one where all connections are classified correctly according to their current *fair share values.*

Specifically, all bottlenecked connections must have stored rates above their fair shares, and likewise all non-bottlenecked connections must have rates below their fair shares. Given consistent connection groupings at a link, a further condition arising from Eq. (D.1) can be derived. In particular, for a non-bottlenecked connection i, clearly the following holds at time t (via Eq. (3.7)):

$$ACR^i_{sw}(t) < \frac{\mu - C_{\overline{\mathcal{I}}}(t) + ACR^i_{sw}(t)}{|\mathcal{I}(t)| + 1}. \tag{D.2}$$

Re-arranging the above expression gives:

$$ACR^i_{sw}(t) < \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|}, \tag{D.3}$$

and summing the above over all non-bottlenecked connections (and simplifying) yields:

$$\begin{aligned} C_{\overline{\mathcal{I}}}(t) = \textstyle\sum_{k \in \overline{\mathcal{I}}(t)} ACR^i_{sw}(t) < |\overline{\mathcal{I}}(t)| \cdot \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} \\ \Rightarrow C_{\overline{\mathcal{I}}}(t) < |\overline{\mathcal{I}}(t)| \cdot \frac{\mu}{N}, \end{aligned} \tag{D.4}$$

(since $|\mathcal{I}(t)| + |\overline{\mathcal{I}}(t)| = N$). The above expression implies that for consistent groupings, the aggregate bandwidth usage by non-bottlenecked connections cannot exceed their *proportional* bandwidth share at the link.

The condition in Eq. (D.1) is similar to the *M-consistency* condition in [35]. In particular, if consistency holds at all times, as it does in more complex $O(n)$ update algorithms [35],[40], it can be shown that the algorithm converges to max-min fair allocations in finite time [34],[35],[41]. However, in the case of the proposed algorithm, consistency cannot be assumed at all times due to the simpler $O(1)$ rate update procedure. This is because only the grouping of the connection corresponding to the received RM cell is updated, even though the subsequent classifications of other connections can change. Since the statuses for other connections are not updated, "misclassification" can occur and this has a subtle impact on the convergence behaviour. To partly address this issue, a looser form of consistency, *G-consistency*, is defined in order to show convergence to fair rates. The implications of this assumption are discussed subsequently in Section D.3. Consider

Definition D.2 (G-consistency)

The groupings $\mathcal{I}(t)$ and $\overline{\mathcal{I}}(t)$ are G-consistent *if:*

$$ACR^i_{sw}(t) < ER^i_{sw}(t) \qquad \forall i \in \overline{\mathcal{I}}(t). \tag{D.5}$$

In addition, connection i satisfies G-consistency if:

$$ACR^i_{sw}(t) < ER^i_{sw}(t) \quad \text{if} \quad i \in \overline{\mathcal{I}}(t), \tag{D.6}$$

(even though G-consistency may not hold at the link). Bottlenecked connections always satisfy G-consistency, regardless of their stored rates.

Clearly, from Eq. (D.1), (regular) consistency is a special case of G-consistency. All Eq. (D.5) requires is that non-bottlenecked connections have stored usages below their fair shares. This implies that Eq. (D.4) also holds for G-consistent groupings. Essential to the subsequent convergence argument is the following:

Lemma D.1

If the connection groupings are G-consistent (or consistent) at time t, then for all bottlenecked connections,

$$ER^i_{sw}(t) \geq \frac{\mu}{N}. \tag{D.7}$$

Proof

Using Eq. (3.7), the fair share is computed as:

$$\begin{aligned} ER^i_{sw}(t) &= \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} \\ &\geq \frac{\mu - |\overline{\mathcal{I}}(t)| \cdot \frac{\mu}{N}}{|\mathcal{I}(t)|} \\ &= \frac{N - |\overline{\mathcal{I}}(t)|}{|\mathcal{I}(t)|} \cdot \frac{\mu}{N} = \frac{\mu}{N}, \end{aligned} \tag{D.8}$$

where the second step makes use of Eq. (D.4). ■

Lemma D.2

If the connection groupings are G-consistent (or consistent) at time t, then all non-bottlenecked connections become bottlenecked in finite time.

Proof

If connection i is non-bottlenecked, then by definition, its latest rate at the link is less than $ER^i_{sw}(t)$, i.e., via Eq. (D.5). As such, this connection is always allowed to increase its rate. Consider the first *non-bottlenecked connection* reaching its *computed* fair share, connection i. Clearly, in this case

$$ACR^i_{sw}(t) = \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} = ER^i_{sw}(t), \tag{D.9}$$

(i.e., Eq. (D.2) with equality) and thus connection i is moved to the bottlenecked grouping. ■

Using the above lemmas, it is now shown that:

Theorem D.1

If connection groupings are G-consistent for all $t \geq \tau$, then the rate allocation algorithm will converge to max-min fair allocations in finite time.

Proof

Clearly, if G-consistency holds, then for any bottlenecked connection i, all computed $ER^i_{sw}(t)$ updates are *at least* as large as the steady-state max-min fair share, $\frac{\mu}{N}$ cells/sec. Furthermore, via Lemma D.2, after a sufficient time duration all the *stored* rates at the link, $ACR^i_{sw}(t)$, will be at least equal to the max-min fair share (regardless of the size of the AIR increment). However, via Eq. (D.4), for G-consistency to hold, necessarily all connections must be classified as bottlenecked. Hence $C_{\overline{\mathcal{I}}}(t) = 0$ and Eq. (D.7) holds with equality. ∎

D.2 Multiple Link Case

The convergence analysis in a multiple link network setting is now considered. Consider the network modelled as a graph, $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where the sets $\mathcal{N}$ and $\mathcal{L}$ represent the network switches and *bidirectional* links, respectively. Link j has capacity μ_j cells/second and supports a set of traversing connections denoted by the set $\mathcal{A}_j$. These connections are classified into the appropriate groupings (partitions) at the link, bottlenecked, $\mathcal{I}_j(t)$, and non-bottlenecked, $\overline{\mathcal{I}}_j(t)$, respectively (i.e., $\mathcal{A}_j = \mathcal{I}_j(t) \cup \overline{\mathcal{I}}_j(t)$). In addition, each connection has as associated fixed set of links representing its traversed path, $\mathcal{P}_i = \{l^i_1, l^i_2, ..., l^i_{|\mathcal{P}_i|}\} \subseteq \mathcal{L}$. A brief review of max-

min fairness is first presented, and then the convergence behaviour is discussed.

D.2.1 Max-Min Bandwidth Fairness

Since MCR guarantees are not considered, clearly the link(s) carrying the most number of connections is the most bottlenecked (i.e., $\max_j|\mathcal{A}_j|$). All connections traversing this link belong to the level-one bottleneck grouping, $\mathcal{B}_1$, and have max-min fair *steady-state* rates

$$\nu_1 = \min_{j\in\mathcal{L}}\left\{\frac{\mu_j}{|\mathcal{A}_j|}\right\}. \tag{D.10}$$

Let the bottleneck-level for link j be given by b^j, and j^k be the level-k bottleneck link. In other words, $b^{j^k} = k$ and $\mathcal{A}_{j^k} \subseteq (\mathcal{B}_k \cup \mathcal{B}_{k-1} \cdots \cup \mathcal{B}_1)$. As per the iterative procedure in [3, Chapter 6], the second-most bottlenecked link is derived by removing all level-one bottlenecked connections and their link(s) from the network. The level-one bottleneck link in this reduced network is now the second-most bottlenecked link. All connections traversing the second-most bottlenecked link belong to $\mathcal{B}_2$, and so on and so forth for higher level bottlenecks. Hence, omitting details, it can be shown that the max-min fair rate for connections in $\mathcal{B}_k$ (bottleneck-level k, $k \geq 1$) is given by:

$$\nu_k = \min_{j\neq j^{k-1},\cdots,j^1}\left\{\frac{\mu_j - \sum_{l=1}^{k-1}\sum_{m\in\mathcal{A}_j}\nu_l \cdot \mathrm{I}\{m\in\mathcal{B}_l\}}{|\mathcal{A}_j| - \sum_{l=1}^{k-1}\sum_{m\in\mathcal{A}_j}\mathrm{I}\{m\in\mathcal{B}_l\}}\right\}, \tag{D.11}$$

where $\mathrm{I}\{\cdot\}$ is the event-indicator function. Therefore the bottleneck-level for connection i, g^i, is

$$g^i = \min_{k \in \mathcal{P}_i}\{b^k\}, \tag{D.12}$$

(i.e., $i \in \mathcal{B}_{g^i}$) and its ideal steady-state *max-min* fair rate allocation, r^i, is

$$r^i = \min_{k \in \mathcal{P}_i}\{\nu_{b^k}\}, \tag{D.13}$$

(i.e., the maximum rate is limited to the minimum bottleneck rate along the connection's path).

Some additional quantities are also defined. Let $\mathcal{I}_j$ and $\overline{\mathcal{I}}_j$ be the *steady-state* bottlenecked and non-bottlenecked groupings at link j, respectively, as given by the max-min criterion. In addition, let $C_{\overline{\mathcal{I}}_j}$ be the total bandwidth usage by non-bottlenecked connections at link j:

$$C_{\overline{\mathcal{I}}_j} = \sum_{i \in \overline{\mathcal{I}}_j} r^i = \sum_{l=1}^{b^j - 1} \sum_{k \in \mathcal{A}_j} \nu_l \cdot \mathrm{I}\{k \in \mathcal{B}_l\}. \tag{D.14}$$

Furthermore, denoting the level-k non-bottlenecked grouping at link j by $\overline{\mathcal{I}}_{k,j}$ ($k < b^j$), its total steady-state bandwidth usage is given by (i.e., $\overline{\mathcal{I}}_j = \overline{\mathcal{I}}_{j,k} \cup \cdots \cup \overline{\mathcal{I}}_{b^j,k}$):

$$C_{\overline{\mathcal{I}}_{k,j}} = \sum_{i \in (\overline{\mathcal{I}}_j \cap \mathcal{B}_k)} r^i = \sum_{i \in \overline{\mathcal{I}}_j} \nu_k \cdot \mathrm{I}\{i \in \mathcal{B}_k\} = |\overline{\mathcal{I}}_{k,j}| \cdot \nu_k \leq C_{\overline{\mathcal{I}}_j}. \tag{D.15}$$

Since all quantities are steady-state ideal values, there is no dependence on time.

D.2.2 Network Convergence

The network convergence behaviour is now analysed, assuming that G-consistency holds throughout at *all* links. First consider the most bottlenecked link, link j^1, and its set of traversing connections, $\mathcal{A}_{j^1} = \mathcal{B}_1$. Careful consideration shows that Lemmas D.1 and D.2 still apply here, since they only require G-consistency to hold at the link. Hence

$$ER^i_{sw}(t) \geq \frac{\mu_{j^1}}{|\mathcal{A}_{j^1}|} = \nu_1 \qquad \forall i \in \mathcal{I}_{j^1}(t), \tag{D.16}$$

(via Lemma D.1) and all non-bottlenecked connections are allowed to ramp-up and enter the bottlenecked state (via Lemma D.2). This means that in finite time all connections in $\mathcal{B}_1$ will eventually stabilize at $\frac{\mu_{j^1}}{\mathcal{A}_{j^1}}$, their steady-state max-min fair rates.

Now consider the more general case of a link j with max-min bottleneck level $b^j \neq 1$ and its associated set of traversing connections, $\mathcal{A}_j \subseteq (\mathcal{B}_1 \cup \cdots \mathcal{B}_{b^j})$. Again, via G-consistency and Lemma D.1, for a bottlenecked connection i at link j:

$$ER^i_{sw}(t) \geq \frac{\mu_j}{|\mathcal{A}_j|} \qquad \forall i \in \mathcal{I}_j(t). \tag{D.17}$$

However, *if* in addition the traversing *lower-level* bottlenecked connections have achieved their max-min fair rates, then the lower bound on the computed fair share can be extended further. Specifically, consider

Lemma D.3

If all connections in $\mathcal{B}_n$ ($n < b^j$) have stabilized at their steady-state max-min rates,

then at link j

$$ER^i_{sw}(t) \geq \nu_{bj} \qquad i \in \mathcal{I}_j(t). \tag{D.18}$$

In other words, the computed fair share for any bottlenecked connection at link j is at least equal to its steady-state max-min rate.

