THE UNIVERSITY OF CALGARY

Effects of Per-VC Queueing and Buffer Management on TCP over ATM Performance

by

Jiayun Zhu

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

December, 1999

Abstract

The performance of Transmission Control Protocol (TCP) over Asynchronous Transfer Mode (ATM) networks is studied in this thesis. TCP is a connection-oriented transport layer protocol designed to provide reliable data transmission between two hosts. ATM is a connection-oriented network technology.

As the performance of TCP over an ATM network can be poor when the network experiences congestion, this thesis examines several congestion control mechanisms. The mechanisms studied include per Virtual Connection (per-VC) queueing, buffer management, and packet-level discard. A simulation model for these mechanisms has been developed to study the effects of per-VC queueing, buffer management, and Early Packet Discard (EPD) on TCP over ATM performance. The simulation results show that per-VC queueing, EPD, and buffer management can improve the TCP over ATM performance under certain circumstances. Finally, the thesis provides some suggestions to improve the performance of TCP over ATM.

Acknowledgements

I would like to thank my supervisor, Dr. Brian Unger, for his continuous support, guidance, and care throughout the whole duration of my stay at the University of Calgary. I thank him for giving me the great opportunity to further my study at the University of Calgary. He also made an enormous contribution in bringing this thesis to completion. I am very grateful to Dr. Carey Williamson who provided me with suggestions, insights, and continual encouragement. I would not have finished this thesis without his help.

I appreciate the assistance I received from Zhonge Xiao. I had many valuable discussions with him and would like to thank him for the time he spent for me. I respect his serious attitude towards research.

Special thanks to Rob Simmonds, a very optimistic and smart person in the Distributed Systems Lab. He is always willing to help. I have learned a lot from him, including research methods, usage of LaTeX, content of the thesis, and most importantly an optimistic way of thinking.

I would also like to thank Mark Fox for his patient and kind help in my early days in the Distributed Systems Lab. I will never forget it is him who guided me to write my first simulation program.

I am indebted to many colleagues and friends for their help. Thanks are extended to the faculty, staff, and fellow graduates in the Department of Computer Science for their assistance and support.

This thesis is dedicated to my dear parents and my beloved brother.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A protocol is a set of rules that two entities must follow to communicate with each other. Transmission Control Protocol (TCP) is a connection-oriented transport layer protocol designed to provide reliable data transmission between two hosts [39]. Internet Protocol (IP) is a connectionless network layer protocol that provides unreliable data transmission. TCP and IP are part of the TCP/IP protocol suite, which is widely used across many network environments including the Internet. Many Internet applications, such as Telnet, File Transfer Protocol (FTP), electronic mail, and World Wide Web (WWW), are TCP/IP-based.

Many underlying network technologies can be used to support TCP/IP. Asynchronous Transfer Mode (ATM) is one of them [44]. It is the new generation networking technology for multimedia communications, and it is a general-purpose, connection-oriented transfer mode for a wide range of services, including text, voice, video and images. It has been chosen by CCITT as the transfer mode for the Broadband Integrated Services Digital Network (B-ISDN), an all-purpose digital network intended to provide diverse services [32, 33, 35, 39]. Each service can be characterized by a quality of service (QoS) level. The ATM service categories include two real-time categories — Constant Bit Rate (CBR) and real-time Variable Bit Rate (rt-VBR), and three non-real-time categories — non-real-time Variable Bit Rate (nrt-VBR), Available Bit Rate (ABR), and Unspecified Bit Rate (UBR) [15].

However, the performance of TCP over IP over ATM (called TCP over ATM in

this thesis) can be poor if an ATM network experiences congestion and loses packets (cells) due to buffer overflow. This was shown in several recent studies [5, 24, 36, 41, 45].

This thesis studies the performance of TCP over ATM. We examine several schemes proposed to improve the performance of TCP over ATM, including per-virtual connection (per-VC) queueing, buffer management, and packet-level discard. To accomplish this, simulation is used. First, a simulation model for the different schemes is built. After validating the model, we run simulations, gather simulation results, analyze performance of different schemes, and propose ways to improve the performance TCP over ATM.

1.1 The Asynchronous Transfer Mode (ATM) Protocol

ATM is a connection-oriented technology. A connection, called virtual channel or virtual circuit (VC), must be established before any data can be transmitted. In ATM, user information is transmitted over virtual channels using fixed-size packets, called cells. A cell is 53 bytes, comprised of a 5-byte header and 48-byte payload. The header contains the VC, cell type, priority, and control information.

The B-ISDN ATM reference model consists of three layers: the physical layer, the ATM layer and the ATM Adaptation Layer (AAL). The physical layer is at the bottom. It is concerned with transporting raw bits over a physical medium. This layer presents a uniform interface to the upper ATM layer.

The ATM layer deals with cells and cell transport. It provides cell multiplexing, demultiplexing, and routing functions using the Virtual Path Identifier (VPI) and

the Virtual Channel Identifier (VCI) fields of the cell header. It can also perform generic flow control which ensures that connections stay within the limits negotiated at the call establishment phase [39].

The layer above the ATM layer is the AAL. It segments large data units into cells, transmits the cells, and reassembles them at the destination. A number of AAL protocols are available to provide a wide range of traffic carried over the ATM networks, such as CBR traffic and VBR traffic. Among them, the most commonly used AAL for TCP is AAL5 [1].

In AAL5, a large data unit (e.g., an IP packet) is padded and an 8-byte trailer is added. After that, it is divided into 48-byte pieces and sent in ATM cells. The Payload Type (PT) in each ATM cell header indicates which cell is the last cell of the data unit. At the receiver, the cells from a data unit are buffered until the last cell of the data unit is received. If the data unit is intact, it is passed to the higher protocol layer [23].

1.2 The Internet Protocol (IP)

IP provides an unreliable, connectionless datagram delivery service [42]. A datagram is a packet (i.e., a chunk of data sent across a network as a single unit) that contains enough information to allow itself to be delivered to the destination. IP packets are datagrams because they contain the global address of the destination [32]. Unreliable means that there is no guarantee that an IP packet successfully gets to its destination. Thus, reliability is provided by the upper layers (e.g., TCP). Connectionless means IP packets may take different paths or routes and thus a sequence of IP packets may

reach the destination in a different order than they were sent.

The size of the longest IP packet that can be transmitted on a given link is called the Maximum Transmission Unit (MTU). TCP segments that are longer than the MTU are broken into several packets to be transmitted over the IP layer. This is called fragmentation.

1.3 The Transmission Control Protocol (TCP)

TCP is a connection-oriented protocol that provides a reliable byte stream transmission between two hosts. The data unit in TCP is the segment, which is sometimes called a packet. A connection must be established between two TCP applications before they can exchange data. Reliable delivery is achieved through the use of a variety of mechanisms, such as sequence numbers, checksums, sliding window flow control, timeout and retransmission [23].

The size of the longest TCP segment allowed is called the Maximum Segment Size (MSS), which is usually set to the MTU minus the size of the TCP/IP headers (i.e., 20 bytes for the TCP header and 20 bytes for the IP header). TCP makes all segments equal to the MSS if possible [23].

1.4 TCP over ATM

Most networks can be organized as a series of logical layers. A list of protocols used by a certain network, one protocol per layer, is called a protocol stack [44]. An example of a TCP over ATM protocol stack is shown in Figure 1.1 [23]. Each protocol layer offers services to the layer above, and uses services provided by the layer below.

Applications use a standard application programming interface, the socket interface, to communicate with the TCP layer. The socket interface includes facilities for making connections and reading, writing and buffering data. The application writes data to the socket interface to be sent to the TCP layer, and reads data received from the socket interface, which in turn receives it from the TCP layer. TCP/IP provides reliable data transmission over the underlying network interface. The lowest layers, ATM and AAL, are used to prepare data for transmission over an ATM network.

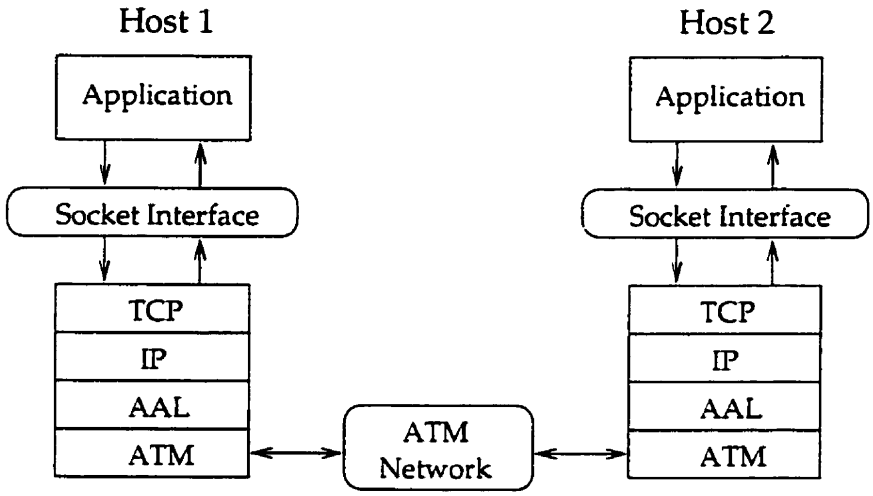

Figure 1.1: TCP over ATM Protocol Stack

A diagram of data processing of TCP over ATM is given in Figure 1.2. At the source, application data is sent to the TCP layer through the socket interface. TCP and IP segment the data into TCP/IP packets. These packets are further segmented into ATM cells. The last cell of a packet contains some padding and the AAL5 trailer. At the destination, ATM cells are reassembled into TCP/IP packets, which are further reassembled into the application data.

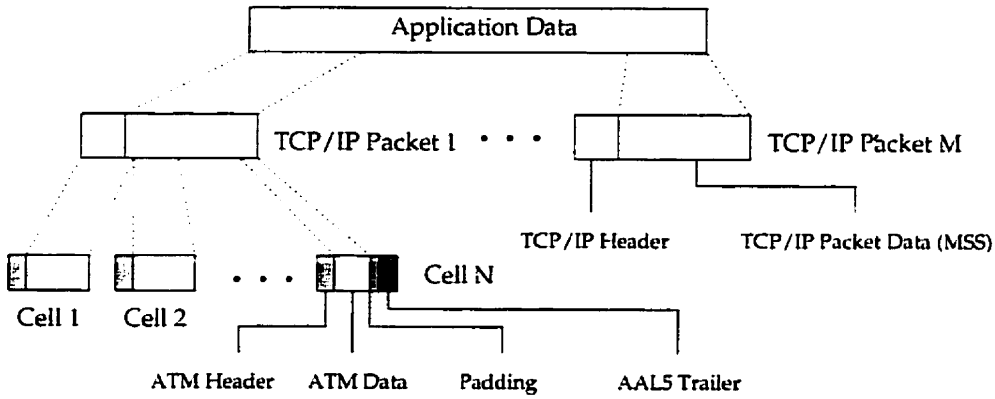

Figure 1.2: Data Processing of TCP over ATM using AAL5

1.5 Simulation

Computer simulation enables one to model and study the behavior of a system without requiring access to the actual physical system. It is a valuable tool for the design and analysis of complex systems. Simulation can involve three stages: model design, model execution, and model analysis [14]. First, a simulation model is built. The model should be small enough so that it only characterizes the aspects that affect the system behavior under investigation. At the same time, the model should have enough detail so that some valid conclusions may be drawn about some aspect of the real system. After the model has been designed, a computer program for that model is written and executed. Many programming languages that have been designed to support simulation can be used to write the program. This phase also includes model verification and validation. The verification is to assure the computer program is performing properly, and validation is to ensure the physical system is represented accurately enough by the simulation model. Then, the information received from

the model execution can be used to analyze the system. The advantages and disadvantages of the system may be examined, and ways may be found to improve the system.

Systems can be categorized as discrete or continuous. A discrete system is a system in which the state changes only at discrete points in time [3]. The system state is a collection of variables containing the information necessary to describe the system at any given time. Discrete Event Simulation (DES) can be used to model discrete or continuous systems. An event is an instantaneous occurrence that may change the state of the system. In DES, the simulation model jumps from one state to another upon the occurrence of an event [16].

Many DESs use the "logical process" modeling view [8]. In the logical process modeling view, the system being modeled is called the "physical system". It is composed of a number of Physical Processes (PPs) that interact at various points in simulated time [16]. In a simulator, PPs are represented by Logical Processes (LPs). All the interactions between PPs are modeled by events sent between the corresponding LPs. Each event contains a timestamp specifying when the event should occur in the receiving LP. Together the collection of LPs, described as the logical system, models the physical system.

1.6 Research Motivation and Objectives

One major factor that affects network performance is congestion. Congestion occurs when the resources (e.g., buffers and link capacity) needed are more than the resources provided. Congestion degrades the performance of a network in the form of

long end-to-end transmission delay, data loss, and low throughput. Throughput is the total number of bytes delivered to the destination divided by the time duration. Figure 1.3 illustrates the impact of congestion. Before the network is overloaded, throughput increases when the load increases. Throughput can drop sharply when the network becomes overloaded. When the network is heavily overloaded, throughput may be close to zero.

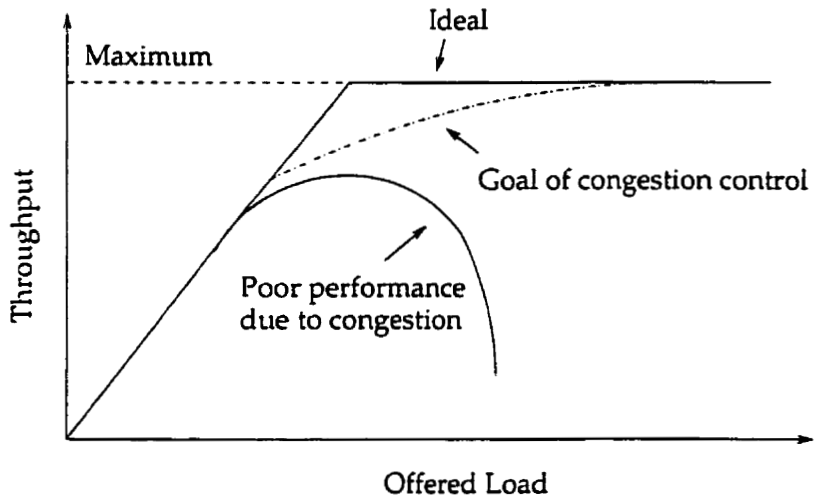

Figure 1.3: Network Congestion

A good system design should respond to congestion. It can either prevent congestion before congestion occurs or reduce congestion after congestion is detected. As seen in Figure 1.3, the goal of congestion control is to prevent or reduce congestion to improve throughput by regulating the traffic. Many congestion control algorithms have been proposed to improve the performance of ATM networks, such as traffic shaping, call admission control, dynamic routing, per Virtual Connection (per-VC) queueing, and buffer management [48].

Traffic shaping regulates the average rate of data transmission to manage congestion. Call admission control controls the number of connections allowed in the network to control congestion. Dynamic routing dynamically selects routes to avoid congestion. Per-VC queueing provides different queues to different VCs to prevent a greedy traffic source from degrading service provided to other sources. Buffer management decides if arriving packets should be allowed entry into the buffer to prohibit a greedy user from taking up all the buffers.

TCP performs well in many network environments. However, the performance of TCP over ATM can be poor. This is mainly caused by fragmentation. TCP segments are broken into cells for transmission over the ATM network; these cells are reassembled into TCP segments at the destination. In an ATM network, some cells from a segment may be dropped due to congestion, while the remaining cells from the same segment are transmitted. The destination cannot reconstruct the segment without the missing cells and thus the whole segment is dropped. Therefore, some bandwidth is wasted to transmit the useless cells, and throughput is low.

To maximize the throughput of TCP over ATM, some packet-level discard algorithms were proposed, including Partial Packet Discard (PPD) [41], Early Packet Discard (EPD) [41], and Early Selective Packet Discard (ESPD) [9]. These algorithms attempt to eliminate packets with incomplete cell counts so as to maximize the total number of complete packets. PPD discards the tail part of packets that have missing cells; EPD attempts to discard entire packets before buffer overflow occurs; and ESPD drops packets from highly active sessions prior to buffer overflow.

The objectives of this thesis are:

1. To build a simulation model for several congestion control algorithms, including per-VC queueing, its relevant buffer management and fair queueing schemes, and packet-level discard schemes.

2. To evaluate the performance of the above schemes with TCP over ATM. In particular, we examine the effect of EPD, the effect of per-VC queueing, the effect of EPD and per-VC queueing, and the effect of buffer management.

3. To suggest ways to improve the performance of TCP over ATM.

1.7 Thesis Outline

The remainder of this thesis is organized as follows.

Chapter 2 reviews the literature related to several congestion control mechanisms, including per-VC queueing, fair queueing, buffer management, and packet-level discard. It also defines the specific problems addressed in the thesis.

Chapter 3 provides the design and implementation of the simulation model in detail. It includes a description of the simulation package—SimKit, an overview of the simulator—ATM-TN, and design and implementation issues of per-VC queueing, fair queueing, buffer management and packet-level discard.

In Chapter 4, the validation of the simulation model is presented. Per-VC queueing and fair queueing schemes are validated both qualitatively and quantitatively.

Chapter 5 contains performance studies and analysis of the simulation model. Simulation scenarios, along with performance metrics, are presented. Simulation results are presented. This chapter studies the effect of EPD, the effect of per-VC

queueing, the effect of EPD and per-VC queueing, and the effect of buffer management, respectively. Several conclusions drawn from the results are presented.

Chapter 6 concludes the thesis with a list of conclusions and suggestions. It also presents interesting areas for further investigation.

Chapter 2

Review of Related Work

In this chapter, we provide a survey of the literature related to several congestion control mechanisms including per-VC queueing, fair queueing, buffer management and packet-level discard. This chapter is organized as follows. We first describe per-VC queueing and several fair queueing algorithms used in the per-VC queueing environment. We then present a simple buffer management scheme. After that we provide an overview of several packet-level discard schemes, including Partial Packet Discard (PPD), Early Packet Discard (EPD), and Early Selective Packet Discard (ESPD). We then define the issues that will be addressed in this thesis.

2.1 Per-VC Queueing and Fair Queueing Schemes

Many computer networks are packet switched. In a packet switching network, user data are segmented into fixed size or variable size units called packets. These packets are then transmitted in a store-and-forward [44] manner across the network. ATM networks are packet switching networks where all packets are of a fixed length.

There are two places where congestion control can be deployed in a packet switching network. Congestion can be controlled at the source by varying the rate at which the source sends packets. Congestion can also be controlled at the switch through routing and queueing algorithms. A switch is a device that stores and forwards packets in a packet switching network. Queueing algorithms (also called schedul-

ing algorithms) determine the order in which packets are sent and the usage of the switch's buffer space [13].

An output queueing switch is shown in Figure 2.1. This switch has a switch fabric and several input ports, output ports, and output queues. The switch ports provide the physical connection to other network entities (e.g., switches). The switch fabric is responsible for transferring packets from an input port to the correct output port. Packets from different input ports may contend for the same output port and may need to be buffered in the output queue. A simple queueing algorithm that can be used for the output queue is first-in-first-out (FIFO). The arrival order of the packets determines which packet will be transmitted next, when a packet will be transmitted, and which packet will be discarded. When a packet arrives at an output port and the output port is busy transmitting a packet, the arriving packet will be put on the end of the queue. When the output port becomes idle, the switch will get the first packet out of the queue and transmit it. When the queue is full, any arriving packet will be discarded.

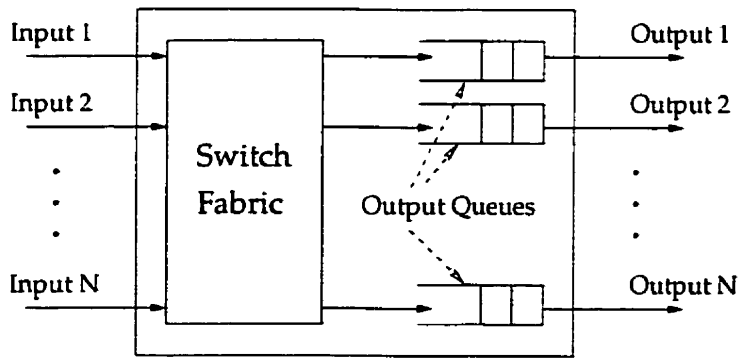

Figure 2.1: An Output Queueing Switch

The FIFO buffer scheme is simple to implement. However, there are several problems with this scheme, including unfair allocation of bandwidth, longer delay for sources using less than their full share of bandwidth, and lack of protection from ill-behaved sources [13]. Different traffic sources tend to affect each other since they are merged into one FIFO queue at an output port. A traffic source sending packets at a high rate can get a higher fraction of the bandwidth. Consider two different traffic sources passing through the same output port of a switch. If the first source has a lot of packets to send and the second source has only a few packets to send, the packets from the second source may still need to wait for a long delay to be transmitted. Some ill-behaved sources may take up all the bandwidth, preventing well-behaved sources from getting their share of the bandwidth.

To solve these problems, several fair queueing algorithms have been proposed, such as fair queueing [38, 13], Weighted Fair Queueing (WFQ) [13], self-clocked fair queueing [18], worst-case fair weighted fair queueing [4], frame-based fair queueing [43], and start-time fair queueing [20]. Fair queueing and WFQ are presented in the following sections.

2.1.1 Fair Queueing

Nagle proposed a fair queueing algorithm in [38]. At each output port, the switch maintains separate queues for different traffic sources, one queue for each traffic source. When an output port becomes idle, the switch scans all the queues at the output port in a round-robin manner, taking the first packet on the next queue for transmission. If a traffic source sends packets too quickly, it only increases its own queue length. Thus, this algorithm allocates buffers fairly among all the traffic

sources, prevents longer delay for well-behaved sources, and prevents a traffic source from arbitrarily increasing its share of the link bandwidth. Some ATM switches use this algorithm.

However, a problem with this algorithm when used in a packet network is that it does not take packet lengths into consideration. A traffic source using large packets gets more bandwidth than a traffic source using small packets. Demers et al. [13] suggested that byte-by-byte round robin should be used instead of packet-by-packet round robin. The switch scans all the queues at an output port repeatedly in a round-robin manner, one bit at a time, until it finds the time when each packet will be completely served. Then, the packets are sorted in the order of their finish service time and will be transmitted in that order [44]. Thus, bandwidth is fairly shared among all the traffic sources.

Demers et al. [13] compared the fair queueing algorithm with the FIFO queueing algorithm using simulations of Telnet and File Transfer Protocol (FTP) sources over a datagram network. Their results show that fair queueing provides several important advantages over FIFO queueing: fair allocation of bandwidth, lower delay for sources using less than their full share of bandwidth, and protection from ill-behaved sources [13]. Davin and Heybey [12] simulated TCP sources transmitting over a packet network. Their results show that the fair queueing algorithm enforces both uniform and non-uniform resource allocation better than the FIFO queueing algorithm does.

2.1.2 Weighted Fair Queueing

One problem with the above fair queueing algorithm is that all the traffic sources are given the same priority. In many situations, some sources should be given more bandwidth than others. Weighted Fair Queueing (WFQ) [13] is a modified version of fair queueing which can provide different bandwidths to different traffic sources. It is identical to the packet-by-packet generalized processor sharing (PGPS) defined in [40].

In WFQ, each traffic source is associated with a weight. The weight determines the relative number of service time units a traffic source gets in each service cycle. For example, if source one has weight one and source two has weight two, then source two has twice the service time units as source one in each service cycle and thus in the switch packets from source two are sent twice as often as packets from source one. Before a packet is put into a queue, the WFQ algorithm estimates the service completion time of the packet based on the arrival time and the size of the packet and the weight of the traffic source. The service completion time is then used as the timestamp of the packet. The switch transmits packets in the increasing order of their timestamps. Supposing two packets have the same arrival time and packet size but different weights, the packet with heavier weight would have a closer service completion time and would be transmitted first. More details about the algorithm are given in the next chapter.

WFQ has several advantages. It inhibits longer delay for well-behaved sources. It prevents ill-behaved sources from taking up all the bandwidth. Above all, it can provide different priorities to different traffic sources. Some traffic sources may obtain

more bandwidth if they have heavier weights.

Demers et al. introduced the concept of WFQ in [13]. Parekh and Gallager [40] presented an implementation of WFQ. Simulation results in [22] show that WFQ with a buffer management scheme can distribute excess bandwidth to flows according to the ratio of their reserved rates.

2.2 Buffer Management for QoS Provisioning

A queueing (or scheduling) scheme controls the transmission opportunities that a traffic flow gets. However, it cannot guarantee transmission opportunities to a flow if the buffer space is occupied by another misbehaving flow. If rate guarantees are to be provided, then buffer management is required independently of any queueing scheme [22].

A buffer management scheme decides if an arriving packet should be allowed entry into the buffer. Buffer management may be used to provide isolation between different service classes and different VCs and to guarantee fairness to competing traffic flows. In [22], a buffer management scheme was proposed to provide rate guarantees to individual flows multiplexed into a FIFO queue.

The paper investigated how to provide rate guarantees by relying solely on buffer management. A buffer management approach which links the amount of buffer space assigned for a flow to the rate it is guaranteed to receive was presented. In particular, a buffer occupancy threshold based on peak rates and a buffer occupancy threshold based on token rates and burst sizes were proposed to provide rate guarantees to individual flows. The paper investigated the tradeoffs involved when providing rate

guarantees by simply using buffer management. The performance of the scheme was compared to that of a scheduler based scheme (i.e., WFQ). Simulation results show that the great simplicity of the buffer management-based scheme comes at the cost of potentially much larger buffer requirements [22]. The proposed scheme requires more buffer space than WFQ to achieve the same rate guarantees.

The paper also investigated the benefit of a hybrid scheme which combined a limited WFQ scheduling with the buffer management scheme. Simulation results show the hybrid scheme is capable of a broad range of tradeoffs between efficiency and complexity [22]. The hybrid scheme can provide performance close to WFQ with buffer sharing [22]. At the same time, the complexity of the hybrid scheme is less than that of WFQ with buffer sharing.

2.3 Packet-level Discard

Transmission Control Protocol (TCP) is a general purpose protocol which provides robust performance in many network environments. However, the throughput achieved by TCP over ATM networks can exhibit very poor performance. Romanow and Floyd [41] presented three possible causes: the delivery of cells from incomplete packets, link idle time, and the retransmission of packets that have already been correctly received. The main reason for the low throughput of TCP over ATM is fragmentation. TCP packets are broken into ATM cells for transmission over the ATM network; at the destination, these cells are reassembled into TCP packets. When an ATM switch drops a cell from a packet, the rest of the cells from the packet are useless since the packet is incomplete and will be retransmitted. But these use-

less cells are still transmitted and all the cells from this packet will be retransmitted, which wastes bandwidth. This is the primary reason for the low throughput of TCP over ATM [23, 41].

The above scheme which discards any incoming cells after buffer overflow is called tail drop. Since tail drop results in low throughput, packet-level discard was proposed to discard cells intelligently to maximize throughput. The goal is to eliminate packets with incomplete cell counts so as to maximize the number of complete packets being transmitted. Several packet-level discard strategies, such as PPD [41], EPD [41], and ESPD [9], were proposed to alleviate the effect of the fragmentation. These strategies are presented next.