Proof

From Eq. (3.7), the bottlenecked fair share is computed as:

$$ER^i_{sw}(t) = \frac{\mu_j - C_{\overline{\mathcal{I}}_j}(t)}{|\mathcal{I}_j(t)|}. \tag{D.19}$$

Since the lower level bottlenecked have stabilized at their max-min rates, these bottlenecked connections are non-bottlenecked at the link: $(\mathcal{A}_j \cap \mathcal{B}_n) \subseteq \overline{\mathcal{I}}_j(t)$ for all $n < b^j$. Therefore the above expression can be re-written as (using Eq. (D.15)):

$$\begin{aligned} ER^i_{sw}(t) &= \frac{\mu_j - \sum_{l=1}^{b^j-1} C_{\overline{\mathcal{I}}_{k,j}} - \sum_{k\in(\overline{\mathcal{I}}_j(t)\cap\mathcal{B}_{bj})} ACR^k_{sw}(t)}{|\mathcal{A}_j| - \sum_{l=1}^{b^j-1} |\overline{\mathcal{I}}_{k,j}| - |(\overline{\mathcal{I}}_j(t)\cap\mathcal{B}_{bj})|} \\ &= \frac{\mu_j' - \sum_{k\in(\overline{\mathcal{I}}_j(t)\cap\mathcal{B}_{bj})} ACR^k_{sw}(t)}{|\mathcal{A}_j'| - |(\overline{\mathcal{I}}_j(t)\cap\mathcal{B}_{bj})|} \\ &= \frac{\mu_j' - C_{\overline{\mathcal{I}}_j'}(t)}{|\mathcal{I}_j'(t)|}, \end{aligned} \tag{D.20}$$

where $\mu_j' = \mu_j - \sum_{l=1}^{b^j-1} C_{\overline{\mathcal{I}}_{k,j}}$, $\mathcal{A}_j' = ((\mathcal{A}_j \backslash \overline{\mathcal{I}}_{1,j}) \cdots \backslash \overline{\mathcal{I}}_{b^j-1,j})$, $\overline{\mathcal{I}}_j'(t) = (\overline{\mathcal{I}}_j(t) \cap \mathcal{B}_{bj})$, $\mathcal{I}_j'(t) = \mathcal{B}_{bj} \backslash \overline{\mathcal{I}}_j'(t)$, and $C_{\overline{\mathcal{I}}_j'}(t) = \sum_{k\in(\overline{\mathcal{I}}_j(t)\cap\mathcal{B}_{bj})} ACR^k_{sw}$ ($\backslash$ denotes set truncation). Basically, the above *legitimately* removes the stabilized lower-level bottleneck connections from the network, and is identical to the reduction step in the global max-min procedure [3]. In this "new" network, link j is now the most bottlenecked

link and it is not difficult to see that at this link (via Lemma D.1)

$$ER^i_{sw}(t) \geq \frac{\mu'_j}{\mathcal{A}'_j} = \nu_{bj}, \tag{D.21}$$

and hence the result is proved. ■

In addition, for non-bottlenecked connections, it can also be shown that:

Lemma D.4

For any connection $i \in \mathcal{B}_n$ ($n < b^j$) which has achieved its steady-state max-min rate, then at link j

$$ER^i_{sw}(t) > ACR^i_{sw}(t) \qquad i \in \overline{\mathcal{I}}_j(t), \tag{D.22}$$

(where $ACR^i_{sw}(t) \leq \nu_{bj-1}\ \forall i \in \mathcal{B}_n,\ n < b^j$). In other words, when all lower-level bottlenecks are resolved, the corresponding connections cannot transition out of their non-bottlenecked *states at higher-level bottleneck links.*

Proof

Assume to the contrary that for connection $i \in \mathcal{B}_n,\ n < b^j$

$$ACR^i_{sw}(t) \geq ER^i_{sw}(t) = \frac{\mu'_j - C_{\overline{\mathcal{I}}'_j}(t) + ACR^i_{sw}(t)}{|\mathcal{I}'_j(t)| + 1} \leq \nu_{bj-1}, \tag{D.23}$$

(as per Eq. (D.20)). Re-arranging the above expression gives:

$$ACR_{sw}^{i}(t) \geq \frac{\mu_j^{'} - C_{\overline{\mathcal{I}}_j}(t)}{|\mathcal{I}_j^{'}(t)|} \geq \nu_k, \tag{D.24}$$

which is not possible via the max-min definition (i.e., since the bottleneck level for connection i is less than b^j). ■

From the above lemmas, it is not difficult to see that after the level-one bottleneck is resolved, the connections in the level-two bottleneck will always be allowed to ramp up to their max-min rates and stabilize. Furthermore, connections in the level-one bottleneck will remain non-bottlenecked at any level-two bottleneck link(s). Inductively repeating this argument shows that eventually all rates stabilize at their max-min values.

D.3 Convergence Properties

The developments in Sections D.1 and D.2 assume G-consistent groupings at all times. However, this may not necessarily be the case. If the network starts out at a random initial state, it is quite possible for this condition to be violated at one or more links. Furthermore, during transience, G-consistency can be compromised. Specifically, as connections ramp-up, link fair shares generally tend to decrease, and other connections previously classified as non-bottlenecked can now become bottlenecked. However, due to its fast $O(1)$ nature, the rate allocation algorithm will not move such misclassified connections into the bottlenecked state until their

RM cells are received. Hence, loosely speaking, the scheme only maintains G-consistency in a delayed sense. This issue is now explored in more detail.

Consider, again, a single link supporting N connections. Clearly, if the connection groupings are not G-consistent at time t, then there exists a subset of connections which have stored usages above their fair share values, yet are misclassified as non-bottlenecked. Let this subset be denoted as $\overline{\overline{\mathcal{I}}}(t)$, the *inconsistent* set, where $\overline{\overline{\mathcal{I}}}(t) \subseteq \overline{\mathcal{I}}(t)$. Now given the above, the condition in Eq. (D.4) *can* be violated, i.e., it is possible that

$$C_{\overline{\mathcal{I}}}(t) \geq |\overline{\mathcal{I}}(t)| \cdot \frac{\mu}{N}, \tag{D.25}$$

and hence the computed fair share (for bottlenecked connections) is *below* the ideal max-min fair share:

$$ER^i_{sw}(t) = \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} < \frac{\mu - |\overline{\mathcal{I}}(t)| \cdot \frac{\mu}{N}}{|\mathcal{I}(t)|} = \frac{\mu}{N}. \tag{D.26}$$

Lemma D.5

If an RM cell for a connection $i \in \overline{\overline{\mathcal{I}}}(t)$ is received at time t then $\overline{\overline{\mathcal{I}}}(t^+) = \overline{\overline{\mathcal{I}}}(t) \backslash i$ (i.e., connection i satisfies G-consistency at time t^+).

Proof

If connection i is improperly classified as non-bottlenecked, this means that

$$ACR^i_{sw}(t) \geq \frac{\mu - C_{\overline{\mathcal{I}}}(t) + ACR^i_{sw}(t)}{|\mathcal{I}(t)| + 1}. \tag{D.27}$$

Re-arranging the above expression, it can be seen that:

$$ACR^i_{sw}(t) \;\; = \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} + \epsilon_i(t), \tag{D.28}$$

where $\epsilon_i(t) \geq 0$ is the bandwidth *deviation* from G-consistency for connection i. Consider the *new* fair share for connection i, which is computed upon RM cell reception as (i.e., Eq. (3.7), $i \in \overline{\mathcal{I}}$):

$$ER^i_{sw}(t^+) = \frac{\mu - C_{\overline{\mathcal{I}}}(t) + ACR^k_{sw}(t)}{|\mathcal{I}(t)| + 1}. \tag{D.29}$$

Using Eq. (D.28) in the above and re-arranging gives the following:

$$\begin{aligned} ER^i_{sw}(t^+) \;\; &= \frac{\mu - C_{\overline{\mathcal{I}}}(t) + \left(\frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} + \epsilon_i(t) \right)}{|\mathcal{I}(t)|+1} \\ &= \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} + \frac{\epsilon_i(t)}{|\mathcal{I}(t)|+1} \\ &\leq ACR^i_{sw}(t), \end{aligned} \tag{D.30}$$

(since $\epsilon_i(t) \geq 0$, via Eq. (D.28). Now if the received cell is a *backward* RM cell, connection i is moved to the bottlenecked group since its computed fair share is less than its stored value (i.e., $\mathcal{I}(t^+) = \{\mathcal{I}(t), i\}$ since $ER^i_{sw}(t^+) \leq ACR^i_{sw}(t)$). Conversely, if the received cell is a *forward* RM cell, the connection either stays in the (current) non-bottlenecked grouping or moves to the bottlenecked grouping, depending upon the value in the received cell, ACR^i_{cell}. Nevertheless, for connection i to maintain non-bottlenecked status, necessarily $ACR^i_{cell} = ACR^i_{sw}(t^+) \leq ER^i_{sw}(t^+) \leq ACR^i_{sw}(t)$, and therefore connection i must have reduced its rate. In

either case, this implies $\overline{\overline{\mathcal{I}}}(t^+) = \overline{\overline{\mathcal{I}}}(t) \backslash i$. ∎

The above result shows that the algorithm always attempts to improve the consistency status at the link, i.e., *self-stabilizing* property. Now generally, simulations show that the scheme is very robust and rectifies inconsistent groupings in all of the scenarios tried. Nevertheless, analytically capturing this behaviour has proved to be very difficult due to the involved asynchronicities. Specifically, pathological scenarios can be derived where transitions out of the violating set, $\overline{\overline{\mathcal{I}}}(t)$, can induce future transitions into this set. However, the *magnitude* of the deviation from G-consistency, $\epsilon_i(t)$, in such cases is typically a *decreasing* value. To show this informally, consider a connection $i \in \overline{\overline{\mathcal{I}}}(t)$ with a deviation of $\epsilon_i(t)$. When this connection is moved to the bottlenecked state (Lemma D.5) at a time t^+, the $C_{\overline{\mathcal{I}}}(t)$ is adjusted by $ACR^i_{sw}(t)$ (i.e., Eq. (D.3)). Thus the fair share for any *bottlenecked* connection i now increases as:

$$\begin{aligned} ER^i_{sw}(t^+) &= \frac{\mu - C_{\overline{\mathcal{I}}}(t^+)}{|\mathcal{I}(t^+)|} \\ &= \frac{\mu - C_{\overline{\mathcal{I}}}(t) + \left(\frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} + \epsilon_i(t) \right)}{|\mathcal{I}(t)|+1} \\ &= \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} + \frac{\epsilon_i(t)}{|\mathcal{I}(t)|+1} \\ &\geq \frac{\mu - C_{\overline{\mathcal{I}}}(t)}{|\mathcal{I}(t)|} \\ &= ER^i_{sw}(t). \end{aligned} \tag{D.31}$$

From the above, hypothetically speaking, the possibility exists that a currently bottlenecked connection can ramp-up for the newly "freed" $\frac{\epsilon_i(t)}{|\mathcal{I}(t)|+1}$ and become

misclassified at this value (or possibly more) at some future point in time. Nevertheless, usually the magnitude of this potential deviation is considerably reduced. Furthermore, the chances of sustaining such "worst case" behaviour are rare in asynchronous environments, especially if connection sets (and thus $|\mathcal{I}(t)|$) are large, and with the more general case of $\rho < 1$. As a result $|\tilde{\tilde{\mathcal{I}}}(t)|$ typically tends to decrease with time, as the rate allocation algorithm "pushes" inconsistent connections towards G-consistency and decreases their deviations.

Finally, it should be mentioned that faster transient convergence is noted if connections are always guaranteed their *minimum* rate at a link, as per Eq. (3.9), repeated here:

$$ER^i_{cell} = \min(ER^i_{cell}, \max(ER^i_{sw}, ER^i_{min})). \tag{D.32}$$

Clearly, this trivializes the convergence analysis for the single link case (and at the most bottlenecked link in the multiple link case). However, Eq. (D.32) does not have the same effect on higher level bottleneck resolutions. To see this, consider the max-min fair rate for connections in the level-two bottleneck, ν_2, where:

$$\nu_2 > \frac{\mu_2}{|\mathcal{A}_{j^2}|}. \tag{D.33}$$

Eq. (D.32) only guarantees a bottlenecked connection (at link j^2) the right-hand-side of Eq. (D.33), and this is less than its ideal fair share. Thus a more general convergence argument, as presented in Sections D.1 and D.2, is called for.

Appendix E

Non-Oscillatory Condition

The non-oscillatory condition in Eq. (3.17) is derived, assuming constant available capacity levels at all links and either constant or infinite demand (persistent) connections. It is further assumed that the *G-consistency* condition (Eq. (D.5)) holds at all links, and that fixed capacity scaling is done during congestion, regardless of the actual magnitude of (queue) overload, i.e., $\gamma = 0$. Hence the computed fair shares, ER^i_{sw} (Eq. (3.7)), will always be greater than or equal to the ideal fair shares. This allows connections rates to increase and after a sufficient time duration, become sufficiently "close" to their ideal fair share values (i.e., network close to optimally fair operating point). Albeit a relatively stringent pre-condition, the above assumption is necessary in order to simplify the subsequent development. As a result, the actual (time-varying) rate of a connection during transience is simply approximated by its ideal MCR-plus-equal share value at the link (along with other pertinent quantities).

Consider the ideal steady-state fair share for a connection i which is bottlenecked

at link j, r_i, determined using the MCR-plus-equal-share criterion:

$$r_i = MCR^i + \left(\frac{\mu_j - C_{\overline{\mathcal{I}}_j} - C_{\mathcal{I}_j}^{MCR}}{|\mathcal{I}_j|}\right), \quad \forall i \in \mathcal{I}_j, \tag{E.1}$$

where $C_{\overline{\mathcal{I}}_j} = \sum_{k \in \overline{\mathcal{I}}_j} r_k$ and $C_{\mathcal{I}_j}^{MCR} = \sum_{k \in \mathcal{I}_j} MCR^k$ (as per notation defined in Section 3.3.2, *Non-Oscillatory Condition*). Now if link j is congested, its *usable* bandwidth is scaled by the constant $(1 - \beta_j)$, where $\beta_j < 1$ (Eq. (3.16)). Hence the computed fair share with this reduced value will be lower than the desired value in Eq. (E.1), and this can induce transitions between the bottlenecked and non-bottlenecked groupings. It is not difficult to see that by preventing $\mathcal{I}_j \rightarrow \overline{\mathcal{I}}_j$ transitions, oscillations can be avoided.