2.3.1 Partial Packet Discard

PPD was first proposed by Armitage and Adams [2] as selective cell discarding. In PPD, if a switch runs out of buffer space and an incoming cell is dropped, it discards all the remaining cells from the same packet except the last cell. The last cell of the packet should be transmitted since it is used to delimit the packet boundary.

PPD discards the tail part of a packet or an entire packet. This results in improved throughput since the bandwidth used to transmit damaged packets is saved to transmit complete packets. However, the improvement is limited because the switch starts dropping cells only when the buffer overflows. Thus, the congested link still transmits a number of cells (namely the first few) belonging to packets that are incomplete.

Simulation results of PPD were presented in [41]. The authors focused on throughput and showed that PPD produced a higher throughput than the tail drop

scheme. Similar results were presented in [9].

2.3.2 Early Packet Discard

EPD aims at discarding an entire packet prior to buffer overflow. In EPD, a certain buffer threshold is set. If the buffer queue length exceeds the threshold, the switch is ready to discard incoming cells. Whenever the switch sees the first cell of an incoming packet and the queue length is still above the threshold, it drops the cell. All the subsequent cells from the same packet except the last cell are dropped as well.

EPD prevents new packets from entering the buffer when the buffer is in danger of overflowing. Thus, incomplete packets are prevented from being transmitted. At the same time, EPD reserves some buffer space for the packets that have already partly entered the buffer, so it increases the chance for these packets to be transmitted successfully. With EPD, the throughput for TCP over ATM is significantly improved.

Romanow and Floyd [41] showed that EPD could bring the throughput performance to its optimal level. Cheon and Panwar [9] also presented some simulation results and validated the significant throughput improvement of EPD.

2.3.3 Early Selective Packet Discard

ESPD has an added packet selection mechanism which tries to drop packets from highly active sessions. The original ESPD scheme [9] has a drop timer and three thresholds which are the high threshold, the low threshold and the drop-list threshold. Drop-list is a set of VCs for which the first cell of a packet has already been dropped. The high threshold is used to capture VCs into the drop-list, the low

threshold is to release VCs from the drop-list, and the drop-list threshold is to limit the number of VCs in the drop-list. The drop timer is designed to maintain fairness across sessions.

In ESPD, if the buffer occupancy exceeds the high threshold, the first cell of an incoming packet is dropped and the corresponding VC is added to the drop-list. All the subsequent cells from the same packet except the last cell are dropped. Once the number of VCs in the drop-list equals the drop-list threshold, no VC is added to the drop-list any more. If the last cell using a VC in the drop-list comes and the buffer occupancy is less than the low threshold, the VC is released from the drop-list. When the first VC is captured into the drop-list, the drop timer is activated; when a VC is released from the drop-list, the drop timer is deactivated. If the drop timer expires prior to the deactivation, all the VCs in the drop-list are released simultaneously [9].

A revised ESPD scheme [10] was proposed to reduce the complexity of the original scheme. In this simplified ESPD, the drop-list threshold is not used since the high threshold and the low threshold perform well to select the sessions for packet discarding. The drop timer is turned off because it does not improve fairness notably.

Simulation results in [10] show that ESPD improves the throughput over EPD by 15% to 20% for the network configuration used. Their simulation results also demonstrate that ESPD improves fairness.

2.4 Research Problems

Since ATM networks are used as backbone networks and many TCP-based applications are running on top of these networks, the performance of TCP over ATM

becomes an interesting issue. In this research, we use simulations to study mechanisms that can be used to optimize the performance of TCP over ATM. In particular, we examine the following problems:

- Per-VC queueing: Simulation results [12, 13] show that fair queueing provides more fairness to traffic sources than FIFO queueing over packet networks. However, no research was done to examine the performance of fair queueing on TCP over ATM. Does fair queueing perform well on TCP over ATM? Does it provide more fairness to the competing TCP sources over an ATM network than FIFO queueing does? This thesis studies the performance of fair queueing on TCP over ATM.

- Buffer management: A buffer management scheme was proposed and studied in [22]. The paper studied the scheme in a packet network and validated that buffer management is required independently of any scheduling scheme in order to provide rate guarantees. In this thesis, we investigate a simpler buffer management scheme, a dynamic buffer allocation scheme with two parameters, to be used with per-VC queueing. We study the performance of this dynamic buffer allocation scheme.

- EPD: EPD was studied by Romanow and Floyd in [41] and Cheon and Panwar in [9, 10]. Romanow and Floyd investigated the throughput performance of TCP over ATM. Since fairness is also a main concern in a congested network, we study the fairness performance in addition to the throughput performance. We also investigate the effect of different EPD thresholds.

- EPD and per-VC queueing: Although much research work was done to study the performance of EPD and the performance of per-VC queueing, no published research has studied the performance of the combination of EPD and per-VC queueing. In this thesis, we examine the performance of the combination of EPD and per-VC queueing.

2.5 Summary

A simple output queueing switch only has a FIFO queue for each output port. The arrival order of packets determines the transmission order of the packets, the departure time of the packets and the buffer space allocations. Several problems with the FIFO buffer scheme exist: unfair allocation of bandwidth, longer delay for sources using less than their full share of bandwidth, and lack of protection from ill-behaved sources. This chapter presents per-VC queueing, fair queueing, and buffer management which were proposed to solve these problems.

TCP over ATM can show very poor performance mainly because of fragmentation. A TCP packet is segmented into smaller 53-byte ATM cells for transmission over an ATM network, and the cells are reassembled into a TCP packet at the destination. One cell missing causes the whole packet to be dropped at the destination and to be retransmitted at the source. Transmitting incomplete packets on the network results in poor performance. PPD, EPD and ESPD, which were proposed to improve the performance, are presented in this chapter.

The issues addressed in this thesis are also defined in this chapter. The next chapter presents the design and implementation issues of our simulation model.

Chapter 3

Model Design and Implementation

Per-VC queueing and packet-level discard have been designed and implemented as part of the ATM traffic and network (ATM-TN) simulator developed within the TeleSim project. The ATM-TN simulator is dedicated to the modeling and simulation of ATM networks at the cell level [47]. The simulator is built on SimKit, an efficient package for general purpose discrete event simulation (DES).

This chapter describes the design and implementation issues of our model. It is organized as follows. We first present features of SimKit. We then provide an overview of ATM-TN. After that, we discuss the design and implementation issues of per-VC queueing, buffer management, WFQ, PPD and EPD.

3.1 SimKit

SimKit, developed as part of the TeleSim project, is designed for fast DES. It represents a simple logical process view of simulation enabling both sequential and parallel execution without code changes to application models [19]. There are two versions of SimKit: a C++ version and a Java version. The C++ version is a C++ class library, and the Java version is a Java class library. In the following sections, we present the application programming interface (API) that SimKit provides, which is followed by a short description of the six simulation execution phases.

3.1.1 SimKit Application Programming Interface

The SimKit API consists of three classes, one type, and several convenience functions and macros. The three classes are: sk_simulation for controlling simulation, sk_lp for modeling application behavior and state transitions, and sk_event for modeling the interaction between the logical processes. The one type is sk_time, which is used to represent simulation time. Several functions and macros are provided to support easy access to simulation information, such as current LP, current event, and current time. A library for random number generation and distributions is also provided, including uniform, normal, exponential, geometric, binomial, Erlang, and Poisson distributions.

The sk_simulation class provides the initialization interface to the SimKit simulation kernel. Classes derived from sk_lp class represent components of the application physical system. Classes derived from sk_event class represent events that take place in the application. An LP communicates with another LP by creating an event and sending it to the destination LP to be processed at a specific time. A simulation model is constructed by deriving LPs from sk_lp class and events from sk_event class. An instance of sk_simulation class is needed to invoke the run time simulation kernel [19].

3.1.2 Simulation Execution Phases

Simulation execution is divided into six phases. Phase one is program initialization in which the function main is called and the simulation object is instantiated. Phase two is SimKit and model global initialization. In this phase, the simulation object is initialized, all model LPs are instantiated and allocated to processors, and the control

is passed from the application's main function to the simulation run time system. Phase three is LP initialization in which all LPs' initialize member functions are invoked. Once all LPs have been initialized and seed events have been sent out, phase four—simulation execution—begins. The SimKit kernel begins dispatching events to their destination LPs. When an event arrives at an LP, the LP member function process is called. This phase ends when there are no more events to process, when the simulation end time is reached, or when an error occurs. Phase five is LP termination, in which all LPs' terminate functions are invoked. At last, the simulation run time system returns control back to the application's main function in the simulation clean-up phase [19].

3.2 ATM-TN

"ATM-TN is an integrated set of simulation tools with a modular architecture that supports the modeling, simulation and analysis of ATM networks." [50] It has been designed for ATM network design and analysis. Its features include modular architecture, a modeler's framework including traffic and network models, an object oriented software system written in C++ and built on SimKit API, and well organized simulation input and output files [50].

3.2.1 ATM-TN Architecture

The general structure of the ATM-TN simulator is illustrated in Figure 3.1 [47]. The major components of ATM-TN are presented in [47]:

1. ATM Network Model: It provides the ATM switch and network models.

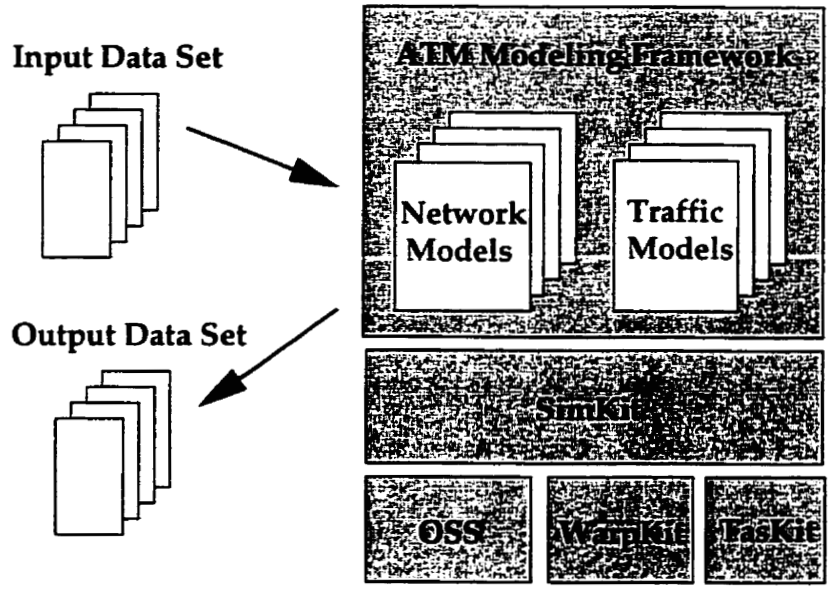

Figure 3.1: ATM-TN Simulator Architecture

2. ATM Traffic Model: It provides the ATM traffic models.

3. ATM Modeling Framework: It provides the modeling framework that builds the user defined simulation scenario and controls the simulation execution.

4. Optimized Sequential Simulator (OSS): It provides an optimized sequential simulation kernel which supports fast, efficient, and sequential execution.

5. WarpKit: It provides a parallel simulation kernel which supports optimistic parallel execution [29] on shared memory multiprocessor platforms.

6. TasKit: It provides a high performance simulation kernel which supports both sequential and conservative parallel execution [7].

7. SimKit C++: It provides a C++ based library on top of the sequential and parallel kernels.

The ATM network model, traffic model, and modeling framework are described further in the subsequent sections.

3.2.2 ATM Network Model

The ATM network model provides the ATM switch and network models. As seen in Figure 3.2 [17], an ATM-TN network model is made up of traffic sources/sinks (TSSs), links, end nodes, and switches. TSSs generate and absorb cells carried by the network. End nodes are a simplified type of switch used at the edges of an ATM network. Switches and end nodes are connected by bidirectional links.

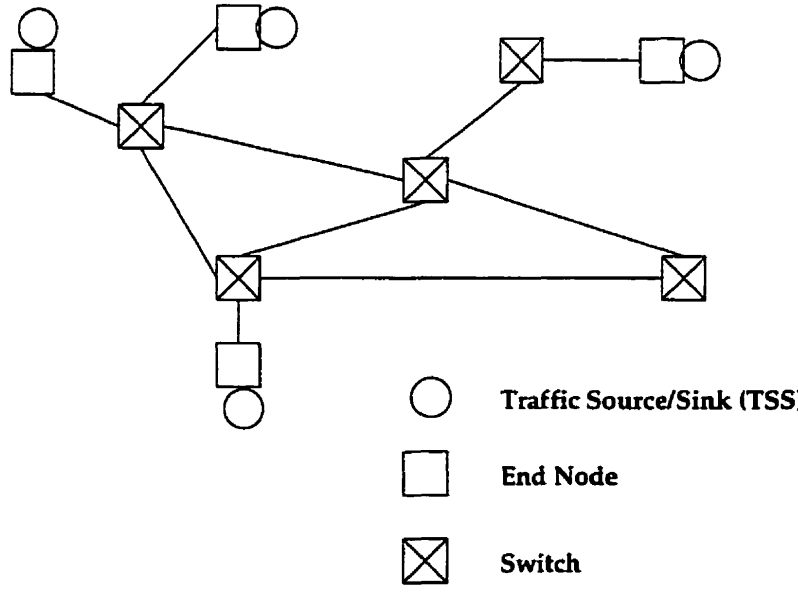

Figure 3.2: An ATM-TN Network Model

The switch models include three single-stage switch models, i.e., per-port, shared,

and crossbar switches, and three multistage switch models, i.e., delta multistage interconnection network versions of per-port, shared, and crossbar switches. Point-to-point switched virtual channels (SVCs) and permanent virtual channels (PVCs) are supported. Virtual paths (VPs) are also supported. The switch models provide several traffic shaping schemes at end nodes, several call admission control schemes at switches, a user parameter control scheme at access switches, a network parameter control scheme at internal switches, separate buffer queues for different service classes, and separate thresholds for dropping cells with low loss priority and for explicit forward congestion notification [17].

3.2.3 ATM Traffic Model

Several kinds of ATM traffic models are provided, including deterministic, Poisson, MPEG, Ethernet, Web, TCP and persistent. Since Poisson and TCP are the only two traffic models used by the experiments in this thesis, we only present these two models here.

Poisson Traffic Model

The Poisson traffic model models a variable bit rate (VBR) cell stream in an ATM network. It supports both unidirectional and bidirectional streams. Cell inter-arrival times are exponentially distributed and independent. The Poisson traffic parameters include peak cell rate, sustainable cell rate, service class, the number of cells to send, inter-cell spacing for both forward and backward channels, connection type, and transmission start time. Statistics include the number of cells transmitted, received and lost in each direction [49].

TCP Traffic Model

The TCP traffic model simulates bulk data transfer between two hosts over an ATM network. It supports both unidirectional and bidirectional streams. The full protocol stack in Figure 1.1 is modeled.

The TCP traffic parameters include the number of bytes to send for both forward and backward directions, send and receive buffer sizes, inter-cell spacing, connection type, and transmission start time. Optional parameters include packet send queue size, maximum transmission unit (MTU), maximum segment size (MSS), fast timer, and slow timer. Higher performance extensions and Nagle's algorithm can be enabled or disabled. Statistics records are generated for each of the socket, TCP/IP, AAL, and ATM layers of the model, which include the number of bytes sent and received, the number of segments retransmitted and duplicated, and the number of cells received and lost for each source/sink. Detailed tracing at the TCP/IP, AAL and ATM level is supported.

3.2.4 ATM Modeling Framework

The ATM modeling framework is a framework for all the model components in the ATM-TN simulator. It provides support and startup functions for the various model components, such as switches, TSSs, links, ports, VPs, and PVCs. These include simulation initialization and control, model construction and termination, model component identification, external event notification, utility functions, and output support. The modular design of the ATM modeling framework allows new simulation model components to be easily added into ATM-TN [50].

3.3 Per-VC Queueing

Per-VC queueing means there are different queues for different VCs. It is applied at an output port of a switch. It can guarantee a low end-to-end delay to a light user, prevent a greedy user from taking up all the bandwidth, and manage priority of each VC. In this section, we first describe the original scheme without per-VC queueing used in ATM-TN. Then, we present some design and implementation issues of per-VC queueing.

3.3.1 Original Scheme without Per-VC Queueing

As seen in Figure 3.3, at every output port of a switch in ATM-TN, there are separate buffer queues for different service classes, CBR, rt-VBR, nrt-VBR, ABR, and UBR. A scheduler decides which cell gets to leave the buffer next. Two cell scheduling schemes, round robin and exhaustive priority, can be used to schedule the removal of cells from these buffer queues.

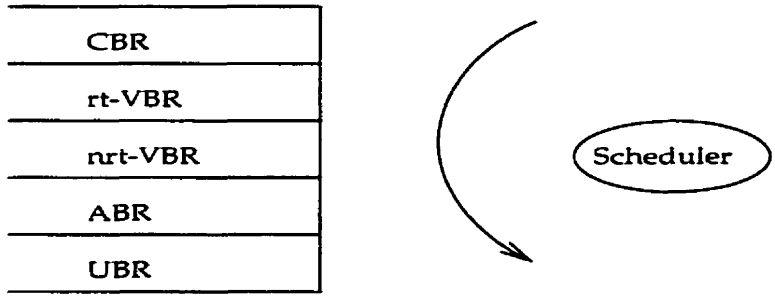

Figure 3.3: Original Scheme without Per-VC Queueing

Round Robin

The round robin scheduling scheme cycles through the queues in turn and services the first non-empty queue, i.e., the first queue with a waiting cell. If all the queues are non-empty, then the scheduler will service every queue one by one. In a network with variable-sized packets, the round robin scheduling scheme cannot allocate bandwidth fairly because traffic sources using large packets get more bandwidth than traffic sources using small packets. However, this scheduling scheme can provide fair bandwidth allocation in ATM networks since ATM networks use fixed-sized cells as their transmission units.

Exhaustive Priority

The exhaustive priority scheduling scheme is usually used in the situations where queues to be serviced are ordered in terms of priority. This scheme is often employed to service multiple ATM service classes where CBR has the highest priority and UBR has the lowest priority. In each service cycle, the scheduler searches queues from the highest priority to the lowest priority and services the first cell it encounters. If the highest priority queue has cells waiting to be transmitted in every cycle, it will be serviced exhaustively and the other queues will not receive any service. The exhaustive priority scheduling scheme can provide an upper bound on the delay for a particular service class based on the sum of the peak cell rates allocated to higher priority service classes, the output link rate, and the queue length of that service class [34]. This characteristic makes the scheme very attractive to the real-time traffic, i.e., CBR and rt-VBR, which requires tightly constrained delay.

The original scheme without per-VC queueing does prevent traffic sources be-

longing to different service classes from affecting each other. However, it does not prevent sources belonging to the same service class from interfering with each other. An aggressive source may still take up all the bandwidth, preventing a conforming source in the same service class from getting its fair share of the bandwidth. Therefore, per-VC queueing is designed and implemented.

3.3.2 Scheme with Per-VC Queueing

The new scheme with per-VC queueing is illustrated in Figure 3.4. There are different queues for different VCs within each service class, instead of one FIFO queue for each service class. There are two levels of scheduling. Scheduler one selects one service class out of those five service classes. Scheduler two decides which VC will get service in that service class. Either round robin or exhaustive priority may be used in the level one scheduling, and either round robin or WFQ may be used in the level two scheduling.

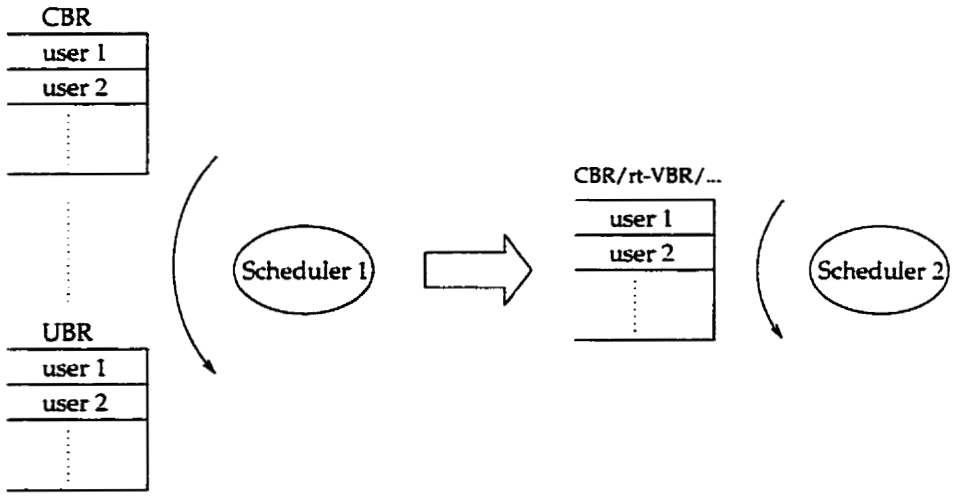

Figure 3.4: Scheme with Per-VC Queueing

In the per-VC queueing implementation, a VC table is built at each output port. Each entry of this VC table has several fields, including virtual path identifier (VPI), virtual channel identifier (VCI) and weight. VPI and VCI are used to identify a VC; they are generated automatically during the connection setup phase. Weight is the weight assigned to that VC; it is obtained from an ATM-TN input file and is used by WFQ.

A buffer management scheme decides if an arriving cell should be allowed entry into the buffer. We investigate a dynamic buffer allocation scheme to be used with the per-VC queueing scheme. It allows more efficient use of the available buffer. Another attractive attribute of this dynamic buffer allocation scheme is that it can limit the maximum buffer space that a VC can use, and can guarantee a minimum buffer allocation to a VC. The scheme is presented in the next section.

3.4 Dynamic Buffer Allocation

Switch buffer size is critical for the switch performance. If the buffer size is too small, the switch throughput and its capacity to handle many connections may be constrained; if the buffer size is large, the cost of the switch is high and the network delay may be intolerable. In the per-VC queueing implementation, there is shared buffer space at each output port. This shared buffer space is used by all the VCs at the output port and is flexibly allocated to VCs as needed. When a traffic source asks for some buffer space to store an incoming cell, the space is obtained from this shared buffer space. After the traffic source finishes using the space, the space is returned to this shared buffer space. It has been shown that a smaller packet

loss ratio is achieved by using dynamic rather than static buffer management [46]. With static buffer management, much of the buffer space allocated to some traffic sources may never be used, and at the same time, packets from other sources may be dropped due to a lack of buffer space for those traffic sources. With dynamic buffer management, the unused buffer space can be used by any traffic source which needs it. The dynamic buffer allocation scheme presented in this thesis uses two parameters, maximum buffer size and minimum buffer size. A similar scheme using maximum and minimum buffer queue lengths was also considered in [30].

Maximum Buffer Size

A user can set a maximum buffer size. The maximum buffer size specifies how much buffer space can be used for a traffic source. Incoming cells from the traffic source for which buffer usage has reached the maximum buffer size are dropped. Thus, a greedy user can be prevented from hogging all the buffers. The maximum buffer size is expressed as a percentage of the total buffer space.

Minimum Buffer Size

A user can set a minimum buffer size. The minimum buffer size specifies how much buffer space should be guaranteed for a traffic source. A cell arriving at a full buffer will get some buffer space if the arriving cell belongs to a VC whose buffer usage is below the minimum buffer size. This buffer space is obtained by releasing the buffer space of the last cell on a VC queue whose length is the longest among all the VC queues and whose buffer usage is over the minimum buffer size. Therefore, a minimum throughput can be guaranteed to a source. The minimum buffer size is expressed as a percentage of the total buffer space.

3.5 Weighted Fair Queueing

In WFQ [13], before a packet is put on a VC queue, the service finish time of the packet is calculated based on the arrival time and the size of the packet and the weight of the traffic source and is used as the timestamp of the packet. The weight represents how much priority a traffic source has and determines the relative number of service time units that the traffic source gets in each service cycle. The switch transmits packets in increasing order of their timestamps. The packet with the smallest timestamp is always sent next. WFQ has been implemented based on the notion of round number [13] or virtual time [40]. In this section, we first describe the concept of virtual time. Then we present the WFQ implementation.

At the beginning of the simulation, the virtual time $V(t)$ is zero since there is no cell in the VC queues and the switch is idle. Here, t is the simulation time maintained by the simulator. Virtual time is updated whenever an inactive session becomes active or whenever an active session becomes inactive. An active (backlogged) session means there is at least one cell waiting in the corresponding VC queue to be transmitted. $V(t)$ is calculated using the following formula:

$$V(t_j) = V(t_{j-1}) + \frac{t_j - t_{j-1}}{\sum_{i \in B_j} W_i}, j = 2, 3, \ldots \tag{3.1}$$

where B_j is the set of active sessions in the interval (t_{j-1}, t_j). It is updated when an active session becomes inactive or when an inactive session becomes active. W_i is the weight of session i; it is obtained from the VC table at the output port.

The rate of increase of virtual time V is $\frac{1}{\sum_{i \in B_j} W_i}$. Each backlogged session k

receives service at rate $W_k \frac{1}{\sum_{i \in B_j} W_i}$. So a session with a heavier weight is serviced faster. Virtual time increases continually. When there is no active session in the system, virtual time stops increasing at that point because no session receives service, and virtual time will increase later when a session becomes active.

When a cell arrives at an ATM switch at time t, virtual time $V(t)$ is calculated first. Then, the expected service finish time of the cell is obtained using the following formula:

$$F_k = \max\{F_{k-1}, V(t)\} + \frac{L}{W_i} \tag{3.2}$$

F_k denotes the service finish time of the k^{th} cell arriving at the session i buffer queue at time t; F_{k-1} denotes the service finish time of the $k-1^{th}$ cell; F_0 is defined to be zero; $V(t)$ is the virtual time at time t; L is a constant, which is the length of an ATM cell; W_i is the weight of the session i.

After the service finish time of a cell is obtained, it is used as the timestamp of the cell. When the switch is ready to transmit the next cell, it picks the cell with the smallest timestamp in the buffer queues. In other words, the switch transmits cells in increasing order of their service finish time.

In addition to fair allocation of bandwidth and protection from ill-behaved traffic sources, WFQ has the advantage of providing bandwidth priorities to sources through the use of weights. Some sources may receive more service if they have heavier weights.

3.6 Partial Packet Discard

TCP over ATM exhibits poor performance mainly caused by fragmentation. Once a cell belonging to a packet is dropped, the subsequent cells of the packet will be useless and the destination will ask the source to retransmit the whole packet. However, those cells are still transmitted on the network, which wastes bandwidth. To improve the throughput of TCP over ATM, PPD [41] is implemented.

With PPD, once a switch drops a cell, the switch continues dropping cells from the same packet until it sees the last cell of the packet. If the second bit in the Payload Type (PT) field of an ATM cell is set, it indicates that this cell is the last cell of a TCP packet; otherwise, it indicates that this cell is the beginning of a packet or the continuation of a packet. The last cell of the packet is not dropped since this cell is used to delimit packet boundaries.

PPD is implemented on a per-VC basis. Two more fields are added to the VC table. One field is used to decide whether PPD is used; the other field is used to decide whether cells from the current packet have been dropped. When a cell arrives at a switch, the switch looks through the VC table. If the cell belongs to a TCP packet, PPD is used, and a cell from this packet has already been dropped, the cell will be dropped unless it is the last cell of the packet. When last cell seen, state information (i.e., incoming cells belong to an incomplete packet) is cleared for that VC.