Consider the computed fair share for connection i with the *scaled* bandwidth value, $(1 - \beta_j)\mu_j$. This value cannot be less than a fraction ρ of that in Eq. (E.1) if transitions to the non-bottlenecked grouping are to be inhibited, i.e.,:

$$MCR^i + \left(\frac{(1 - \beta_j)\mu_j - C_{\overline{\mathcal{I}}_j} - C_{\mathcal{I}_j}^{MCR}}{|\mathcal{I}_j|}\right) \geq \rho r_i, \quad \forall i \in \mathcal{I}_j, \tag{E.2}$$

where $\rho < 1$ and $(1 - \beta_j)\mu_j \geq (C_{\overline{\mathcal{I}}_j} - C_{\mathcal{I}_j}^{MCR})$ by prior bandwidth reservation. Substituting Eq. (E.1) into the above expression, and re-arranging gives:

$$\beta_j \leq (1 - \rho)\left(1 - \frac{C_{\overline{\mathcal{I}}_j} + C_{\mathcal{I}_j}^{MCR} - |\mathcal{I}_j| MCR^i}{\mu_j}\right). \tag{E.3}$$

Thus if β_j is bounded as above, then $\mathcal{I}_j \rightarrow \overline{\mathcal{I}}_j$ transitions for connection i are prevented. Taking the minimum of Eq. (E.3) over all connections at all links gives Eq. (3.17).

Although the assumptions used to derive Eq. (3.17) are very tight, simulation results show that the value does give non-oscillatory behaviour for all of the network scenarios tested (see Chapter 4).

Appendix F

Worst Case Transient Analysis: Single Link

The analytical treatment of flow control schemes in a generic network setting is typically a very difficult problem. Most current analysis is limited to *proportional rate control*-type (PRCA) algorithms using differential equations to model special single link cases [13],[29],[66]-[73],[76]-[78]. Furthermore, important issues such as bandwidth-delay performance (i.e., diverse propagation delays) have not been fully addressed either. The bandwidth delay behaviour of the rate allocation algorithm (Section 3.3.1) is now treated analytically. Specifically, bounds on the worst case transient queueing behaviour at a generic *single* isolated link are derived (as presented in [79]). This is an important issue, and the results can be used for buffer dimensioning purposes to prevent cell losses. Furthermore, the approach is in contrast to most existing analytical treatments which try to derive exact queue buildups for more specific cases.

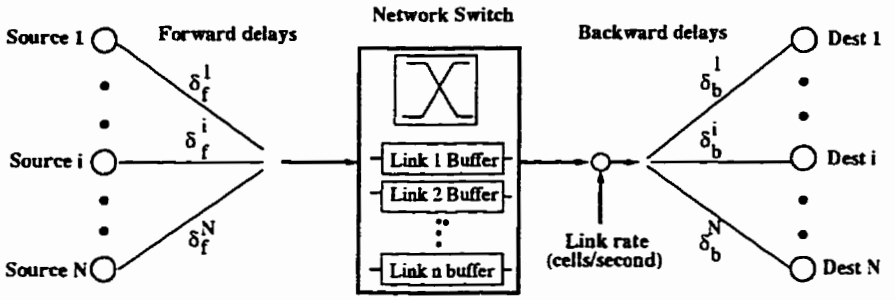


Figure F.1: Single link network, all connections using same link

The model assumes persistent connections with diverse propagation delays and uses fluid-flow analysis techniques. To simplify the development, it is assumed that strictly capacity allocation is done (i.e., $\beta, \gamma = 0$), and that sources perform direct rate adjustments to the received ER value in backwards RM cells (i.e., AIR values very large, $N_{rm} \cdot AIR \gg PCR$, Eq. (3.2)). Note that this also gives worst case transient queueing behaviour. Although attempts have been made to incorporate the reactive congestion detection feature (Section 3.3.2) into the analysis, the large intractabilities pose many problems. As a result, the derived transient bounds (for strictly rate allocation) may be significantly larger than those observed in practice with smaller, more realistic additive increments and reactive congestion control. The analytical modelling is first presented, and then its results compared with those obtained via simulation.

F.1 Analytical Development

Consider N *persistent* ABR connections sharing a link of capacity μ cells/second with *non-uniform* (distinct) *source end-system* and *destination end-system* propa-

gation delays, δ_f^i, δ_b^i, respectively, as shown in Fig. F.1. Since the environment contains only ABR sources, the usable capacity (Eq. (3.11)) will always equal the link rate (i.e., $C_u(k) = \mu$ cells/sec.). For simplicity all connections are assumed to have zero-MCR requirements, although the analysis can be extended to incorporate non-zero MCR guarantees. All connections start at time zero, transmitting at very low initial cell rates ($\ll \frac{\mu}{N}$) by first emitting an RM cell. Hence a connection must wait at least one full roundtrip delay to receive its initial ACR value from the network. Similarly, the switch will not know about a connection's ramp-up until its *second* RM cell is received. Extending the bounds to handle non-simultaneous starting times is also possible and will be briefly described. Since distinct connection delays significantly complicate the analysis, only the express RM cell queueing case is considered. This yields (roughly) bounded inter-RM cell delays which can be used for computing overload period durations.

Let the maximum transient queue buildup induced by connection i be denoted by q_{max}^i. Hence the total transient queue buildup is upper-bounded by:

$$q_{max} = \sum_{i=1}^{N} q_{max}^i. \tag{F.1}$$

Central to the development is the fact that when a given connection ramps up, others which have ramped up before it (i.e., shorter delays) must reduce their rates to the appropriate fractions of link capacity. However, due to the reactive nature of the scheme, there are delays in source rate reduction (i.e., propagation delays, inter-RM cell times, express RM cell buffer waiting times). Therefore, by deriving the rate overload and its duration, bounds on the queue buildups can be computed.

Now, in general, when the link "knows" that k connections have ramped up (i.e., are bottlenecked), the correct rate for all connections is $\frac{\mu}{k}$. Therefore, given the delays, actual source rates can be approximated and, by subtracting their ideal values, the rate overload determined. Similarly, by using the approximated rates, the maximum overload durations can be computed. Details are now presented. All times are in seconds, queue sizes in cells, and rates in cells/second.

Several important quantities of interest are first defined. The roundtrip propagation delay for the i-th connection, δ^i, is defined as (see Fig. F.1):

$$\delta^i = 2\delta_f^i + 2\delta_b^i. \tag{F.2}$$

The *backward notification* time, T_b^i, for connection i is defined as the *minimal* time in which its first RM cell is sent by the switch to the source:

$$T_b^i = 2\delta_b^i + \delta_f^i. \tag{F.3}$$

Similarly, the *forward notification* time, T_f^i, is defined as the *minimal* time in which the switch is *notified* that connection i has ramped up to its initial rate, i.e., is bottlenecked. This occurs after the source receives its first RM cell back, ramps up, and then notifies the switch δ_f^i seconds later:

$$T_f^i = \delta^i + \delta_f^i = 2\delta_b^i + 3\delta_f^i. \tag{F.4}$$

Eq. (F.4) is valid since connection groupings are also updated on *forward* RM cell passes in the rate allocation algorithm (see Fig. 3.4). Note that Eq. (F.4) does not

take into account the waiting time for the second "messenger" RM cell. Rather, this is accounted for in the subsequent analysis. Another quantity of interest is the maximum waiting time for a messenger RM cell given an input rate of r cells/second to the switch, $T_{N_{rm}}(r)$:

$$T_{N_{rm}}(r) = \frac{N_{rm}}{r}. \tag{F.5}$$

Although inter-RM cell times can be expressed by the right-hand-side of Eq. (F.5), this is not done for reasons of clarity.

For large N, the waiting time in the express RM cell buffer can be significant, and this must also be incorporated. To this end, connection i's *express grouping*, $\mathcal{E}_i$, is defined as those connections whose forward propagation delays are within a cell transmission time:

$$\mathcal{E}_i = \left\{ j, 1 \leq j \leq N \ s.t. \ |\delta_f^i - \delta_f^j| \leq \frac{1}{\mu} \right\}. \tag{F.6}$$

Hence the approximate worst case waiting time (for the first RM cell) in the express buffer is $\frac{|\mathcal{E}_i|}{\mu}$ seconds. Note that if there is sufficient diversity in the propagation delays, the case can arise where a shorter delay source can ramp-up at the switch *and* send more RM cells before the forward propagation delay of another connection. Hence, Eq. (F.6) will have to be modified to compute such possibilities. However, since this is very unlikely, it is not considered in the formulation. For later analytical facility, an additional $(N+1)$-th connection with zero-PCR and infinite delay is also defined (i.e., $\delta_b^{N+1}, \delta_f^{N+1} \to \infty$).

Due to persistent sources, the rate behaviour for all connections is monotonically decreasing after the initial ramp-up. This occurs since a connection is classified as bottlenecked after ramping up and the number of connections ramping up only increases with time. As a result the rates can only decrease with time. Without loss of generality, assume that connection indices are ordered by increasing forward notification times (i.e., $T_f^i \leq T_f^j$ for $i < j$). Fig. F.2 depicts the generic behaviour for the connection i input rate at the switch, dividing it into three major stages. The true start-up time for connection i is always greater than the T_f^i shown in Fig. F.2 (due to the inter-RM cell time for the second RM cell). However, since the main concern is bounding maximum queue buildups, setting it to T_f^i is acceptable (pessimistic). After the switch is notified that a source has ramped up, its second RM cell still has to propagate to the destination and then return to the source. So for at least one roundtrip interval the source cannot be controlled (Stage 1, Fig. F.2). Subsequent to this, the RM cell stream is "established" and response times are reduced to roughly $2\delta_f^i$ (plus an appropriate inter-RM cell time). This is the case for Stages 2 and 3 in Fig. F.2, and the difference between these will be elaborated on shortly. Worst case queue buildups are computed for each stage and summed to give the contribution by the connection:

$$q_{max}^i = q_1^i + q_2^i + q_3^i. \tag{F.7}$$

To approximate the initial start-up rate in Stage 1, *delay groupings*, $\mathcal{D}_i$, are defined. The delay grouping for a connection comprises the connection and a minimal subset of connections with *smaller* roundtrip delays whose forward notification

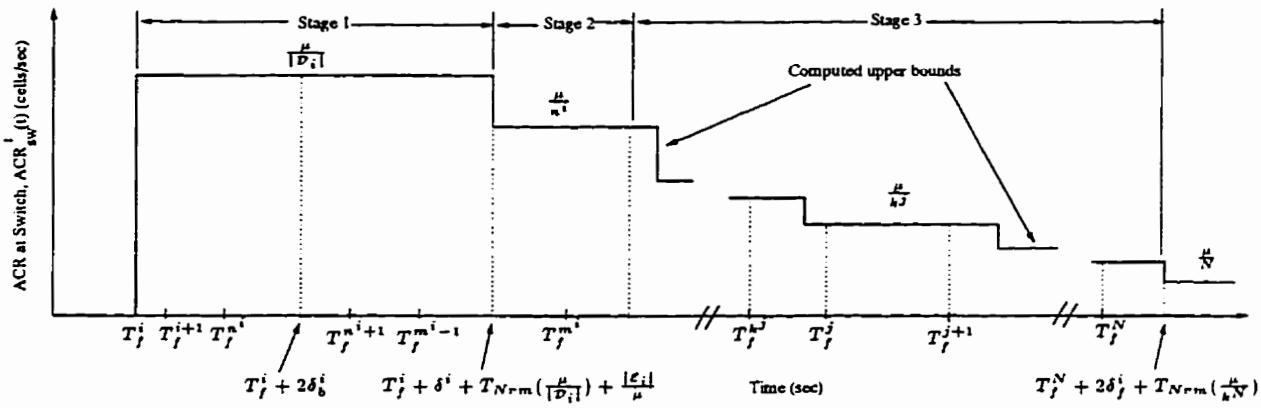


Figure F.2: Generic connection i rate at switch in Fig. F.1, $ACR^i_{sw}(t)$

times are less than the reference's backward notification time, T^i_b. Specifically, connection j is bottlenecked at the switch after its first RM cell waits in the express buffer ($\frac{|\mathcal{E}_j|}{\mu}$ worst case), propagates back to the source (T^j_b), and it generates another RM cell ($T_{Nrm}(\frac{\mu}{j})$ worst case), and then notifies the switch δ^j_f seconds later, i.e., $T^j_f + T_{Nrm}(\frac{\mu}{j}) + \frac{|\mathcal{E}_j|}{\mu}$. Therefore when connection i's first RM cell is sent, the switch "knows" that, at the very least, the connections in its delay grouping have ramped up to their initial rates and are bottlenecked (i.e., $j \neq i$ and $j \in \mathcal{D}_i$ implies $j \in \mathcal{I}$ at T^i_b). Formally, this is expressed as:

$$\mathcal{D}_i = \{i \cup j\} \quad \text{where} \quad \left\{ j, 1 \leq j \leq N \; s.t. \; T^j_f + T_{Nrm}(\frac{\mu}{j}) + \frac{|\mathcal{E}_j|}{\mu} \leq T^i_b \right\}. \tag{F.8}$$

Thus the initial rate for connection i is $\frac{\mu}{|\mathcal{D}_i|}$, since all connections are initially non-bottlenecked (i.e., $i \in \overline{\mathcal{I}}$ at start-up, Section 3.3.1). Since delayed information

is being used, $\mathcal{D}_i \neq \{1, 2, \cdots, i\}$ necessarily, and thus transient over allocation (Section 3.3.1) can occur. Note that Eq. (F.8) is pessimistic since the $T_{N_{rm}}(\frac{\mu}{j})$-term assumes that connection j starts up at its minimum (correct) rate and hence its inter-RM cell waiting time is maximal (Eq. (F.5)). In addition it ignores express RM buffer waiting times for connection i. This means that the computed $\mathcal{D}_i$ sets will be smaller than the actual $\mathcal{I}$ sets at time $T_f^i + T_{N_{rm}}(\frac{\mu}{|\mathcal{D}_i|})$, yielding larger overloads and over-estimating the actual buildup. Using more computations, an iterative procedure can be used to derive more accurate delay groupings.

During Stage 1 in Fig. F.2, other connections can ramp up, increasing the overload magnitude. The minimal index of a connection which ramps up *after* connection i's first stage is:

$$m^i = \min\left\{j, i < j \leq N+1 \quad s.t.\ T_f^j > T_f^i + \delta^i + T_{N_{rm}}(\frac{\mu}{|\mathcal{D}_i|}) + \frac{|\mathcal{E}_i|}{\mu}\right\}. \tag{F.9}$$

Therefore, by using m^i the queue buildup during Stage 1 is determined by summing the (possible) overload resulting from the connection's initial start-up rate and any subsequent connection ramp-ups (i.e., for all j, $i < j \leq m^i - 1$). This is given by:

$$q_1^i = \sum_{j=i}^{m^i-1} \left(\frac{\mu}{|\mathcal{D}_i|} - \frac{\mu}{j}\right) \cdot \left(\min\left(T_f^{j+1}, T_f^i + \delta^i + T_{N_{rm}}(\frac{\mu}{|\mathcal{D}_i|}) + \frac{|\mathcal{E}_i|}{\mu}\right) - T_f^j\right). \tag{F.10}$$

The min-term in Eq. (F.10) ensures that overloads are only summed up to the end of Stage 1.

The queue buildup in Stage 2 is now considered. This stage occurs after the end

of Stage 1, when the source (possibly) reduces its rate. For this consider another index, n^i, defined as:

$$n^i = \max\{j, 1 \le j \le N \quad s.t. \; T_f^j + T_{N_{rm}}(\frac{\mu}{|\mathcal{D}_j|}) + \frac{|\mathcal{E}_j|}{\mu} < T_f^i + 2\delta_b^i\}. \tag{F.11}$$

Specifically, n^i is the maximal index to which connection i will react after its first stage (clearly $n^i < m^i$, Fig. F.2). In the special case where n^i is undefined, it is arbitrarily set to one. This means that at the start of Stage 2, the source rate at the switch will decline to at *least* $\frac{\mu}{n^i}$, and via the definition in Eq. (F.11), this is a very loose bound. Stage 2 ends at the first ramp-up time for a connection after the end of Stage 1, namely $T_f^{m^i}$ (see Fig. F.2). Therefore the ideal rate during the second stage is $\frac{\mu}{m^i - 1}$ and the queue buildup is given by:

$$\begin{aligned} q_2^i = \; & (\tfrac{\mu}{n^i} - \tfrac{\mu}{m^i - 1}) \cdot \Big(\min\Big(2\delta_f^i + T_{N_{rm}}(\tfrac{\mu}{|\mathcal{D}_i|}) \; , \\ & T_f^{m^i} - (T_f^i + \delta^i + T_{N_{rm}}(\tfrac{\mu}{|\mathcal{D}_i|}) + \tfrac{|\mathcal{E}_i|}{\mu})\Big)\Big). \end{aligned} \tag{F.12}$$

Again, the min-term in Eq. (F.12) ensures that the overload is not summed for more than $2\delta_f^i + T_{N_{rm}}(\frac{\mu}{|\mathcal{D}_i|})$ seconds after the end of the first stage, the maximum time before which the source will reduce its rate. In general, though, the buildups in Stage 2 are much smaller than those in the other stages.

The final Stage 3 is now analysed. Consider connection j coming on at T_f^j ($j \ge m^i$, Fig. F.2). The minimum connection index to which source i has responded

to by this time is equal to:

$$k^j = \max\left\{k, m^i \le k < j \quad s.t.\ T_f^k + 2\delta_f^i + T_{N_{rm}}(\frac{\mu}{k-1}) < T_f^j\right\}. \tag{F.13}$$

Note that the $T_{N_{rm}}(\frac{\mu}{k-1})$ term is pessimistic, giving maximum possible inter-RM cell times. Thus the incoming connection rate at the switch at T_f^j is at most $\frac{\mu}{k^j}$, and the rate overload, ε_j, is:

$$\varepsilon_j = \frac{\mu}{k^j} - \frac{\mu}{j}. \tag{F.14}$$

Now this overload can last for either $2\delta_f^i$ (plus an appropriate inter-RM cell time) after which the source reduces its rate, or until the next connection ramps up, T_f^{j+1}, whichever occurs first. This is expressed as the overload duration, τ_j:

$$\tau_j = \min(2\delta_f^i + T_{N_{rm}}(\frac{\mu}{k^j}), T_f^{j+1} - T_f^j). \tag{F.15}$$

By summing overloads for all connection indices after m^i, the queue buildup in Stage 3 is given by:

$$q_3^i = \sum_{j=m^i}^{N} \varepsilon_j \tau_j = \sum_{j=m^i}^{N} \left(\frac{\mu}{k^j} - \frac{\mu}{j}\right) \cdot \left(\min\left(2\delta_f^i + T_{N_{rm}}(\frac{\mu}{k^j}), T_f^{j+1} - T_f^j\right)\right). \tag{F.16}$$

Summing Eqs. (F.10), (F.12), and (F.16) over all connections gives the maximum buildup of Eq. (F.1).

In LAN environments, since the inter-RM cell times are much larger than the

propagation delays, the q_1^i and q_2^i terms will dominate the queue increases. However, in WAN environments propagation delays are larger, and thus the q_3^i term will usually account for most of the queue buildup. Tighter approximations for q_2^i and q_3^i can also be derived by accounting for possible rate decreases between successive forward notification times (at the expense of added computations). Finally, non-simultaneous starting times can be handled by offsetting the T_b^i and T_f^i values appropriately (but not the actual propagation delays themselves, δ_b^i, δ_f^i).

The derived expressions for non-uniform delays can be simplified for the special case of *uniform* delays, where all connections have equal forward, backward, and roundtrip propagation delays, i.e., denoted δ_b, δ_f, δ, respectively. Here, all forward and backward notification times are equal, and therefore the delay groupings contain only one connection, allowing sources to ramp up to the full link rate (i.e., $\mathcal{D}_i = \{i\}$, $|\mathcal{D}_i| = 1$). Also, the express groupings contain all connections, giving worst case RM express buffer delays of $\frac{N}{\mu}$ sec. Furthermore, all the m^i values are equal to $N+1$, Eq. (F.9). Using these values in Eq. (F.16) gives zero q_3^i buildup. Meanwhile the expression for q_1^i, Eq. (F.10), simplifies to:

$$q_1^i = \left(\mu - \frac{\mu}{N}\right)\left(\delta + T_{N_{rm}}(\mu) + \frac{N}{\mu}\right). \tag{F.17}$$

Specifically, since $m^i = N+1$ and $T_f^j = T_f^k \; \forall j,k$, for $j = i, \cdots, (m^i - 2)$ the delay term in Eq. (F.10) is zero. However, since $T_f^{N+1} \to \infty$, for $j = m^i - 1$ the delay term is now $(\delta + T_{N_{rm}}(\mu) + \frac{N}{\mu})$ and the rate overload is $(\mu - \frac{\mu}{N})$, resulting in Eq. (F.17). The expression for q_2^i, however, is more complicated and depends upon the magnitude of δ_b itself. From Eq. (F.11), if the backward propagation delay, δ_b, is

sufficiently large, then $n^i = N$, yielding zero q_2^i. However, if δ_b is very small, then $n^i = 1$ and

$$q_2^i = \left(\mu - \frac{\mu}{N}\right)\left(2\delta_f + T_{N_{rm}}(\mu)\right), \tag{F.18}$$

since the min-term in Eq. (F.12) becomes $(2\delta_f + T_{N_{rm}}(\mu))$. Eq. (F.18) is more applicable to LAN-type delays, i.e., microseconds. The above indicates the complexity in the bandwidth-delay performance of rate-based schemes in general. By summing Eqs. (F.17) and (F.18) over all N connections, a strict upper-bound on the transient buildup for the uniform delay case is formed:

$$\begin{aligned} q_{max} &= (N-1)\mu\left(\delta + 2\delta_f + 2T_{N_{rm}}(\mu) + \frac{N}{\mu}\right) \\ &= (N-1)(\mu(\delta + 2\delta_f) + 2N_{rm} + N). \end{aligned} \tag{F.19}$$

It is clear from the above expression that the queue buildups with direct ramp-ups and no reactive congestion control can be significant, $O(N^2)$.

F.2 Simulation Results

The ABR (and EDERA) parameter settings used in all tests are tabulated in Table F.1. The bounds are tested for various propagation delays with differing connection sets, and results with uniform connection delays are first presented. Specifically, all end-system-to-switch propagation delays are set equal (i.e., $\delta_b = \delta_f$) and varied from LAN ranges of milliseconds to WAN ranges of microseconds. Figs. F.3 and F.4 show simulation results plotted against their analytical bounds for

Table F.1: Simulation parameter settings

Parameter	Setting
Link rates	150 Mbps
N_{rm}	32
AIR	5 Mbps (i.e., $N_{rm} * AIR \gg \mu$)
ICR	200 kbps
β, γ	0
ρ	1.00

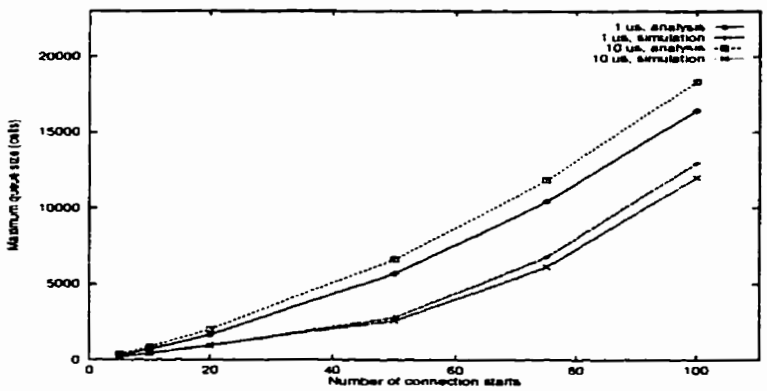


Figure F.3: Uniform delays in Fig. F.1, $\delta_f = \delta_b = 1$ μs, $\delta_f = \delta_b = 10$ μs

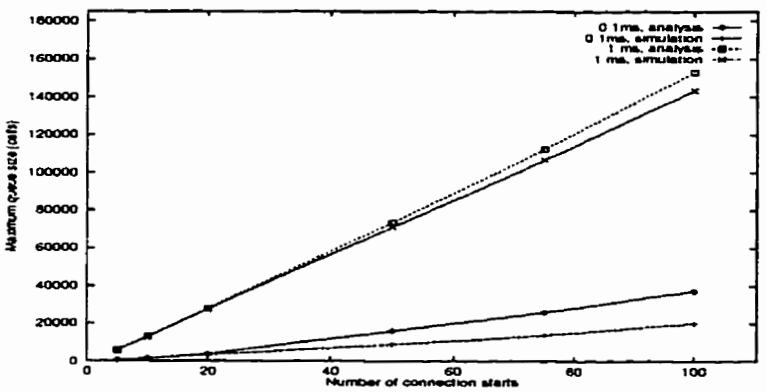


Figure F.4: Uniform delays in Fig. F.1, $\delta_f = \delta_b = 0.1$ ms, $\delta_f = \delta_b = 1$ ms

varying numbers of startup connections. The analysis shows close agreement with the simulation results and indicates that the $O(N^2)$ queueing behaviour is more evident at smaller LAN delays, Fig. F.3. This occurs because the express buffer waiting times are relatively more significant at these delays. The cell transmission time at 150 Mbps is 2.82 μsec, the same order as propagation delays, making the N^2-term in Eq. (F.19) dominate the computed bounds. At larger WAN delays, the linear term corresponding to the propagation delays in Eq. (F.19) dominates the queue build-ups for relatively small N. This can be seen in Fig. F.4, where the queue build-up has a more linear behaviour with respect to the number of connection starts. The results clearly indicate that direct ramp-ups give huge transient queue build-ups. For example, considering end-system delays of 1 ms, for $N = 20$ connections, the maximum queue buildup is about 27,000 cells, well beyond the (approximate) 1000 cells maximum obtained with $AIR = 100$ kbps, Fig. 4.1.

Subsequent tests are done for connection sets with different (non-uniform) propagation delays. Consider a single-switch network with five staggered connections, as shown in Fig. F.5. Fig. F.6 plots the transient queue build-up and the computed upper bound. In this case the computed bound of 3,818 cells is close to the actual value observed in the simulations, 3,770 cells. The rate behaviour at the switch

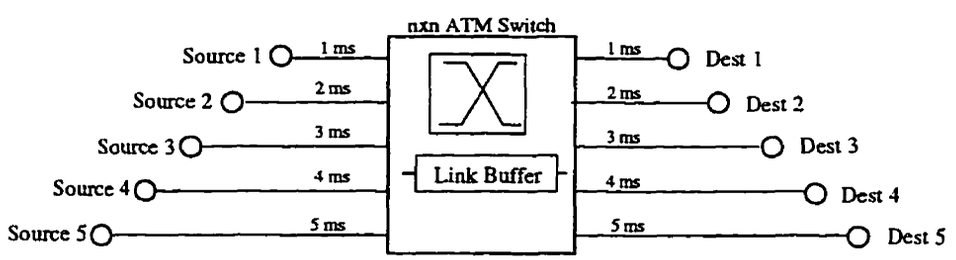


Figure F.5: Single-switch test network with five staggered connections

for connection 1 is depicted in Fig. F.7 and follows the detailed generic form given in Fig. F.2. After the ramp-up of the longest delay connection, connection 5, the connection 1 rate settles down to one-fifth of the link capacity, 30 Mbps.