PPD provides better performance than TCP over plain ATM. When a cell must be discarded due to the buffer overflow, PPD discards all subsequent cells of the incomplete packet. Thus, the discarded cells are clustered in fewer packets and

the number of packets to be retransmitted is reduced. Performance is improved. However, the improvement is limited because PPD only discards the end part of a damaged packet, and the first part of the damaged packet is still transmitted.

3.7 Early Packet Discard

A more effective packet-level discard scheme is EPD [41]. EPD was proposed to drop an entire packet prior to buffer overflow. Thus, incomplete packets will be prevented from being transmitted through switches.

With EPD, when a VC queue length exceeds a certain threshold, all the cells but the last cell of any incoming packet will be dropped. Thus, entire packets are discarded before congestion. On the other hand, EPD does not discard subsequent cells in the packets that have already been in transmission; therefore, the packets in transmission can be guaranteed to reach their destinations.

Similar to PPD, EPD is implemented on a per-VC basis. The switch needs to know whether EPD is used on a VC and whether cells from the current packet have been dropped. In addition to this information, the switch needs to monitor whether the current buffer size exceeds a fixed threshold. Detailed pseudocode of an EPD implementation is presented in Figure 3.5 [10].

In the above pseudocode, BOM represents Begin Of Message (packet), and EOM represents End Of Message (packet). The drop-list is a set of VCs for which the BOM cell of a packet has already been dropped. The EOM cells enter the buffer whenever there is space available, because they are used to delimit packet boundaries.

Our EPD implementation follows this pseudocode. When a cell arrives at a

```
We have a cell arrival at an ATM switch:
if the cell's VPI/VCI belongs to drop-list
  if the cell is an EOM cell
    if queue_length < buffer_size
      insert the cell into buffer
    else
      discard the cell
    remove the VPI/VCI from the drop-list
  else
    discard the cell
else
  if queue_length < Threshold
    insert the cell into buffer
  else if (BOM cell or (the buffer is full))
    discard the cell
    capture the VPI/VCI into drop-list
  else
    insert the cell into buffer
```

Figure 3.5: Pseudocode of an EPD Implementation

switch, the switch examines the VC table. If the cell belongs to a TCP packet, EPD is used, and the queue length exceeds the EPD threshold or a cell from this packet has already been dropped, the cell will be dropped unless it is the last cell of the packet. The EPD threshold can be read from an input file which is defined by a user. It is expressed as a percentage of the total buffer size.

PPD prevents the remaining cells of the incomplete packets from being transmitted when the buffer overflows, while EPD prevents the incomplete packets from being transmitted when the buffer is in danger of overflowing to save more bandwidth and buffer space for good packets. Hence, EPD is more efficient than PPD.

3.8 Summary

This chapter discusses the design and implementation issues of our simulation model. Since per-VC queueing and packet-level discard have been designed and implemented as part of the ATM-TN simulator, the ATM-TN architecture, network model, traffic model, and modeling framework are presented. ATM-TN is built on SimKit which is an efficient package for DES, so some issues related to SimKit, such as SimKit API and simulation execution phases, are also presented. This chapter also presents the detailed design and implementation of per-VC queueing, buffer management, WFQ, PPD and EPD. The next chapter provides the validation of our model.

Chapter 4

Model Validation

The design and implementation issues of our per-VC queueing, fair queueing, buffer management and packet-level discard model are presented in Chapter 3. Before we use this model, we may ask: Is this model correct? Validation of our simulation model is very important. It is necessary to assure the correctness of simulation results.

In this chapter, we try to validate our model. We first describe the qualitative validation of per-VC queueing and its related fair queueing schemes. Then we present the quantitative validation of these schemes. The validation of packet-level discard is presented in the next chapter when the characteristics of packet-level discard are studied in detail.

4.1 Qualitative Validation for Per-VC Queueing

This section presents the qualitative validation for per-VC queueing. A scenario using two Poisson traffic sources is built to validate the simulation model of per-VC queueing and its related fair queueing schemes, round robin and WFQ.

4.1.1 Network Topology

The network topology used in this experiment is shown in Figure 4.1. There are two switches in the network. Two Poisson sources share a common output link en route

to their respective destinations. The simulation parameters are as follows:

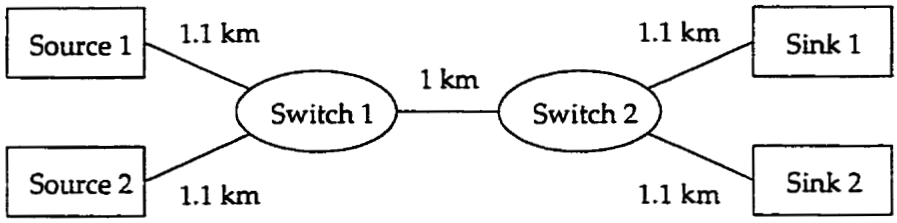

Figure 4.1: Network Scenario for Qualitative Validation for Per-VC Queueing

- Network links are 10.000 cells/second (4.24 Mbps).
- Source 1 and source 2 belong to the same service class, UBR.
- Each switch output port has a 100-cell buffer for the UBR service class.
- Rate of source 1 is fixed at 5000 cells/second.
- Rate of source 2 is varied from 3000 cells/second to 8000 cells/second. with a step of 500 cells/second.
- Duration of the simulation runs is 55 seconds, including 5 seconds warmup period.

Different scheduling schemes are tested, including the scheme without per-VC queueing, round robin scheme with per-VC queueing, and WFQ with per-VC queueing. Different weights for different sources are used to further explore the characteristic of WFQ. Dynamic buffer management is also applied to improve the performance of per-VC queueing.

4.1.2 Performance Metrics

The following performance metrics are used in the experiment:

- Mean End-to-End Delay. Mean end-to-end delay is the average time to transmit a cell from the source to the destination.
- Cell Loss Ratio (CLR). CLR is expressed as a ratio of the number of cells lost to the number of cells transmitted.

4.1.3 Simulation Results

The simulation results are shown in Figure 4.2, 4.3, 4.4 and 4.5. FIFO represents the scheme without per-VC queueing, Round Robin represents the round robin scheme with per-VC queueing, WFQ (w1=w2=1) represents the WFQ scheme with per-VC queueing when two traffic sources have the same weight, and WFQ (w1=1,w2=2) represents the WFQ scheme with per-VC queueing when traffic source two has twice the weight as traffic source one. Overall represents the average.

Dynamic buffer management is not used in Figure 4.2 and 4.3, i.e., the buffer space is allocated to traffic sources in a first-come-first-serve way. Dynamic buffer management is used in Figure 4.4 and 4.5 where 30% of the total buffer space (i.e., 30 cells in this case) is guaranteed to each traffic source.

Figure 4.2 shows that in FIFO both sources have nearly the same delay over the full range of rates. Delay increases sharply near the saturation point (i.e., aggregate input rate = output rate), which is 5000 cells/second. Delay flattens out afterwards, because the buffer at the output port of switch 1 is always full. In Round Robin, before the saturation point source 1 has a longer delay than source 2. This behavior

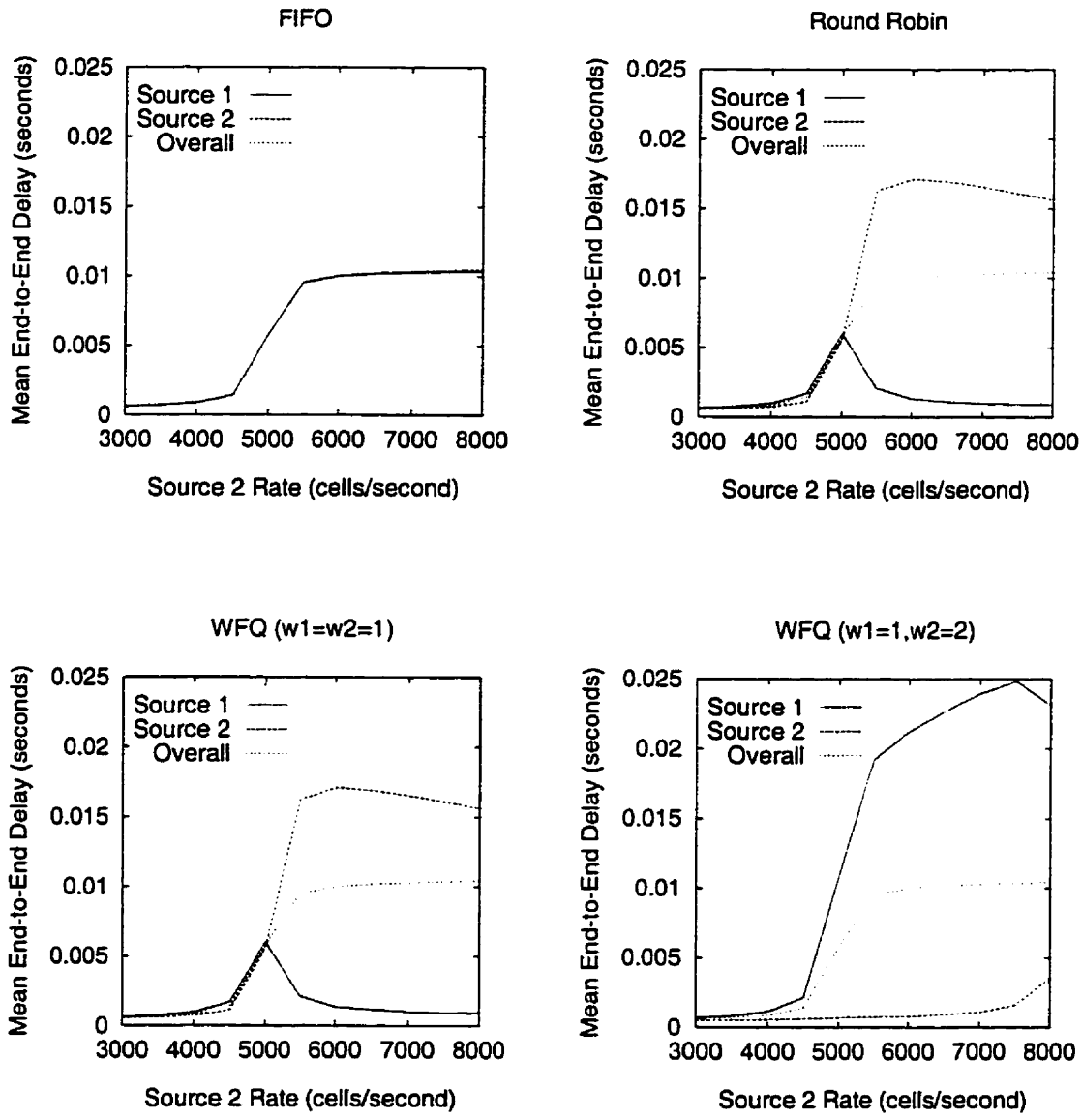

Figure 4.2: Mean End-to-End Delay for FIFO, Round Robin, WFQ (w1=w2=1) and WFQ (w1=1,w2=2). No Dynamic Buffer Management.

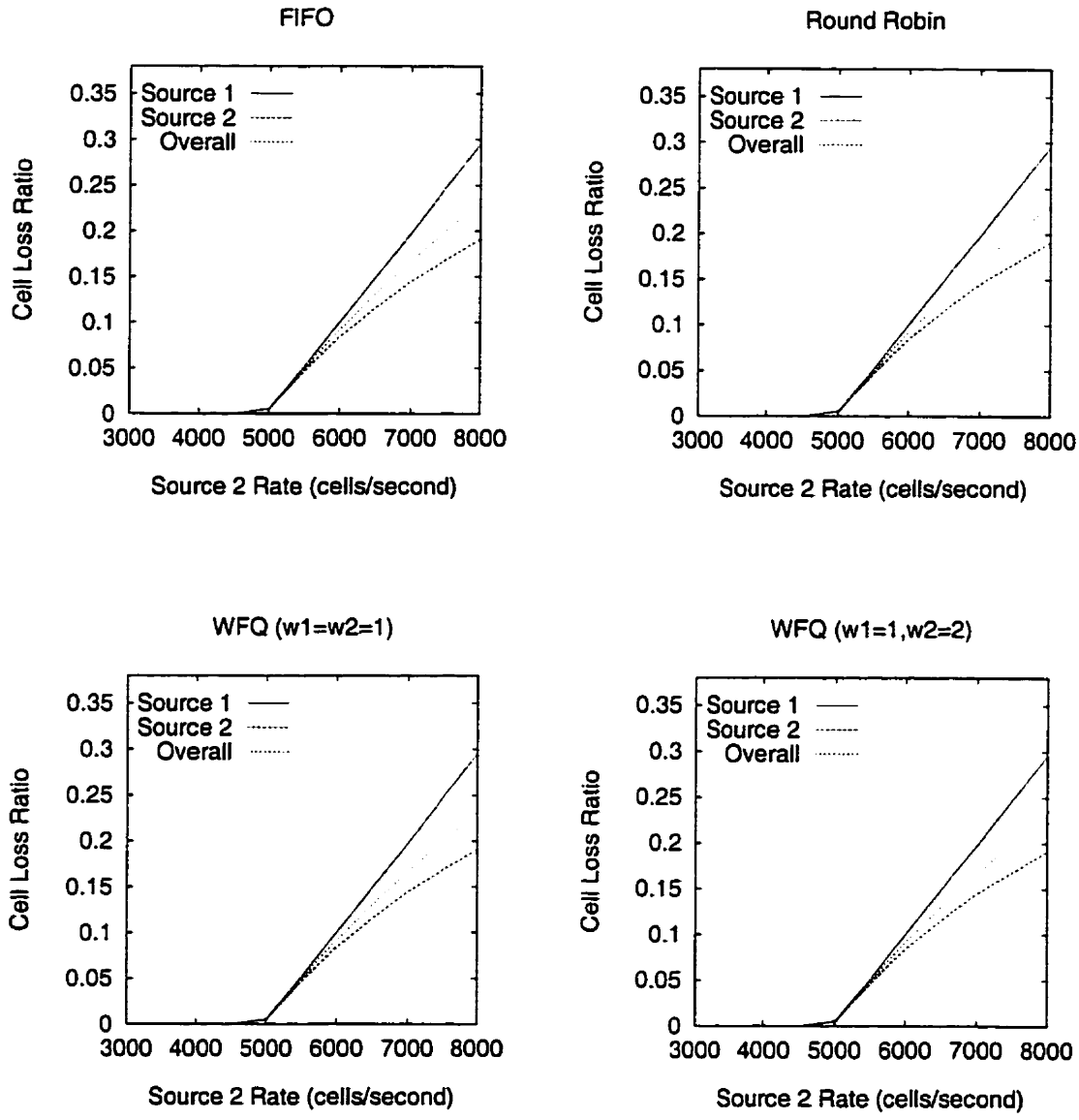

Figure 4.3: Cell Loss Ratio for FIFO, Round Robin, WFQ (w1=w2=1) and WFQ (w1=1,w2=2). No Dynamic Buffer Management.