To test the formulation rigorously, more realistic scenarios are also tried. Specifically, the forward and backward propagation delays are chosen *randomly* and the number of connection starts is varied. Fig. F.8 compares simulations results against the computed bounds for LAN-type delays, where the delays are chosen using a uniform distribution ranging from 1 μs to 10 μs. The bounds show relatively good agreement with the simulation results, even for larger connection sets. The above test case is now repeated with larger propagation delays. For example, delays representative of *metropolitan area networks* (MAN's) are chosen using a uniform distribution from 0.1 ms to 0.5 ms, and the results shown in Fig. F.9. Here, for larger connection sets, the bounds begin to show increasing discrepancies from the simulation results. This is due to the overly-conservative formulation whose effect will be more pronounced at larger delays (i.e., earliest-possible ramp-up times, smaller delay groupings etc.). Finally in order to increase the delay *diversity* even further, the connection delays are chosen using two different distributions. Specifically, half of the delays are chosen from a uniform distribution between 1 μs and 10 μs (i.e., LAN-range), and the other half from a uniform distribution between 1 ms and 5 ms (i.e., WAN-range). This increased delay diversity is representative of switches supporting both LAN and WAN connections. Results for this scenario are shown in Fig. F.10, and again indicate that the computed bounds are significantly larger than the actual buildups for larger connection sets.

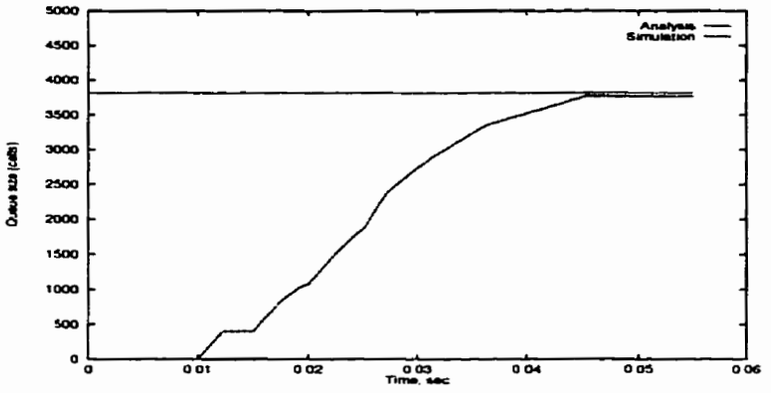


Figure F.6: Queue length for network in Fig. F.5

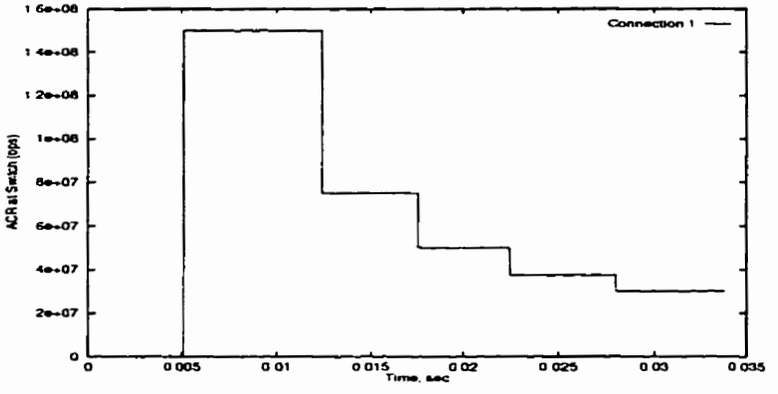


Figure F.7: Connection 1 rate at switch in Fig. F.5, $ACR^1_{sw}(t)$

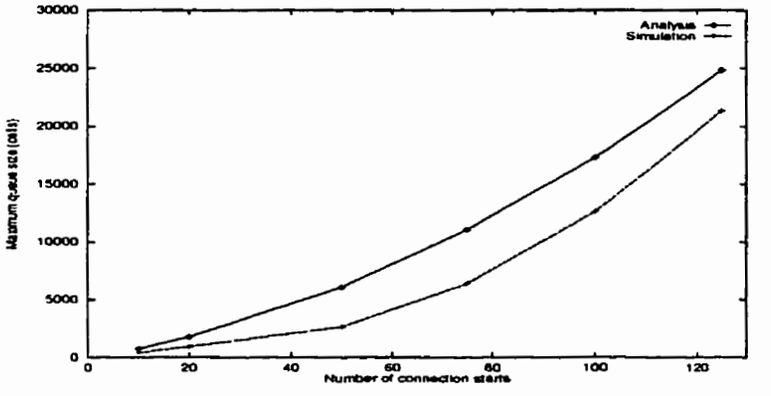


Figure F.8: Non-uniform delays, Fig. F.1, δ_f, δ_b uniform $\sim$ (1 μs, 10 μs)

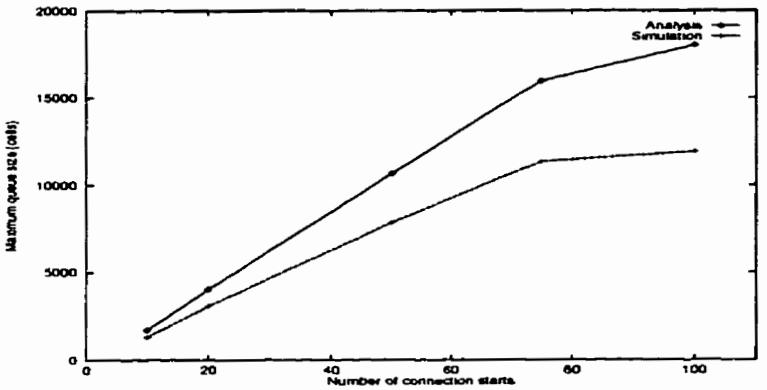


Figure F.9: Non-uniform delays, Fig. F.1, δ_f, δ_b uniform $\sim$ (0.1 ms, 0.5 ms)

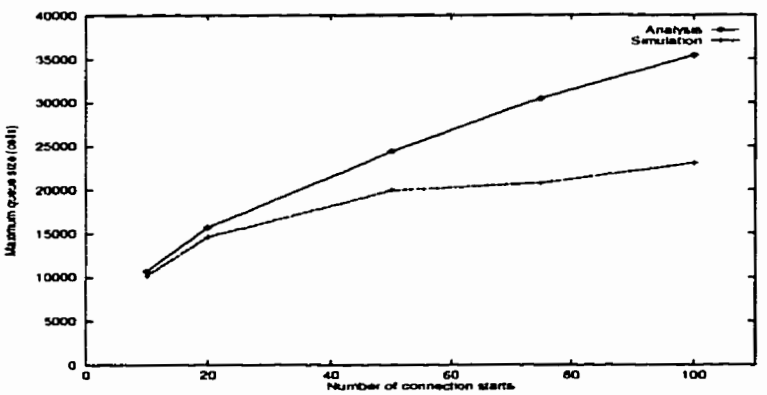


Figure F.10: Non-uniform delays, Fig. F.1, δ_f, δ_b uniform $\sim$ (1 μs, 5 ms)

Overall, the results show that the analytical bounds are relatively good for uniform propagation delays and/or smaller connection sets. However, the source rate envelope, Fig. F.2, is usually not as tight for non-uniform delays, and hence the computed bounds are looser. Nevertheless, the analytical results do bound the queue behaviour in all scenarios tested.

Appendix G

Worst Case Transient Analysis: Multiple Link

Analysis of the end-to-end performance of ABR control schemes has not received much attention. This is because rate behaviour in multiple link networks is much more complicated and generally highly intractable. Due to multiple bottlenecks, links can carry both bottlenecked and non-bottlenecked connections, and thus the rate behaviour (even with simultaneous starts) is generally no longer monotonically decreasing. Nevertheless, looser bounds can be derived to approximately bound the worst case transient queue buildups for simultaneous starts (as in [79]). All of the assumptions from the single link case (Appendix F) are also made here (i.e., direct rate adjustments, strictly capacity allocation, express RM cell queueing). The necessary notation is first introduced and then the analytical development is presented. The bounds are then compared against simulation results for various network scenarios.

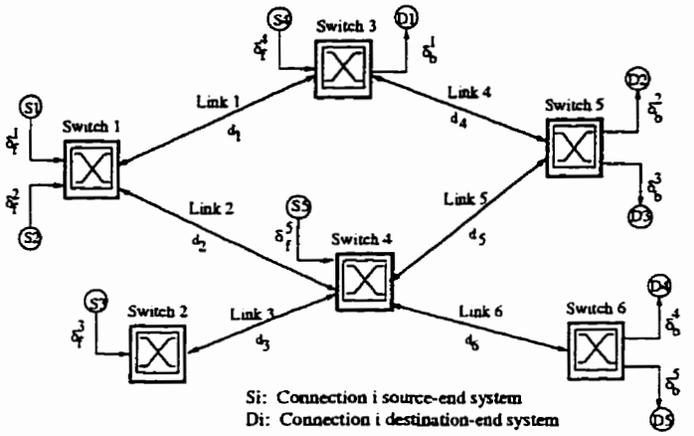


Figure G.1: Sample multiple link network

G.1 Analytical Development

Consider the network modelled as a graph, using the same notation as that presented in Section D.2.1. A sample six-link, five-connection network is shown in Fig. G.1. Link j has capacity μ_j and propagation delay d_j. The end system delays, δ_f^i, δ_b^i, respectively, are defined analogous to the single link case (see Figs. F.1 and G.1). However, the actual forward/backward delays for a connection to the different links along its path vary and these must be approximated. Consider the m-th link of $\mathcal{P}_i$, say link j (i.e., $j \in \mathcal{P}_i$, $i \in \mathcal{A}_j$, and $m \leq |\mathcal{P}_i|$). The forward propagation delay between source i and link j's output buffer is given by

$$\delta_f^{i,j} = \delta_f^i + \sum_{k=1}^{m-1} d_{l_k^i}, \tag{G.1}$$

the backward propagation delay between destination i and link j's output buffer is

$$\delta_b^{i,j} = \delta_b^i + \sum_{k=m}^{|\mathcal{P}_i|} d_{l_k^i}, \tag{G.2}$$

and the roundtrip delay for connection i for all links along its path is given by

$$\delta^i = 2\delta_f^{i,j} + 2\delta_b^{i,j} = 2\delta_f^i + 2\delta_b^i + \sum_{k=1}^{|\mathcal{P}_i|} 2d_{l_k^i}. \tag{G.3}$$

Since delay estimates for multi-link connections are approximate at best, no further attempt is made at accounting for RM express buffer delays (as is done in the single link case, i.e., Eq. (F.6)). Using Eqs. (G.1)-(G.3), the forward notification time for connection i to link j is

$$T_f^{i,j} = \delta^i + \delta_f^{i,j} = 2\delta_b^{i,j} + 3\delta_f^{i,j}. \tag{G.4}$$

Furthermore, after the first roundtrip delay, the *minimum rate* guaranteed to connection i is

$$r_{min}^i = \min_{j \in \mathcal{P}_i} \left\{ \frac{\mu_j}{|\mathcal{A}_j|} \right\}. \tag{G.5}$$

Although, the steady-state allocation, r^i (Section D.2.1), may be larger than r_{min}^i, this cannot be assumed during the initial transient period (i.e., $r_{min}^i \leq r^i$).

Intuitively, it could be postulated that the transient queue buildup at a given link is bounded by that at the equivalent *isolated* single link network (i.e., Fig. F.1 with modified propagation delays, Eqs. (G.1)-(G.2)). However, careful observation

indicates otherwise. Since the single link formulation *implicitly* assumes infinite bandwidth on end systems links, connections can ramp up to any rate dictated by the switch. With multiple links, however, this is not the case. In fact, traversing (steady-state) bottlenecked connections can be *temporarily* bottlenecked elsewhere at lower rates. This technicality breaks the forward notification time ordering between connections (Appendix F), and allows ramp-ups to possibly higher rates, causing larger queue buildups. A sample scenario showing this is presented in the subsequent simulations. As a result the maximum queue buildups can only be bounded by a much looser formulation using *bottleneck resolution* times to constrain the source rate envelope.

Let the bottleneck resolution time for the level-k bottleneck, τ_k, be defined as the maximum time in which connections in $\mathcal{B}_k$ settle to their max-min values. Now, at the most-bottlenecked link, link j^1, after $\max_{i \in \mathcal{A}_{j^1}}\{T_f^{i,j^1}\}$, all connections have ramped up (i.e., are bottlenecked). Hence, the respective resolution time is:

$$\tau_1 = \max_{i \in \mathcal{A}_{j^1}}\{T_f^{i,j^1}\} + \max_{i \in \mathcal{A}_{j^1}}\{\delta^i\} + T_{N_{rm}}(\nu_1), \tag{G.6}$$

where the additional terms account for the maximal response times to the last ramp-up. Higher-level bottleneck resolution times are more difficult to determine due to the transient convergence behaviour of the rate allocation algorithm (Appendix D). As a result, these values are merely approximated by the values in the *global*

max-min rate computation algorithm [3]:

$$\tau_k = \max(\tau_{k-1}, (\max_{i \in \mathcal{A}_{j^k}}\{T_f^{i,j^k}\} + \max_{i \in \mathcal{A}_{j^k}}\{\delta^i + T_{N_{rm}}(r^i_{min})\})) \\ + \quad \max_{i \in (\mathcal{A}_{j^k} \cap \mathcal{B}_k)}\{2\delta_f^{i,j^k} + T_{N_{rm}}(r^i_{min})\}. \tag{G.7}$$

The first term in Eq. (G.7) forces τ_k to be at least as large as the response time for the last connection ramp-up at link j, a necessary condition. However, this term also forces τ_k to be at least as large as τ_{k-1}, which can be pessimistic (i.e., follows global max-min computation in [3]). The second term in Eq. (G.7) accounts for additional response times to the last bottleneck resolution time. Note that Eq. (G.7) does not have any added provisions for possible inter-switch notification delays between bottleneck resolution times. Results show, however, that Eq. (G.7) is usually an extremely pessimistic estimate of the actual resolution times.

Using the above bottleneck resolution times, along with maximum rate overloads, queue buildups can be bounded. For analytical facility, a level-0 bottleneck grouping with $\tau_0 = 0$ and $\nu_0 = 0$ is also defined. Without loss of generality, consider the connections at link j ordered by their $T_f^{i,j}$ values (as in Section F.1). Let k^i be the *highest-level* bottleneck resolved by connection i's forward notification time at link j (i.e., Eq. (G.4), Fig. G.2):

$$k^i = \max\left\{k, 0 \le k < g^i \le b^j \;\; s.t. \;\; \tau_k < T_f^{i,j}\right\}. \tag{G.8}$$

Since $T_f^{i,j}$ is an "early" estimate of the true notification time of connection i at link j, the above expression is a conservative estimate of the actual bottleneck-level

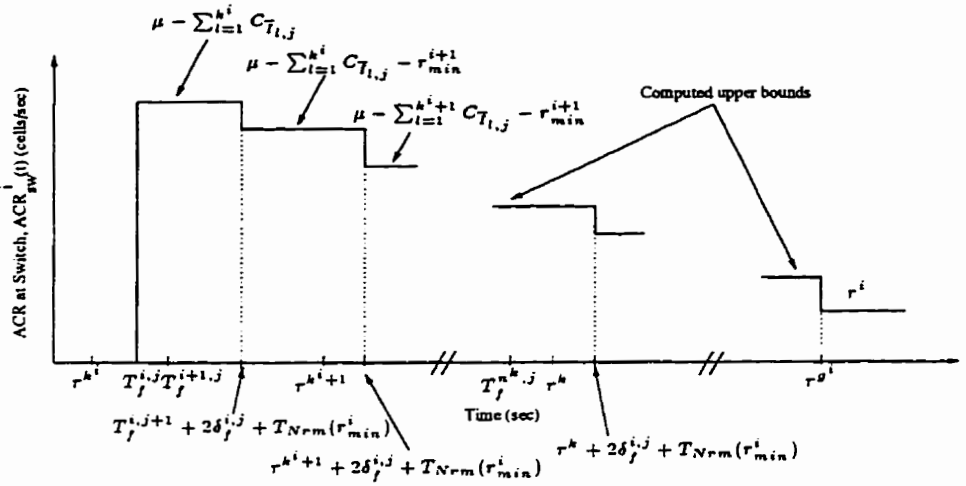


Figure G.2: Generic rate behaviour of connection i at link j

resolved at the true ramp-up time. Furthermore, let the maximum connection index that has ramped up *before* the resolution of the level-k bottleneck be (see Fig. G.2)

$$n^k = \max\left\{n, 1 \leq n \leq |\mathcal{A}_j| \;\; s.t. \;\; T_f^{n,j} < \tau_k\right\}, \tag{G.9}$$

(and $n^k = 1$ otherwise). Now clearly, by time τ_k all connections in $\overline{\mathcal{I}}_{l,j}$, $l \leq k$, have stabilized at their max-min rates. Therefore by time $(\tau_k + T_{Nrm}(r^i_{min}) + 2\delta_f^{i,j})$, connection i's service rate will be upper-bounded by $\mu_j - \sum_{l=1}^{k} C_{\overline{\mathcal{I}}_{l,j}}$ (see Fig. G.2). This is a very pessimistic envelope in that is assumes that a connection will use *all* of the non-resolved bandwidth. Now by the time n connections have ramped up, $T_f^{n,j}$, the worst case minimum rate for connection i to *avoid* overload is $\frac{\mu}{k}$ cells/second (pessimistic). Hence the bound on the maximum buildup induced by connection i at link j is given by summing the total overload in the interval (T_f^i, τ_{g^i}), i.e., up to

the resolution time of connection i's bottleneck level (namely τ_{g^i}):

$$
\begin{aligned}
q_{max}^{i,j} = & \sum_{k=k^i}^{g^i-1} \sum_{n=max(i,n^k)}^{n^{k+1}} \\
& \Bigg(\mu_j - \frac{\mu_j}{n} - \sum_{l=1}^{k} (C_{\overline{\mathcal{I}}_{l,j}} - \sum_{h \in \overline{\mathcal{I}}_{l,j}} r_{min}^h) \cdot \mathrm{I}\{\tau_l + T_{Nrm}(r_{min}^i) + 2\delta_f^{i,j} < \tau_k\} \\
& \qquad - \sum_{m=1}^{|\mathcal{A}_j|} r_{min}^m \cdot \mathrm{I}\{T_f^{m,j} + T_{Nrm}(r_{min}^m) + 2\delta_f^{i,j} < T_f^{n,j}\} \Bigg)^+ \\
& \qquad \cdot \Big(\min(\tau_{k+1}, T_f^{n+1,j}) - \max(\tau_k, T_f^{n,j}) \Big).
\end{aligned}
\tag{G.10}
$$

The first, large bracketed term in Eq. (G.10) bounds the maximum rate overload in a given $T_f^{n,j}$ or τ_k interval (i.e., by removing resolved bottleneck bandwidths and any additional, known minimum rates, Eq. (G.5)). The second bracketed term bounds the duration. The indicator functions are used to counter the asynchronous effects, since the actual ramp-ups may not occur according to the $T_f^{i,j}$ orderings. No doubt the formulation is very pessimistic. The extremely complex rate behaviour in the multiple link case makes it very difficult to formulate a generic, tighter bound on the source rate envelopes. Thus the maximum buildup at link j is given by

$$
q_{max}^j = \sum_{i \in \mathcal{A}_j} q_{max}^{i,j}. \tag{G.11}
$$

G.2 Simulation Results

The analysis is compared with simulation results for several networks. All simulation parameter settings are identical to those used in the single node scenarios, Table F.1. First consider the multiple link network shown in Fig. G.3, with thirty connections traversing three links (all end system-to-switch delays (δ_f^i, δ_b^i) are fixed

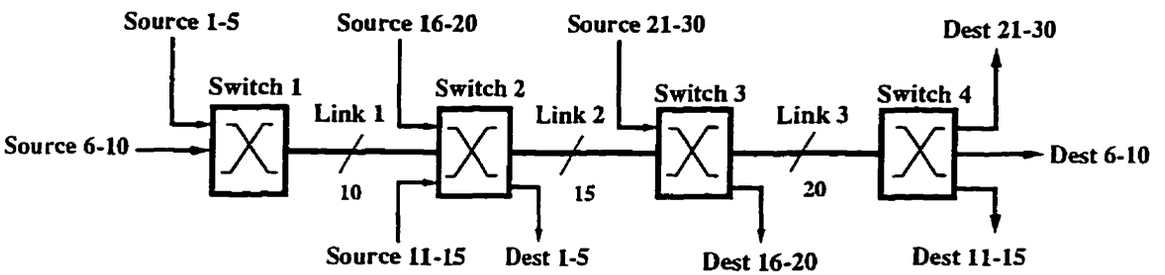


Figure G.3: Multiple link network with 30 connections

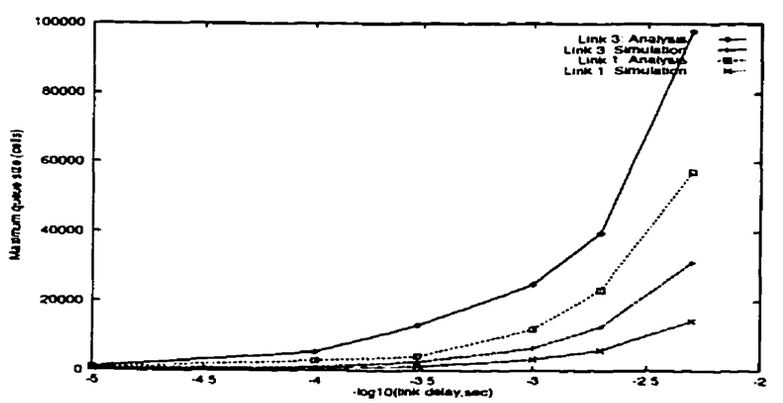


Figure G.4: Queue buildups, increasing link delays, Fig. G.3

at 1 μs). Each link has a different bottleneck level, with the first-level bottleneck being at link 3, supporting 20 connections (i.e., $b^3 = 1$, $\nu_1 = 7.5$ Mbps). Similarly, the second and third-level bottlenecks are at links 2 and 1, supporting 15 and 10 connections each, respectively. The computed bounds are tested for a wide range of *link* delays, ranging from 10 μs (LAN) to 5 ms (WAN). Fig. G.4 plots the bounds at the most and least bottlenecked links, links 3 and 1, respectively, and compares them against the values observed in the simulations. It is clear that the derived bounds significantly over-estimate the true link buildups, especially at larger WAN-like delays (i.e., greater than 1 ms). For example, the predicted values at the most congested link, link 3, are roughly three times the size of the true buildups. In general, Eq. (G.11) is more accurate for lower-level bottleneck links (i.e., more congested) and roughly equivalent propagation delays. Results also indicate that the higher-level bottleneck resolution times (τ_2, τ_3) are greatly over-estimated by Eq. (G.7), worsening the accuracy of the bounds. Furthermore, the true buildups with similar networks and moderate AIR values are roughly an order of magnitude smaller than those observed with *direct* rate adjustments (i.e., 100 kbps, Chapter 4). In other words, gradual ramp-ups yield buildups about *two* orders of magnitude below the derived worst case bounds in Eq. (G.11). Nevertheless, by bounding the worst case behaviour, its is felt that insights into further analysis can be developed.

It was mentioned in Section G.1 that the queue buildups at a given link are not necessarily upper-bounded by corresponding values in an equivalent, isolated link. A sample scenario showing this property is presented in Fig. G.5. There are fourteen connections and three links with moderate MAN-type (*metropolitan*

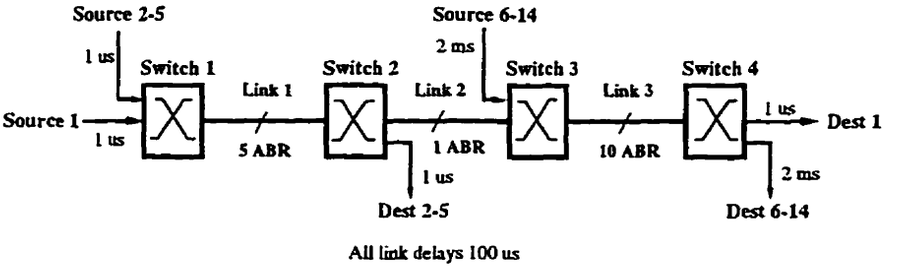


Figure G.5: Multiple link network with 14 connections

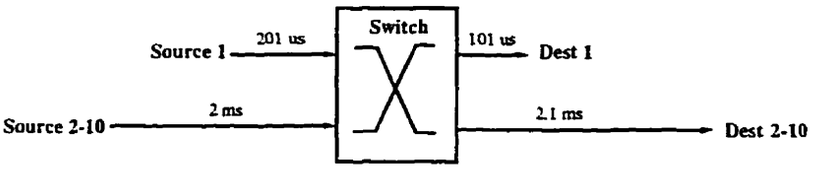


Figure G.6: Equivalent single link network for link 3 in Fig. G.5 (10 connections)

area network) delays of 100 μs. The level-one bottleneck is at link 3, carrying 10 connections with WAN-type roundtrip delays (i.e., connection 1 at 0.6 ms and connections 6-14 at 8.2 ms). The level-two bottleneck is at link 1, carrying 5 connections with much shorter MAN-type roundtrip delays, roughly 200 μs (with the exception of connection 1, slightly larger delays). Now, in particular, consider the link 3 queue, and its equivalent isolated single link representation in Fig. G.6 (i.e., similar to Fig. F.1 with forward and backward delays given by Eq. (G.1) and (G.2) and connections 6-14 re-labelled as connections 2-10).

The maximum queue buildups in both scenarios are graphed in Fig. G.7. The corresponding rate behaviour of sample connections is also show in Fig. G.8 and Fig. G.9. First consider the equivalent single link network in Fig. G.6. Now since connection 1 has significantly shorter response times than connections 2-10, it will ramp-up up for the full link bandwidth of 150 Mbps after about one forward

notification time (i.e., $T_f^{1,3}$, Eq. (G.4)). This means that the delay groupings for connections 2-10 will already contain connection 1, so these connections will ramp up to half the link bandwidth, 75 Mbps, as shown in Fig. G.9. (i.e., $|\mathcal{D}_i| = 2$, $i = 2, ..., 10$, Eq. (F.8)). Now consider the equivalent link 3 in Fig. G.5. Again, connection 1 will ramp up faster than the other connections (connections 6-14) at this link. However, the higher-level bottleneck at link 1 will prevent this connection from ramping up beyond 30 Mbps, since it must share the link 1 bandwidth with connections 2-5. Hence, the longer delay connections at link 3 will be allowed to ramp-up to 120 Mbps, which is significantly larger than the 75 Mbps in the single link case (see Fig. G.8). This results in larger buildups in the actual network, as evidenced by Fig. G.7 (i.e., 18,300 cells versus 10,717 cells). The computed bound at link 3, 22,230 cells (Eq. (G.11)) is also plotted in Fig. G.7. Although it overestimates the true buildups, by about 18%, in this particular case it is much closer than in the previous scenario.