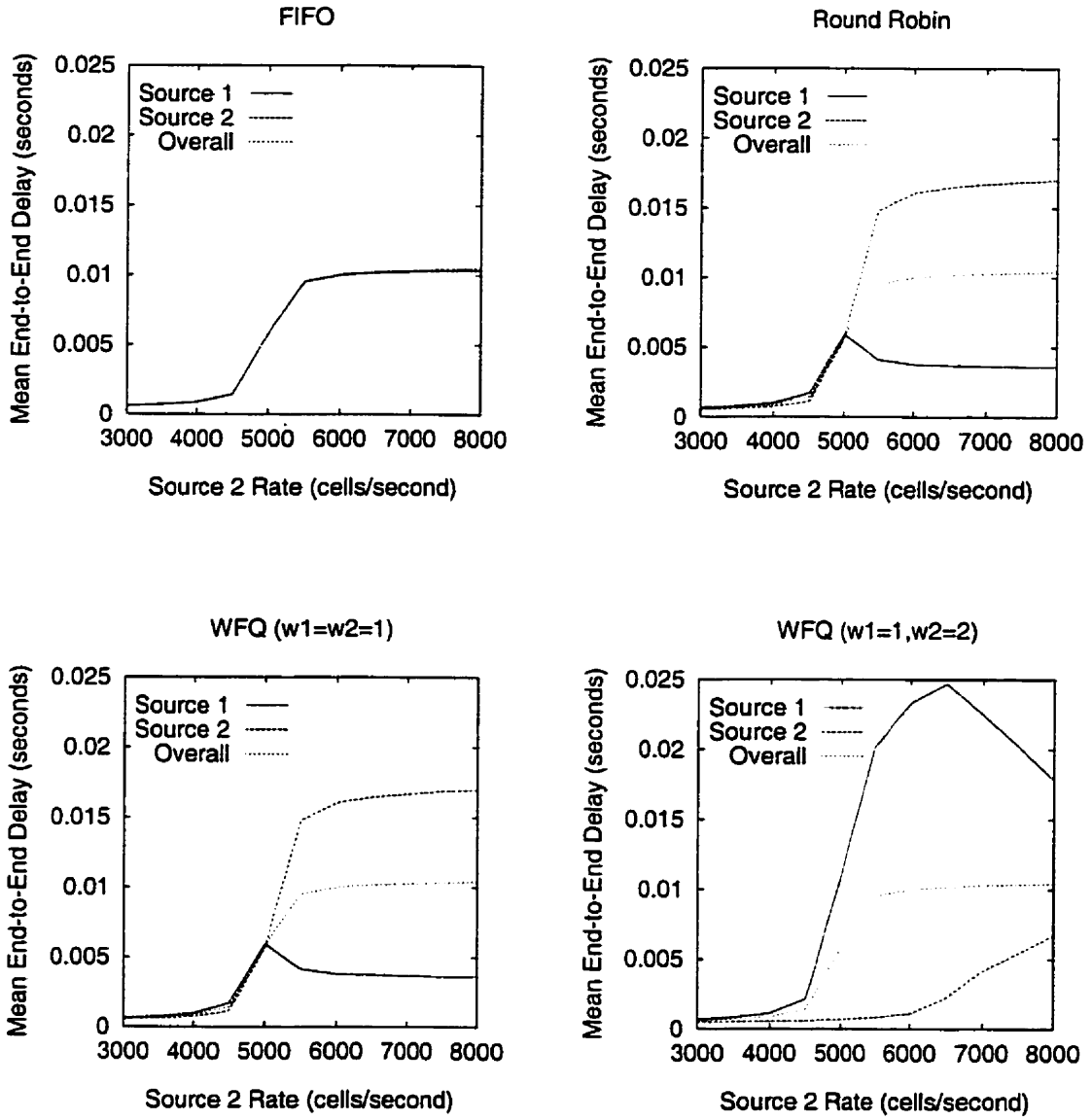

Figure 4.4: Mean End-to-End Delay for FIFO, Round Robin, WFQ (w1=w2=1) and WFQ (w1=1,w2=2). Minimum Buffer Size = 30 cells, Maximum Buffer Size = 100 cells.

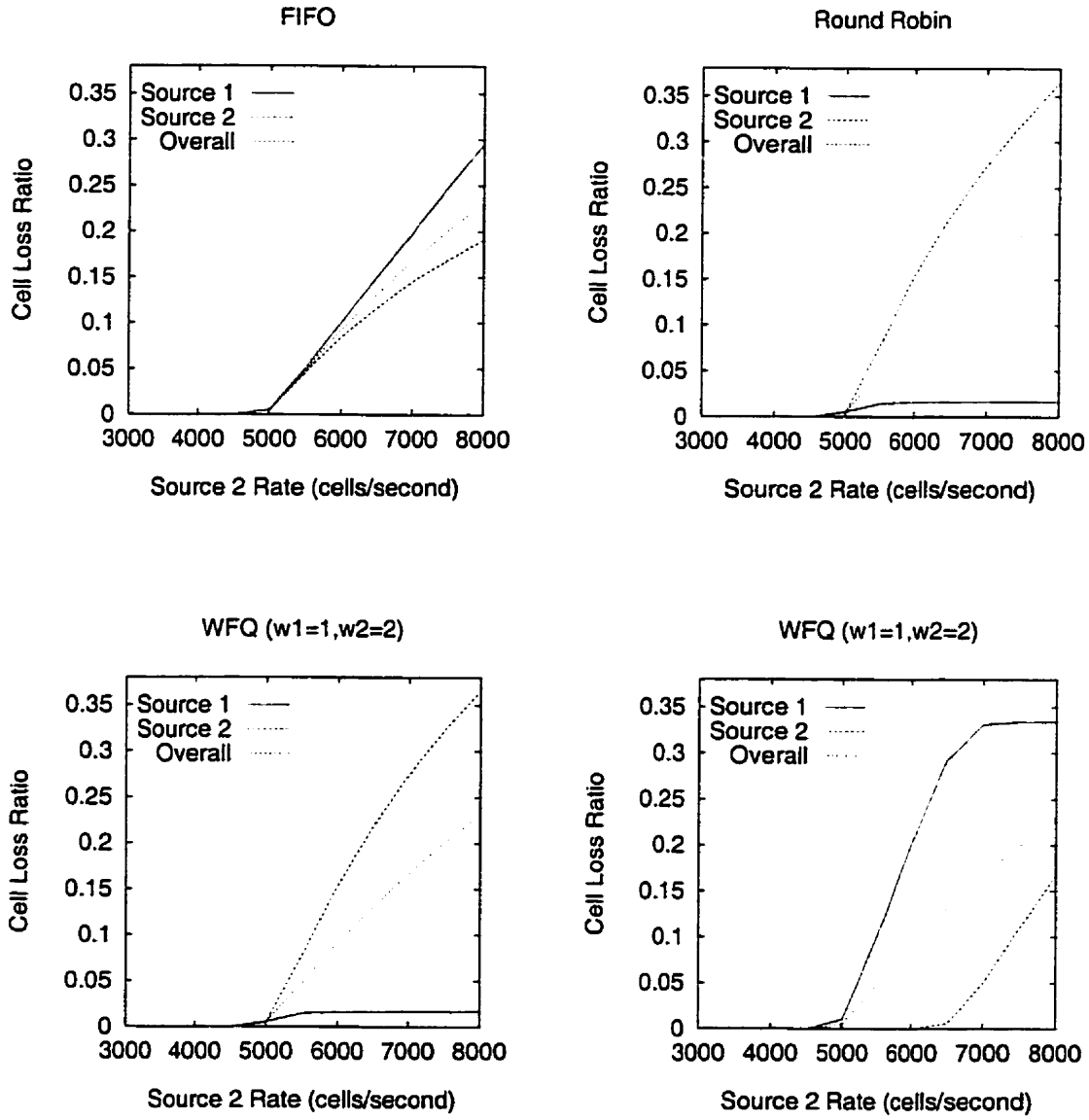

Figure 4.5: Cell Loss Ratio for FIFO, Round Robin, WFQ (w1=w2=1) and WFQ (w1=1,w2=2). Minimum Buffer Size = 30 cells, Maximum Buffer Size = 100 cells.

occurs because source 1 has a higher rate than source 2, so the queue length of source 1 is longer than that of source 2 and the delay for source 1 is longer than that for source 2. After the saturation point, source 2 has a longer delay than source 1 because source 2 has a higher rate than source 1. From the results, per-VC queueing achieves traffic isolation as expected. The delay seen by source 1's cells is largely unaffected by source 2's excessive arrival rate.

In Round Robin and WFQ (w1=w2=1), after the 6000 cells/second point, a slight decrease in delay for both traffic sources is observed. This is because source 2 takes up more buffer space as its rate increases. So source 1's queue length is decreased and thus source 1's delay is decreased. For source 2, since source 1's queue length is decreased. the time it spends waiting for source 1's cells to be transmitted is decreased. So source 2's delay is decreased too.

WFQ (w1=w2=1) has similar results as Round Robin since source 1 and source 2 have the same weight. In WFQ (w1=1,w2=2), source 1 has a longer delay than source 2 since source 2 has twice the weight as source 1 and is serviced twice as fast as source 1. Thus, at most points the queue length of source 2 is shorter than that of source 1; although at some points the queue length of source 2 may be longer than that of source 1, source 2 still has a shorter delay than source 1. After the 7500 cells/second point, as source 2's rate increases, source 2's queue length further increases and source 1's queue length decreases. So source 2's delay increases and source 1's delay decreases. The simulation results illustrate that WFQ can provide different priorities to different traffic sources.

As shown in Figure 4.3, there is no cell loss before the saturation point. Some cells get lost at and after the saturation point. CLR starts to increase at the saturation

point. When the aggregate input rate is larger than the link output rate, the network is overloaded. The overall CLR in the overload case follows the shape (*Input Rate* – *Output Rate*)/*Input Rate*. In other words, the overall CLR follows the shape $(L - 1)/L$ where L is the relative input load (i.e., *Input Rate/Output Rate*). For example, CLR is 0.23 when L is 1.3 at the point 8000 cells/second.

In FIFO, source 1 has a higher CLR than source 2 after the saturation point because source 2 has a higher traffic arrival rate and it is more likely to seize any free buffer space. Round Robin, WFQ (w1=w2=1) and WFQ (w1=1.w2=2) have similar CLR results because dynamic buffer management is not used. Free buffer space is allocated to traffic sources on a first-come-first-serve basis. In this case, source 2 tends to grasp all of the buffer space.

When dynamic buffer management is applied (see Figure 4.5), in Round Robin and WFQ (w1=w2=1) source 2 has a higher CLR than source 1. In this case, although source 2 has a higher traffic arrival rate than source 1, it cannot use all the buffer space. Source 1 is guaranteed at least 30% of the total buffer space. Thus, the performance of per-VC queueing is improved.

In WFQ (w1=1,w2=2) (see Figure 4.5), source 1 has a higher CLR than source 2 because source 2 is serviced twice as fast as source 1 so at most points the average queue length of source 1 is longer than that of source 2.

The mean end-to-end delay of source 2 after the saturation point 5000 cells/second in Round Robin in Figure 4.4 is longer than that in Round Robin in Figure 4.2. In the former case, 30% of the total buffer space is guaranteed to source 1, so source 1's average queue length is longer than that in the latter case.

4.1.4 Different Seeds

A random number generator generates random numbers used by the Poisson traffic. A pair of fixed seeds were used to obtain the above simulation results. To validate the stability of our results, two other pairs of seeds were also used, which have produced results very close to the above results.

Figure 4.6 is the error bar graph of the mean end-to-end delay for FIFO and Round Robin (without dynamic buffer management) using three different pairs of seeds. The dotted line represents the average results from the three different pairs of seeds. An error bar is a line with a limited length, connecting the lowest value to the highest value at a certain point. The smaller the length of the error bar, the closer the results. From this figure, it is obvious that the results from different pairs of seeds are very close to each other. The same behavior is observed for WFQ.

4.2 Quantitative Validation for Per-VC Queueing

Van Melle validated the implementation of per-VC queueing and buffer management in [34]. We use the same methodology here to validate quantitatively the simulation model of per-VC queueing, dynamic buffer management, and fair queueing schemes.

4.2.1 Simulation Configuration

Figure 4.7 shows the network topology used by the simulations in this section. There are two switches in the network. N Poisson sources share a common output link en route to their respective destinations. The simulation parameters are as follows:

- Network links are 10,000 cells/second (4.24 Mbps).