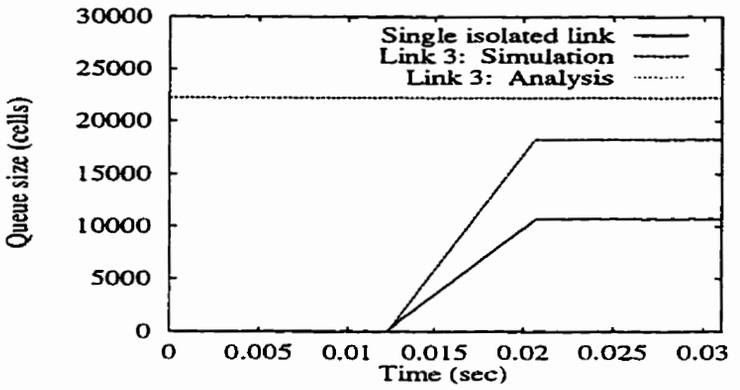


Figure G.7: Queue buildups, link 3 (Fig. G.5) vs. isolated link (Fig. G.6)

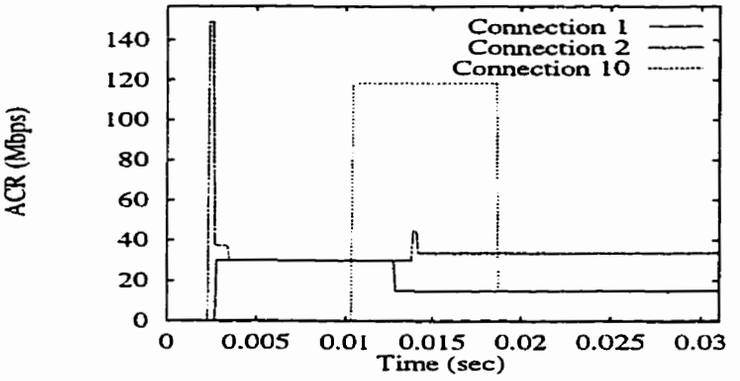


Figure G.8: Sample connection rates in Fig. G.5

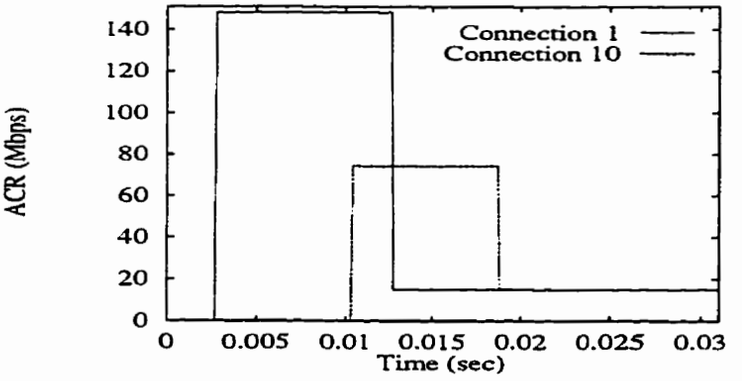


Figure G.9: Sample connection rates in Fig. G.6

Appendix H

Discrete-Event Simulator

A brief overview of the *C++* discrete-event software simulator developed to test flow control and cell scheduling algorithms is given. A simplified network model of the simulator is shown in Fig. H.1. Since an output-buffering model is assumed, network switches are simplified to a series of *slices* connected bi-directionally using separate links. A slice is basically the output buffering portion of a network switch, associated with a given link. Connections are defined as source-destination pairs, and both must be connected to a slice. All major entities (i.e., slices, links, sources, destinations) are written as *C++* class entities with certain core member functions. These classes are defined in a very flexible manner, allowing for a wide variety of extensions to adequately represent the differing source types (i.e., open and closed-loop) and buffering architectures (i.e., FIFO and per-connection buffering).

Each slice contains two sets of *directional* buffers and is connected to two outgoing links, one for each direction. A buffer itself is a class, containing a count of the number of cells, a data structure to hold cells, and cell insertion/deletion

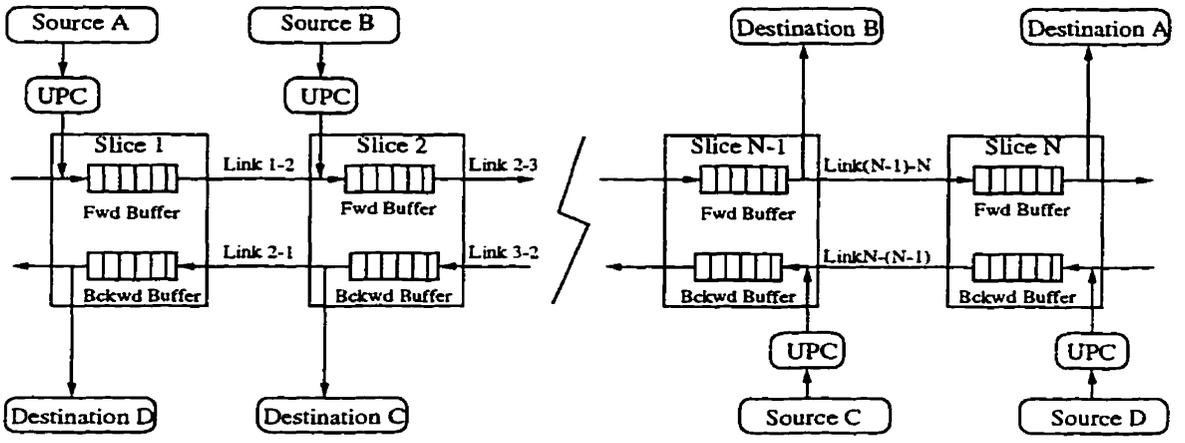


Figure H.1: Overview of network simulator model

functions. The implementation uses a straightforward linked list setup and each buffer performs FIFO queueing. By allowing a slice to contain a set of buffers, more advanced buffer management/service schemes can be added without affecting the overall slice class (i.e., such as per-connection/per-class queuing, buffer sharing strategies, etc.). Meanwhile, the link class (representing network links) has a given rate, two slice endpoints, and a delay function which models its propagation delay. This function chooses a delay based on a given random distribution about a mean link delay (i.e., fixed, uniform, etc.).

The source class represents the network sources (i.e., connection inputs), and has an associated type, either *parametric* or *file stream*. The parametric source type defines the parameters of a source's behaviour, such as persistent, on-off (bursty), exponential, etc. Source cell generation patterns are modelled as generic on-off types, with the user specifying the actual distributions for the various periods and

inter-cell times (in the on period). These distributions can be specified as constant (deterministic), exponential, uniform, and Gaussian. Note that by specifying a deterministic on period along with an off period of zero, a constant-bit rate source can be modelled. The file stream source type, meanwhile, specifies a given input data file containing the inter-cell arrival times. This is used for feeding in video streams into the network from converted MPEG (*moving pictures experts group*) data files. Other parameters for the source include the start and stop times (i.e., late-start/early-stop connections for transient analysis), and a delay function to model delays to the switch (i.e., in the *customer premises network*, CPN). Sources also have counts indicating the number of cells sent and lost. A cell generation function is provided for each source type to generate cells at the proper rate. Furthermore, a leaky bucket class is defined to police the source emissions for all traffic classes. The leaky bucket parameters are specified by the user and each source has an associated leaky bucket. A corresponding destination object is also defined for each source. CPN delays from the last slice of the connection to the destination are modelled for this class as above. Furthermore, a count is also kept for the number of cells received for the respective connection.

An overall network class is declared, consisting of arrays of slices, links, sources, destinations, and a set of core functions. A network function initializes the whole network at startup. This includes declaring network elements (i.e., slices, links and their endpoints, source-destination pairs), constructing a routing table, and starting statistics gathering routines. After initialization another network class function jump starts the sources. This routine generates the first set of events (i.e.

cell generations) to feed the event handler function. Once cells are generated they are propagated along their paths based upon delays and queue lengths. This is done by scheduling future events based upon present events. For example a cell generation event will schedule a future arrival event after the link delay to the respective slice. Once at the slice, the cell arrival event will generate future cell transmission events on the outgoing links of the slice, etc. Each event has a time value associated with it, along with other event-specific information (i.e., source id, hop count, cell type, etc.). Events are kept in a linked list sorted by earliest time, and the event handler basically pulls the head entry from the list (i.e., next closest event) and processes it (see [96, Chapter 8]). This is repeatedly done until either the event list becomes empty or the run time expires. As a result the event handler is the core function which defines the whole functionality of the network, deciding what the events are and how to handle them.

To implement feedback schemes, the basic open-loop source and slice classes are inherited and expanded to add parameters, functions, and events for feedback operation. For example, to implement ABR sources, the above generic source class is enhanced with the required control parameters (N_{rm}, AIR, etc.), and functions added to appropriately increase/decrease the leaky-bucket token generation rate (i.e., spacer-controller, Fig. 1.4). Similarly the slice class is augmented to include utilization thresholds, queue thresholds, and congestion flags for both directions. For closed-loop connections, the destinations also have the ability to filter out RM cells, set the EFCI bits appropriately, and echo RM cells back to the source. Overall, the major changes are absorbed by the network class's event han-

dler routine which essentially controls the operation. For example, to implement the EDERA scheme, additional events are also required such as RM cell arrivals at the source/destination/switch, switch timer interval expirations, source time-outs, etc. In addition, existing event handlers from the basic network need to be changed. For example upon cell arrival at a slice, unlike before, queue levels have to be checked and congestion indicators set appropriately. By concentrating all the major changes to the event types and handlers, and by using class inheritance, various control schemes can be implemented without changing the base classes.

Bibliography

[1] ATM Forum, "Traffic Management Specification Version 4.0," *AF-TM 95-0013R8*, August 1995.

[2] ITU-T, "Traffic Control and Congestion Control in B-ISDN," *Recommendation I.371 Frozen Issue*, Geneva, July 1995.

[3] D. Bertsekas, R. Gallager, *Data Networks, Second Edition*, Prentice Hall Inc., New Jersey, 1992.

[4] R. Onurval, *Asynchronous Transfer Mode Networks: Performance Issues*, Artech House Inc., Boston, MA, 1994.

[5] D. McDysan, D. Spohn, *ATM: Theory and Application*, McGraw Hill, Inc., New York, 1995.

[6] C. Partrige, *Gigabit Networking*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1993.

[7] R. Jain, "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey," *Computer Networks and ISDN Systems*, Vol. 28, No. 13, October 1996, pp. 1723-1738.

[8] R. Jain, "Congestion Control in Computer Networks: Issues and Trends," *IEEE Network Magazine*, May 1990, pp. 24-30.

[9] P. Newman, "Traffic Management for ATM Local Area Networks," *IEEE Communications Magazine*, August 1994, pp. 44-50.

[10] W. Fischer et. al., "Data Communications Using ATM: Architectures, Protocols, and Resource Management," *IEEE Communications Magazine*, Vol. 32, No. 8, August 1994, pp.24-33.

[11] H. Gilbert, O. Aboul-Magd, V. Phung, "Developing a Cohesive Traffic Management Strategy for ATM Networks," *IEEE Communications Magazine*, Vol. 29, No. 10, October 1991, pp. 36-45.

[12] F. Bonomi, K. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service," *IEEE Network Magazine*, Vol. 9, No. 2, March/April 1995, pp. 25-39.

[13] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, H. Miyahara, "Rate-Based Congestion Control for ATM Networks," *Computer Communication Review*, Vol. 25, No. 2, April 1995, pp. 60-72.

[14] A. Arulambalam, X. Chen, N. Ansari, "Allocating Fair Rates for Available Bit Rate Service in ATM Networks," *IEEE Communications Magazine*, Vol. 34, No. 11, November 1996, pp. 92-100.

[15] E. Valencia, L. Benmohamed, R. Nagarajan, S. Chong, "Rate Control Algorithms for the ATM ABR Service," *European Transactions on Telecommunications*, Vol. 8, No. 1, January/February 1997, pp. 7-20.

[16] A. Lin, H. Suzuki, "ABR Open Issues," *AF-TM 95-0575*, June 1995.

[17] H. Fowler, W. Leland "Local Area Network Traffic Characteristics, with Implications for Broadband Network Congestion Management," *IEEE Journal on Selected Areas in Communication*, Vol. 9, No. 7, September 1991, pp. 1139-1149.

[18] A. Romanow, S. Floyd, "Dynamics of TCP Traffic over ATM Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 4, May 1995, pp. 633-641.

[19] R. Simcoe, L. Roberts, "The Great Debate Over ATM Congestion Control," *Data Communications*, September 1994, pp. 75-80.

[20] K. Ramakrishnan, R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks," *ACM Transactions on Computer Systems*, Vol. 8, No. 2, May 1990, pp. 158-181.

[21] H. Kung, R. Morris, "Credit-Based Flow Control for ATM Networks," *IEEE Network*, Vol. 9, No. 2, March/April 1995, pp. 40-48.

[22] H. Kung, "Flow Controlled Virtual Connections Proposal for ATM Traffic Management (Revision R2)," *AF-TM 94-0632R2*, September 1994.

[23] K. Ramakrishnan, P. Newman, "Integration of Rate and Credit Schemes for ATM Flow Control," *IEEE Network*, Vol. 9, No. 2, March/April 1995, pp. 49-56.

[24] R. Jain, S. Kalyanaraman, S. Fahmy, R. Goyal, "Source Behaviour for ATM ABR Traffic Management: An Explanation," *IEEE Communications Magazine*, Vol. 34, No. 11, November 1996, pp. 50-57.

[25] D. Hughes, "Fair Share in the Context of MCR," *AF-TM 94-0977*, October 1994.

[26] N. Yin, "Fairness Definition in ABR Service Model," *AF-TM 94-0928R2*, November 1994.

[27] P. Newman, "Backward Explicit Congestion Notification for ATM Local Area Networks," *Proceedings of IEEE Globecom 1993*, Houston, TX, November 1993, Vol. 2, pp. 719-723.

[28] L. Roberts, "Enhanced PRCA (Proportional Rate-Control Algorithm)," *AF-TM 94-0735R1*, August 1994.

[29] K. Siu, H. Tzeng, "Intelligent Congestion Control for ABR Service in ATM Networks," *Computer Communication Review*, Vol. 24, No. 5, October 1995, pp. 81-106.

[30] A. Barnhart, "Use of the Extended PRCA with Various Switch Mechanisms," *AF-TM 94-0898*, September 1994.

[31] K. Siu, H. Tzeng, "Limits of Performance in Rate-Based Control Schemes," *AF-TM 94-1077*, November 1994.

[32] K. Siu, H. Tzeng, "Adaptive Proportional Rate Control with Intelligent Congestion Control," *AF-TM 94-0888*, September 1994.

[33] E. Gafni, D. Bertsekas, "Dynamic Control of Session Input Rates in Communication Networks," *IEEE Transactions on Automatic Control*, Vol AC-29, No. 11, November 1984, pp. 1009-1016.

[34] J. Jaffe, "Bottleneck Flow Control," *IEEE Transactions on Communications*, Vol. 29, No. 7, July 1981, pp. 954-962.

[35] A. Charny, *An Algorithm for Rate Allocation in a Packet-Switching Network with Feedback*, Master's Degree Thesis, MIT, Department of Electrical Engineering and Computer Science, May 1994.

[36] A. Charny, D. Clark, R. Jain, "Congestion Control with Explicit Rate Indication," *AF-TM 94-0692*, July 1994.

[37] A. Charny, D. Clark, R. Jain, "Congestion Control with Explicit Rate Indication", *Conference Record IEEE International Conference on Communications, ICC'95*, Seattle, WA, June 1995, pp. 1954-1963.

[38] A. Charny, K. Ramakrishnan, "Time-Scale Analysis of Explicit Rate Allocation in ATM Networks", *Proceedings of the IEEE Infocom 1996*, San Francisco, CA, March 1996, Vol. 3, pp.1182-1189.

[39] A. Charny, K. Ramakrishnan, A. Lauck, "Scalability Issues for Distributed Explicit Rate Allocation in ATM Networks", *Proceedings of the IEEE Infocom 1996*, San Francisco, CA, March 1996, Vol. 3, pp. 1198-1205.

[40] A. Charny, K. Ramakrishnan, A. Lauck, "Time-Scale Analysis and Scalability Issues for Explicit Rate Allocation in ATM Networks", *IEEE/ACM Transactions on Networking*, Vol. 4, No. 4, August 1996, pp. 569-581.

[41] A. Charny, K. Ramakrishnan, A. Lauck, "Time-Scale Analysis and Scalability Issues for Explicit Rate Allocation in ATM Networks", *DEC Technical Report, DEC-TR-979*, March 1996.

[42] L. Kalampoukas, A. Varma, K.K. Ramakrishnan, "An Efficient Rate Allocation Algorithm for ATM Networks Providing Max-Min Fairness," *Proceedings of the 6th IFIP International Conference on High Performance Networking, HPN'95*, September 1995.

[43] R. Jain, S. Kalyanaraman, R. Viswanathan, "The OSU Scheme for Congestion Avoidance Using Explicit Rate Indication," *AF-TM 94-0883*, September 1994.

[44] R. Jain, S. Kalyanaraman, R. Viswanathan, R. Goyal, "A Sample Switch Algorithm," *AF-TM 95-0178R1*, February, 1995.

[45] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, F. Lu, "ERICA+: Extensions to the ERICA Switch Algorithm," *ATM Forum-Traffic Management 95-1346*, October 1995.

[46] R. Jain, S. Kalyanaraman, S. Fahmy, R. Viswanathan, "ERICA Switch Algorithm: A Complete Description," *ATM Forum-Traffic Management 96-1172*, August 1996.

[47] S. Muddu, F.M. Chiussi, C. Tryfonas, V.P. Kumar, "Max-Min Rate Control Algorithm for Available Bit Rate Service in ATM Networks," *Proceedings of the IEEE Infocom 1996*, San Francisco, CA, March 1996, Vol. 1, pp. 412-418.

[48] F. Chiussi *et al.*, "Dynamic Max Rate Control Algorithm for Available Bit Rate Service in ATM Networks," *Proceedings of IEEE Globecom 1996*, November 1996, London, UK, Vol. 3, pp. 2108-2117.

[49] H. Kanakia, P. Mishra, A. Reibman, "An Adaptive Congestion Control Scheme for Real Time Packet Video Transport," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 6, December 1995, pp.671-682.

[50] N. Ghani, J. W. Mark, "Dynamic Rate-Based Control Algorithm for ABR Service in ATM Networks," *Proceedings of IEEE Globecom 1996*, November 1996, London, UK, Vol. 2, pp. 1074-1079.

[51] N. Ghani, J. W. Mark, "Measurement-Based Flow Control for ABR Services in ATM Networks," *European Transactions on Telecommunications*, Vol. 8, No. 1, January/February, 1997, pp. 39-53.

[52] N. Ghani, J. W. Mark, "An Enhanced Distributed Explicit Rate Allocation Algorithm for ABR Services," *15th International Teletraffic Congress (ITC 15)*, Washington, D.C., June 1997, Vol. 2b, pp. 1119-1128.

[53] Y. Afek, Y. Mansour, Z. Ostfeld, "Phantom: A Simple and Effective Flow Control Scheme," *Proceedings of the ACM Sigcomm 1996*, Stanford, CA, August 1996, pp. 169-182.

[54] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, H. Miyahara, "Performance Evaluation of Rate-Based Congestion Control Algorithms in Multimedia ATM Networks," *Proceedings of IEEE Globecom 1995*, Singapore, November 1995, Vol. 2, pp. 1243-1248.

[55] R. Jain, S. Fahmy, S. Kalyanaraman, R. Goyal, "ABR Switch Algorithm Testing: A Case Study with ERICA," *ATM Forum-Traffic Management 96-1267*, October 1996.

[56] N. Ghani, J. W. Mark, "Transient Effects in ABR Flow Control," *Conference on Information Sciences and Systems (CISS) 1997*, Baltimore, MD, March 1997.

[57] H. Saito *et al.*, "Performance Issues in Public ABR Service," *IEEE Communications Magazine*, Vol. 34, No. 11, November 1996, pp. 40-48.

[58] S. Keshav, "A Control-Theoretic Approach to Flow Control," *Computer Communication Review*, Vol. 21, No. 4, September 1991, pp. 3-15.

[59] L. Benmohamed, S. Meerkov, "Feedback Control of Congestion in Packet Switching Networks: The Case of a Single Congested Node," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 6, December 1993, pp. 693-708.

[60] D. Cavendish, Y. Oie, M. Murata, H. Miyahara, "Proportional Rate-Based Control Under Long Propagation Delay," *International Journal of Telecommunications*, Vol. 8, May 1995, pp. 79-89.

[61] S. Mascolo, D. Cavendish, M. Gerla, "ATM Rate Based Congestion Control Using a Smith Predictor: An EPRCA Implementation," *Proceedings of the IEEE Globecom 1996*, London, UK, November 1996, pp. 569-576.

[62] Y. Zhao, S. Li, S. Sigarto, "A Linear Dynamic Model for Design of Stable Explicit-Rate ABR Control Schemes," *ATM Forum-Traffic Management 96-0606*, April 1996.

[63] Y. Zhao, S. Q. Li, "Feedback Control of Multiloop ABR Traffic in Presence of CBR/ABR Traffic Transmission," *Conference Record IEEE International Communications Conference, ICC'96*, Dallas, Texas, June 1996, Vol. 3, pp. 1717-1721.

[64] A. Kolarov, G. Ramamurthy, "A Control Theoretic Approach to the Design of Closed Loop Rate Based Flow Control for High Speed ATM Networks," *Proceedings of the IEEE Infocom 1997*, Kobe, Japan, April 1997.

[65] H. Zhang, O. Yang, "Design of Robust Congestion Controllers for ATM Networks," *Proceedings of the IEEE Infocom 1997*, Kobe, Japan, April 1997.

[66] D. Chiu, R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems*, Vol. 17, No 1, June 1989, pp. 1-14.

[67] M. Hluchyj et al., "Closed-Loop Rate-Based Traffic Management," *AF-TM 94-0438R2*, September 1994.

[68] O. Aboul-Magd, "Performance of Link-by-Link Rate Control for ABR Services," *AF-TM 94-0767*, September 1994.

[69] N. Yin, M. Hluchyj, "On Closed-Loop Rate Control for ATM Cell Relay Networks," *Proceedings of the IEEE Infocom 1994*, Toronto, June 1994, Vol. 1, pp. 99-108.

[70] N. Yin, M. Hluchyj, "A Dynamic Rate Control Mechanism for Source Coded Traffic in a Fast Packet Network," *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 7, September 1991, pp. 1005-1012.

[71] A. Gersht, K. Lee "A Congestion Control Framewoork for ATM Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 7, September 1991, pp. 1119-1129.

[72] J. Bolot, U. Shankar, "Dynamical Behaviour of Rate-Based Flow Control Mechanisms," *Computer Communication Review*, Vol. 20, No.2, April 1990, pp. 35-49.

[73] Y. Wang, B. Sengupta, "Performance Analysis of a Feedback Congestion Control Policy," *Proceedings of the ACM SIGCOM 1991*, Zurich, Switzerland, September 1991, pp. 149-157.

[74] A. Kolarov, G. Ramamurthy, "Comparison of Congestion Control Schemes for ABR Service in ATM Local Area Networks," *Proceedings of IEEE Globecom 1994*, San Francisco, CA, Vol. 2, pp. 913-918.

[75] A. Elwalid, "Analysis of Adaptive Rate-Based Congestion Control for High-Speed Wide-Area Networks," *Proceedings of the IEEE International Conference on Communications*, June 1995, pp. 1948-1953.

[76] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, H. Miyahara, "Analysis of Rate-Based Congestion Control Algorithms for ATM Networks Part 1: Steady-State Analysis," *Proceedings of IEEE Globecom 1995*, Singapore, November 1995, Vol. 1, pp. 296-303.

[77] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, H. Miyahara, "Analysis of Rate-Based Congestion Control Algorithms for ATM Networks Part 2: Initial Transient State Analysis" *Proceedings of IEEE Globecom 1995*, Singapore, November 1995, Vol. 2, pp. 296-303.

[78] M. Ritter, "Network Buffering Requirements of the Rate-Based Control Mechanism for ABR Service," *Proceedings of the IEEE Infocom 1996*, San Francisco, CA, March 1996, Vol. 3, pp. 1190-1197.

[79] N. Ghani, J. W. Mark, "Queueing Analysis of a Distributed Explicit Rate Allocation Algorithm for ABR Services," *Proceedings of the IEEE Infocom 1997*, Kobe, Japan, April 1997.

[80] F. Bonomi, D. Mitra, J. Seery, "Adaptive Algorithm for Feedback-Based Flow Control in High Speed, Wide-Area ATM Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 7, September 1995, pp. 1267-83.

[81] S. Floyd, V. Jacobsen, "Link-Sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, August 1995, pp. 365-386.

[82] J. Nagle, "On Packet Switches with Infinite Storage," *IEEE Transactions on Communications*, Vol. 35, April 1987, pp. 435-438.

[83] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE*, Vol. 83, No. 10, October 1995, pp. 1374-1396.

[84] G. Kesidis, *ATM Network Performance*, Kluwer Academic Press, Boston, MA, 1996.

[85] P. Goyal, H. Vin, "Generalized Guaranteed Rate Scheduling Algorithms: A Framework," *Technical Report TR-95-30*, Department of Computer Sciences, University of Texas, Austin, 1995.

[86] D. Ferrari, D. Verma, "A scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, April 1990, pp. 368-379.

[87] A. Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proceedings of the ACM Sigcomm 1989*, Austin, TX, September 1989, pp. 1-12.

[88] A. Greenberg, N. Madras, "How Fair is Fair Queueing," *The Journal of the ACM*, Vol. 39, No. 3, July 1992, pp. 568-598.

[89] A. Parekh, R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case," *IEEE/ACM Trans. on Networking*, Vol. 1, No. 3, June 1993, pp. 344-57.

[90] J. Roberts, "Virtual Spacing for Flexible Traffic Control," *International Journal of Communication Systems*, Vol. 7, No. 4, October-December 1994, pp. 307-318.

[91] S. Golestani, "A Self-Clocked Fair Queueing Scheme for Broad-band Applications," *Proceedings of the IEEE Infocom 1994*, Toronto, Canada, April 1994, Vol. 2, pp. 636-646.

[92] J. Bennett, H. Zhang, "WF^2Q: Worst-Case Fair Weighted Fair Queueing," *Proceedings of the IEEE Infocom 1996*, San Francisco, CA, Vol. 1, pp. 120-128.

[93] J. Rexford, A. Greenberg, F. Bonomi, "Hardware-Efficient Fair Queueing Architectures for High-Speed Networks," *Proceedings of the IEEE Infocom 1996*, San Francisco, CA, CA, Vol. 2, pp. 638-646.

[94] P. Goyal, H. Vin, H. Cheng, "Start-time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," *Proceedings of the ACM Sigcomm 1996*, Stanford, CA, August 1996, pp. 143-156.

[95] K. Lee, "Performance Bounds in Communications Networks with Variable-Rate Links," *Proceedings of the ACM Sigcommm 1995*, August 1995, pp. 126-136.

[96] U. Pooch, J. Wall, *Discrete Event Simulation: A Practical Approach*, CRC Press, Boca Raton, FL, 1993.