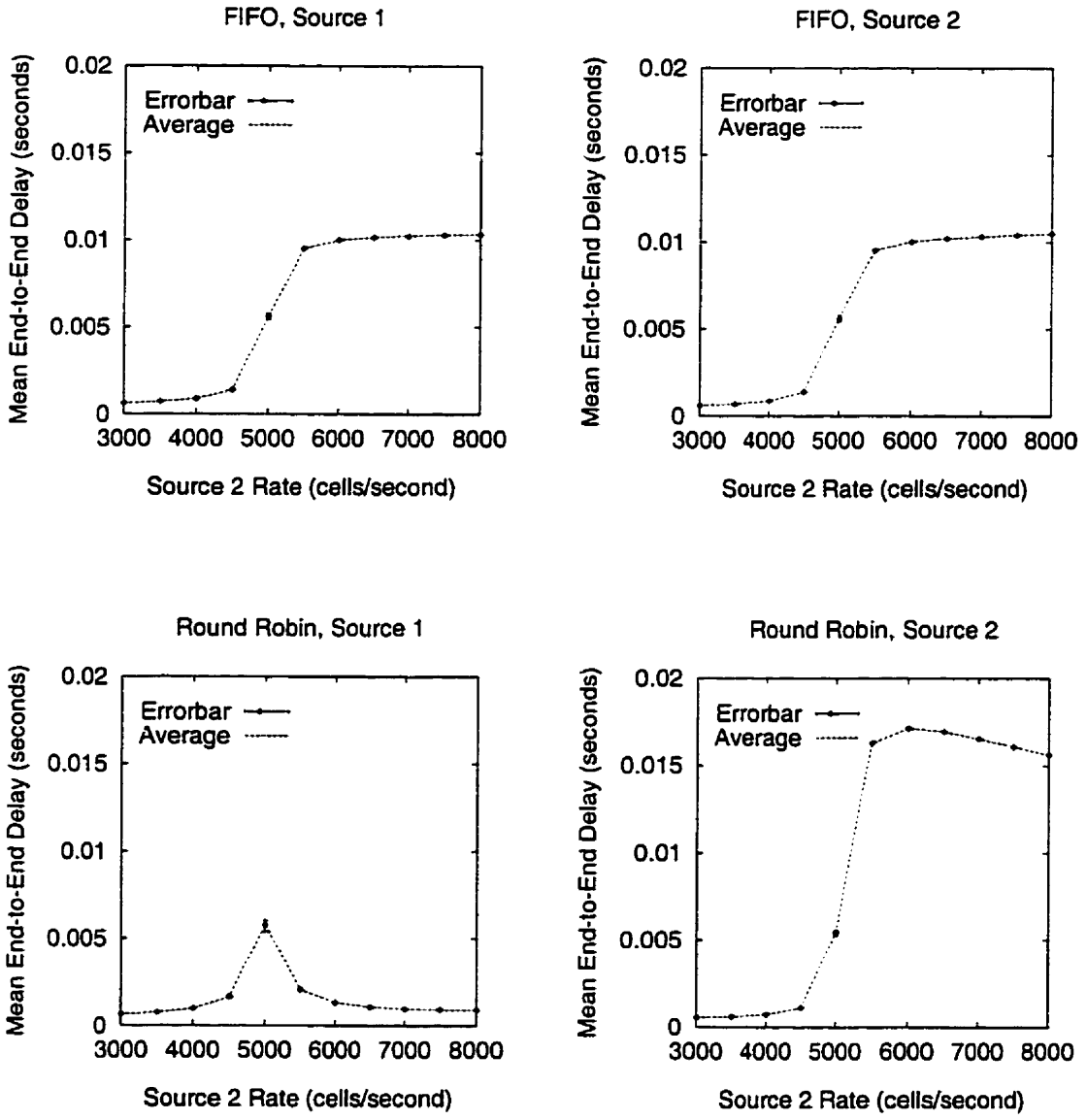

Figure 4.6: Error Bar Graph of Mean End-to-End Delay for FIFO and Round Robin Using Three Different Pairs of Seeds. No Dynamic Buffer Management.

- All the traffic sources belong to the same service class, UBR.
- Each switch output port has a 1000-cell buffer for the UBR service class.
- Duration of the simulation runs is 55 seconds, including 5 seconds warmup period.

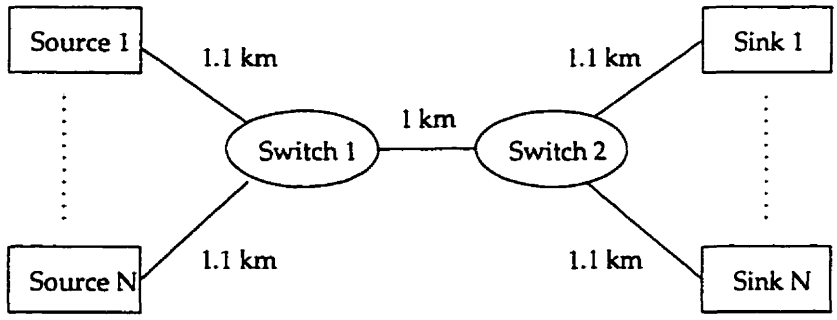

Figure 4.7: Network Scenario for Quantitative Validation for Per-VC Queueing

CLR, a ratio of the number of cells lost to the number of cells sent, is used as a performance metric to validate the simulation model of per-VC queueing, fair queueing, and dynamic buffer management.

4.2.2 Experiment 1: FIFO

This experiment tests the scheme without per-VC queueing. Since there is only a FIFO queue for each service class at each output port of a switch, we call it FIFO. The FIFO queueing has no dynamic buffer management.

Parameters

As described in Table 4.1, four Poisson sources with the same transmission rate are used in this experiment.

Table 4.1: Source Configuration for FIFO

Source	Rate (cells/second)
1	5000
2	5000
3	5000
4	5000

Expected Results

Since the total input rate is 20,000 cells/second and the network bandwidth is only 10,000 cells/second, the network is heavily overloaded. All the sources are expected to suffer from cell loss. The overall cell loss ratio should follow the shape $(L-1)/L$ where L is the relative input load, as described in Section 4.1.3.

Simulation Results

The simulation results are shown in Table 4.2. All the cells from different traffic sources are put on one FIFO queue at an output port of switch 1, waiting to be transmitted to their destinations. The arrival patterns of the sources determine the buffer allocation and the bandwidth allocation. Since all the sources have the same transmission rate, their CLRs are close to each other. The overall CLR is 0.5, which follows the shape $(L-1)/L$ where L is 2.

$Received + Lost \neq Sent$ is observed in nearly all the simulations in this section. There are two reasons for it. One is warmup. At the end of the warmup period, statistics such as cells_sent, cells_received, and cells_missing are reset to 0. However, there are still some cells on the way to their destinations. This is the main cause for $Received + Lost > Sent$. When the simulations end at time 55 seconds, some cells

Table 4.2: Simulation Results for FIFO

Source	Sent (cells)	Received (cells)	Lost (cells)	CLR (%)
1	250256	125626	124595	49.8
2	250325	125268	125121	50.0
3	250235	124256	125993	50.3
4	250331	124850	125493	50.1
Overall	1001147	500000	501202	50.1

may still be in transmission. This is the main cause for $Received + Lost < Sent$. Generally, these two situations occur together.

4.2.3 Experiment 2: Round Robin

This experiment tests the round robin scheme in the per-VC queueing environment. We will verify that round robin and per-VC queueing work correctly.

Parameters

As described in Table 4.3, five Poisson sources with different rates are used in this experiment. We set the maximum buffer ratio to 1.0 and minimum buffer ratio to 0.05. Thus, we guarantee a small amount of buffer space for each VC. This is to ensure that the results are not dominated by the buffer management behavior.

Expected Results

Like the network in experiment 1, the network in this experiment is heavily overloaded because the total input rate is much higher than the network bandwidth. We can calculate the expected CLRs for the traffic sources according to the following rule. First, the 10,000 cells/second bandwidth is divided evenly among five sources.

Table 4.3: Source Configuration for Round Robin

Source	Rate (cells/second)
1	1000
2	2000
3	4000
4	6000
5	8000

Each traffic source should get 2000 cells/second, which is greater than that required by source 1; therefore the CLR for source 1 should be 0. Then, the remaining 9000 cells/second is divided among four sources, which is 2250 cells/second per source. Therefore, source 2 should also have 0 CLR. Finally, 7000 cells/second is divided among the remaining three sources, which is 2333 cells/second per source. The CLRs of the remaining sources can now be easily calculated as $(Rate - 2333)/Rate$ and are given in Table 4.4.

Table 4.4: Expected Results for Round Robin

Source	Expected CLR (%)
1	0.0
2	0.0
3	41.7
4	61.1
5	70.8

Simulation Results

Simulation results are shown in Table 4.5. The round robin scheduler in the per-VC queueing environment behaves exactly as expected.

Table 4.5: Simulation Results for Round Robin

Source	Sent (cells)	Received (cells)	Lost (cells)	CLR (%)
1	49749	49748	0	0.0
2	100248	100251	0	0.0
3	200239	116667	83554	41.7
4	300614	116667	183964	61.2
5	399556	116667	282904	70.8
Overall	1050406	500000	550422	52.4

4.2.4 Experiment 3: Round Robin with Buffer Management

This experiment examines the round robin scheduling scheme used in conjunction with different buffer management settings. It verifies that the dynamic buffer allocation works properly and can improve the performance of per-VC queueing.

Parameters

The source configuration is the same as that in experiment 2 (see Table 4.3). This experiment is composed of four separate trials which have different buffer management settings. The buffer management settings of these four trials are shown in Table 4.6.

Table 4.6: Buffer Management Settings

Trial	Maximum Buffer Ratio	Minimum Buffer Ratio
1	1.0	0.0
2	1.0	0.05
3	0.5	0.0
4	0.2	0.0

Expected Results

When dynamic buffer allocation is not applied (i.e., maximum buffer ratio = 1.0 and minimum buffer ratio = 0.0), buffer space is allocated to traffic sources on a first-come-first-serve basis. The source with a high transmission rate is more likely to seize any free buffer space than the source with a low rate; therefore, the CLR for the fast source is lower than the CLR for the slow source. When certain buffer space is guaranteed for each traffic source, the expected CLR for each traffic source is described in Table 4.4.

Simulation Results

The simulation results are shown in Table 4.7. Dynamic buffer management works correctly as expected. In trial 1 where dynamic buffer management is not applied, source 5 has the lowest CLR since source 5 has the highest transmission rate. When 5% of the total buffer space is guaranteed for each traffic source in trial 2, the CLRs are exactly the same as what we have expected. This means that 5% of the total buffer space is enough to accommodate occasional bursts of cells from the Poisson traffic sources. In trial 3 where each traffic source can use at most 50% of the total buffer space, source 5 experiences the highest CLR since most of its cells are lost due to the limited usable buffer space. At the same time, source 1 and source 2 still experience cell loss because no minimum buffer space is guaranteed for them. In trial 4, each traffic source can use at most 20% of the total buffer space; therefore each traffic source is allocated 200-cell buffer space. The CLRs are the same as those in Table 4.4 because the 200-cell buffer space is enough for the bursts of the Poisson traffic sources.

Table 4.7: Simulation Results for Round Robin with Buffer Management

Trial	Source	Sent (cells)	Received (cells)	Lost (cells)	CLR (%)
1	1	49749	19603	30144	60.6
	2	100248	40742	59508	59.4
	3	200239	87712	112535	56.2
	4	300614	141939	158674	52.8
	5	399556	210004	189628	47.5
	overall	1050406	500000	550489	52.4
2	1	49749	49748	0	0.0
	2	100248	100251	0	0.0
	3	200239	116667	83554	41.7
	4	300614	116667	183964	61.2
	5	399556	116667	282904	70.8
	overall	1050406	500000	550422	52.4
3	1	49749	26441	23306	46.8
	2	100248	54655	45592	45.5
	3	200239	117892	82343	41.1
	4	300614	150506	150109	49.9
	5	399556	150506	249047	62.3
	overall	1050406	500000	550397	52.4
4	1	49749	49748	0	0.0
	2	100248	100251	0	0.0
	3	200239	116667	83582	41.7
	4	300614	116667	183949	61.2
	5	399556	116667	282931	70.8
	overall	1050406	500000	550462	52.4

4.2.5 Experiment 4: WFQ

This experiment tests the WFQ scheme in the per-VC queueing environment. Different weights are used for different sources. The experiment verifies that WFQ works properly.

Parameters

The source configuration is shown in Table 4.8. Four Poisson sources with the same transmission rate and different weights are used in this experiment. For the same reason as that stated in experiment 2, the maximum buffer ratio is set to 1.0, and the minimum buffer ratio is set to 0.05.

Table 4.8: Source Configuration for WFQ

Source	Rate (cells/second)	Weight
1	5000	1
2	5000	2
3	5000	3
4	5000	4

Expected Results

The expected CLRs can be calculated as follows. Each traffic source should get a bandwidth in proportion to its weight. The sum of the source weights is 10, so source 1 should get 1/10 of the total bandwidth, source 2 2/10, source 3 3/10. and source 4 4/10. Thus, source 1 gets 1000 cells/second, source 2 2000 cells/second, source 3 3000 cells/second, and source 4 4000 cells/second. The expected CLRs are obtained using the formula $(Rate - Allocated\ Bandwidth)/Rate$ and are shown in Table 4.9.

Table 4.9: Expected Results for WFQ

Source	Expected CLR (%)
1	80.0
2	60.0
3	40.0
4	20.0

Simulation Results

The simulation results are shown in Table 4.10. The simulation results match the expected results almost perfectly, which indicates that WFQ works properly.

Table 4.10: Simulation Results for WFQ

Source	Sent (cells)	Received (cells)	Lost (cells)	CLR (%)
1	250256	50000	200214	80.0
2	250325	100000	150363	60.1
3	250235	150000	100231	40.1
4	250331	200000	50330	20.1
Overall	1001147	500000	501138	50.1

4.3 Summary

This chapter provides validation of our per-VC queueing, fair queueing, and buffer management model. The validation is carried out in two respects: qualitatively and quantitatively. In qualitative validation, simulation results from FIFO, round robin in per-VC queueing, and WFQ in per-VC queueing are compared and analyzed, which qualitatively validates our model. In quantitative validation, simula-

tion results from FIFO, round robin in per-VC queueing, round robin with buffer management in per-VC queueing, and WFQ in per-VC queueing are compared with the expected results to validate our model quantitatively. The next chapter focuses on the experiments conducted to discover the effects of EPD, per-VC queueing and buffer management. Validation for packet-level discard is also presented in the next chapter.

Chapter 5

Experimental Results and Analysis

This chapter presents the performance study of TCP over ATM. We first describe the two scenarios used throughout the chapter. Then we present the performance metrics used to assess the performance of TCP over ATM. After that, simulation results are presented and analyzed in detail. Several conclusions drawn from the results are also presented.

5.1 Experimental Design

This section describes our simulation scenarios. Two scenarios are used in this chapter. The first scenario is used to study the effect of EPD, the effect of EPD and per-VC queueing[1], and the effect of buffer management. The second scenario is used to study the performance of per-VC queueing.

5.1.1 Scenario 1: The 10 TCP Sources Configuration

The network topology for the first scenario is shown in Figure 5.1. Ten identical TCP sources connect to their sinks through two switches and a shared link. The simulation parameters are as follows:

- The link length between the two switches is 1 km for the local area network (LAN) and 1000 km for the wide area network (WAN). In our experiments,

[1]Per-VC queueing in this chapter refers to the fair queueing algorithm in which a round robin scheduler is used to schedule the departure of cells from different VCs.

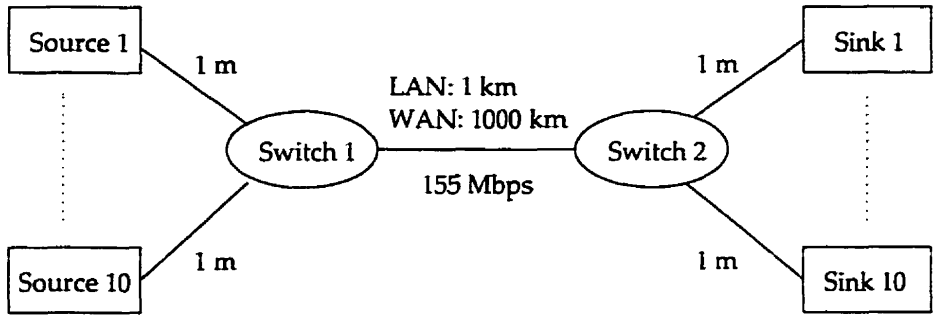

Figure 5.1: The 10 TCP Sources Configuration

all the other parameters are the same for LAN and WAN. All the other link lengths are 1 m.

- All the links have a bandwidth of 155.52 Mbps and a propagation delay of 3.7 μs per kilometer.

- All the sources are infinite TCP sources belonging to the same service class—UBR. The sources always send a packet if permitted by the TCP flow control window size.

- The TCP traffic is unidirectional. Only the source sends data. The sink sends only acknowledgements for the received data.

- The switch buffer size is varied from 1000 cells to 8000 cells, with a step of 1000 cells.

- Four different TCP packet sizes are used, namely 512, 1500, 4352, and 9140 bytes. 512-byte packet size is often used in IP networks. Ethernet networks use a packet size of 1500 bytes. 4352 is the packet size for Fiber Distributed Data

Interface (FDDI) networks, and 9140 is the default for IP over ATM [41]. In the ATM-TN simulator, different packet sizes are obtained by setting different maximum segment size (MSS) values.

- TCP timer granularity is set to 100 ms. This value is used to determine the timeout for retransmission. TCP provides a reliable byte stream transmission between two hosts through the use of several mechanisms. One mechanism is timeout and retransmission. Each end needs to acknowledge the data it receives from the other end. The data packets and acknowledgements may get lost during the transmission across the network (e.g., due to congestion). A timer is set when a TCP source sends data. If the data isn't acknowledged before the timeout occurs, the TCP source retransmits the data. The two commonly used values for the TCP timer granularity are 500 ms and 100 ms. We use 100 ms, which is also used by Goyal et al. [21].

- TCP maximum receiver window size is 64K bytes. TCP performance depends on the bandwidth-delay product which is the product of the bandwidth and round-trip time (RTT). The bandwidth-delay product is the capacity of the pipe between the sender and the receiver. In order to obtain good performance, the TCP receiver window size should not be less than the bandwidth-delay product [27, 42, 44]. 64K bytes window size is enough for the TCP sources to fill up the pipe in both the LAN and the WAN scenarios simulated.

- The transmission start times of all the sources are staggered 10 ms apart. This is to minimize conflicts between the sources when starting the simulations. Different start time intervals were tested. 10 ms interval produces the best

results.

- All the simulations are run for 150 seconds, including 60 seconds warmup period. The 60 seconds warmup period is long enough to put the simulation system into stable status. No statistics are gathered during this period. The simulation run time is also long enough to obtain meaningful results.

All the other optional parameters are set to their default values: TCP fast retransmit and recovery [26], delayed acknowledgement [6, 11], and the Nagle algorithm [37] are enabled; the TCP high performance extensions [27] are disabled.

5.1.2 Scenario 2: The 2 TCP Sources Configuration

Scenario 2 is used in the study of per-VC queueing performance. As seen in Figure 5.2, two TCP sources connect to their sinks through two switches. The simulation parameters are as follows:

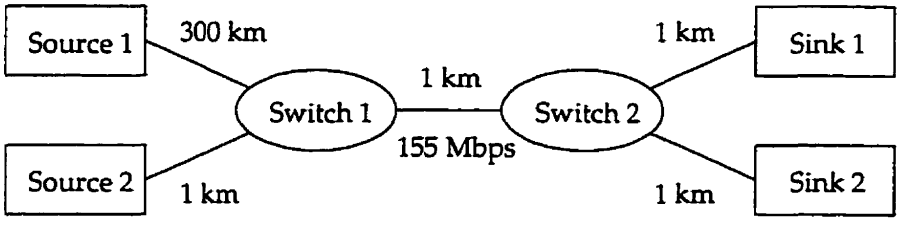

Figure 5.2: The 2 TCP Sources Configuration

- The link length between source 1 and switch 1 is 300 km. All the other link lengths are 1 km.

- The switch buffer size is 2000 cells.

- TCP maximum receiver window size is 32K bytes. This value is enough to fill up the pipe in this scenario.

All the other parameters are the same as those in scenario 1.

5.2 Performance Metrics

This section presents the performance metrics used to assess the performance of EPD, per-VC queueing, and buffer management.

5.2.1 Aggregate Effective Throughput

The aggregate effective throughput is the ratio of the actual TCP throughput to the maximum possible throughput.

- The actual TCP throughput is also called goodput. It is the total number of bytes delivered to the destination (not including retransmissions, lost segments, or duplicate data) divided by the total transmission time.

- The maximum possible throughput is the throughput achievable at the TCP layer running over ATM. It is determined by the segment size and the link bandwidth. Consider TCP over a 155.52 Mbps ATM network with an MSS of 9140 bytes. All the packets have 9140 bytes of data, 20 bytes of TCP header, and 20 bytes of IP header. The 9180 bytes are segmented and padded into ATM cells using AAL5. So 192 cells (i.e., 10,176 bytes) are transmitted over the ATM network for 9140 bytes of data. The maximum possible throughput is 9140/10176 = 89.8% of the link bandwidth, which is 139.7 Mbps.

The aggregate effective throughput is expressed as a percentage. Large values are preferred. 100% means the link bandwidth is fully utilized.

5.2.2 Fairness Index

In addition to providing high aggregate effective throughput, the network must allocate the bandwidth fairly among all the competing sources. Jain et al. [28] proposed an index of fairness which could be used to measure the fairness. The fairness index used in a network where all the traffic sources expect to obtain an equal share of the bandwidth is defined as

$$\frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

where x_i is the actual TCP throughput (i.e.. goodput) for source i and n is the number of TCP sources.

When each traffic source gains the same amount of throughput, the fairness index is 1, indicating all the sources share the bandwidth fairly. If one traffic source takes up all the bandwidth, then the fairness index is $1/n$. The lower the fairness index, the poorer the fairness. A fairness index close to 1 is desired.

5.2.3 Cell Loss Ratio

Cell loss ratio (CLR) is expressed as a ratio of the number of cells lost to the number of cells transmitted. It is often used as a metric to evaluate the performance of an ATM network. CLR increases as the network congestion increases.

5.2.4 Packet Loss Ratio

Packet loss ratio (PLR) is the number of packets lost divided by the total number of packets transmitted (including duplicate and retransmission packets). The basic transmission unit in the TCP layer is the segment or the TCP packet. Since we are studying the performance of TCP over ATM, PLR is more relevant to the network throughput and is expected to tell us more about the network performance than CLR.

5.2.5 Average Sender Window Size before Timeout

The sender window size is the number of bytes that the sender may send before it receives an acknowledgement and can send more bytes. It is the minimum of the congestion window size and the receiver's advertised window size. The congestion window and the advertised window will be discussed next. The "sender window size before timeout" is the window size when the timeout occurs. The "average sender window size before timeout" is the average of the sender window sizes before a timeout occurs.

- The congestion window is a flow control mechanism imposed by the sender. It is based on the sender's assessment of perceived network congestion [42]. When a connection is established, the congestion window size is initialized to the size of the maximum segment (i.e., MSS). Then one maximum segment is sent. When the acknowledgement for this segment arrives before the retransmission timer goes off, the congestion window size is increased by one MSS, and then two maximum segments are sent. Upon receiving two acknowledgments,

the congestion window size is increased by two MSSs, one for each acknowledgement, and then four maximum segments are sent. Thus, the congestion window size is increased exponentially until either a timeout occurs or the receiver's window size is reached [44]. This algorithm is called slow start [25]. When a timeout occurs, the congestion window size is reset to one MSS. When the congestion window size approaches half the size of the previous congestion window size when the timeout occurred, it is increased more slowly at one MSS per Round-Trip Time (RTT, defined in 5.2.6) instead of one MSS per acknowledgement. This is known as congestion avoidance [25]. When the receiver's window size is reached, the congestion window size stops growing and remains constant as long as no more timeouts occur and the receiver's window does not change in size [44].

- The receiver's advertised window is a flow control mechanism imposed by the receiver. It is the amount of available buffer space at the receiver for the current connection [42].

When a timeout occurs, the congestion window size is decreased. So is the sender window size. In most cases, the larger the average sender window size, the more packets transmitted. The average sender window size before timeout will help us further understand the performance of TCP.

5.2.6 Average Minimum Round-Trip Time

The Round-Trip Time (RTT) is the time interval between sending a segment and receiving an acknowledgment for it [25]. The RTT is used to determine the timeout,

and has some influence on the TCP performance as explained in Section 5.2.5. The RTT value can be estimated using Equation 5.1.

$$RTT \approx (CDS + PD + CTD) \times NCS + DD \tag{5.1}$$

where CDS is *Cell Delay in the Switch* which is the queueing delay for a cell in the switch buffer, PD is *Propagation Delay* which depends on the speed of signal in the transmission medium and the distance between the source and the destination, CTD is *Cell Transmission Delay* which depends on the link speed and the size of a cell, NCS is *The Number of Cells in a Segment*, and DD is *Delay at the Destination* which is the time interval between receiving a segment and sending an acknowledgment for it at the destination.

The minimum RTT can be defined as $(CDS + PD + CTD) \times NCS$, where CDS, PD, CTD, and NCS are the same as above. We will further understand the TCP performance by examining the average minimum RTT.

5.3 Effect of Early Packet Discard

Scenario 1 is used to study the performance of EPD. Both the LAN and WAN versions are tested. Different TCP packet sizes are used. Two conclusions can be drawn from the simulation results.

Conclusion 1: EPD improves throughput, with little impact on fairness. The simulation results of aggregate effective throughput for LAN and WAN are shown in Figure 5.3. The TCP packet size is 9140 bytes. Tail Drop represents the tail

drop scheme which discards any incoming cells after buffer overflow. PPD represents the partial packet discard scheme which discards the tail part of an incomplete packet after buffer overflow. EPD (90%) represents the early packet discard scheme with 90% of the total buffer space as its threshold; it discards an entire packet after 90% of the total buffer space is occupied.

The tail drop scheme drops cells when the buffer becomes full. One or more cells from a packet may be dropped while other cells from the same packet are transmitted to the destination. The incomplete packet will be retransmitted. The aggregate effective throughput for this scheme is low. PPD has higher throughput than the tail drop scheme, because cells from incomplete packets are discarded. EPD improves throughput to an optimal level since it minimizes the transmission of partial packets. The same trend of the performance of tail drop, PPD, and EPD is observed from the simulation results presented in [41], which roughly validates our model.

When EPD (or PPD) drops cells, there is no preference towards a particular source. Which cell will be dropped depends on the arrival order of the cells. So EPD (or PPD) has little impact on fairness. The fairness index for these experiments is shown in Table 5.1.

The simulation results of aggregate effective throughput and fairness index for the three other TCP packet sizes, 512, 1500 and 4352 bytes, show the same trend.

Since EPD improves throughput, PLR in EPD is expected to be lower than PLR in the tail drop scheme. However, the simulation results show that in most cases PLR in EPD is higher. How can we explain the throughput improvement of EPD?

Our simulation results show that in EPD traffic sources tend to push more data to the network; thus, in EPD, although PLR is higher, throughput is still higher

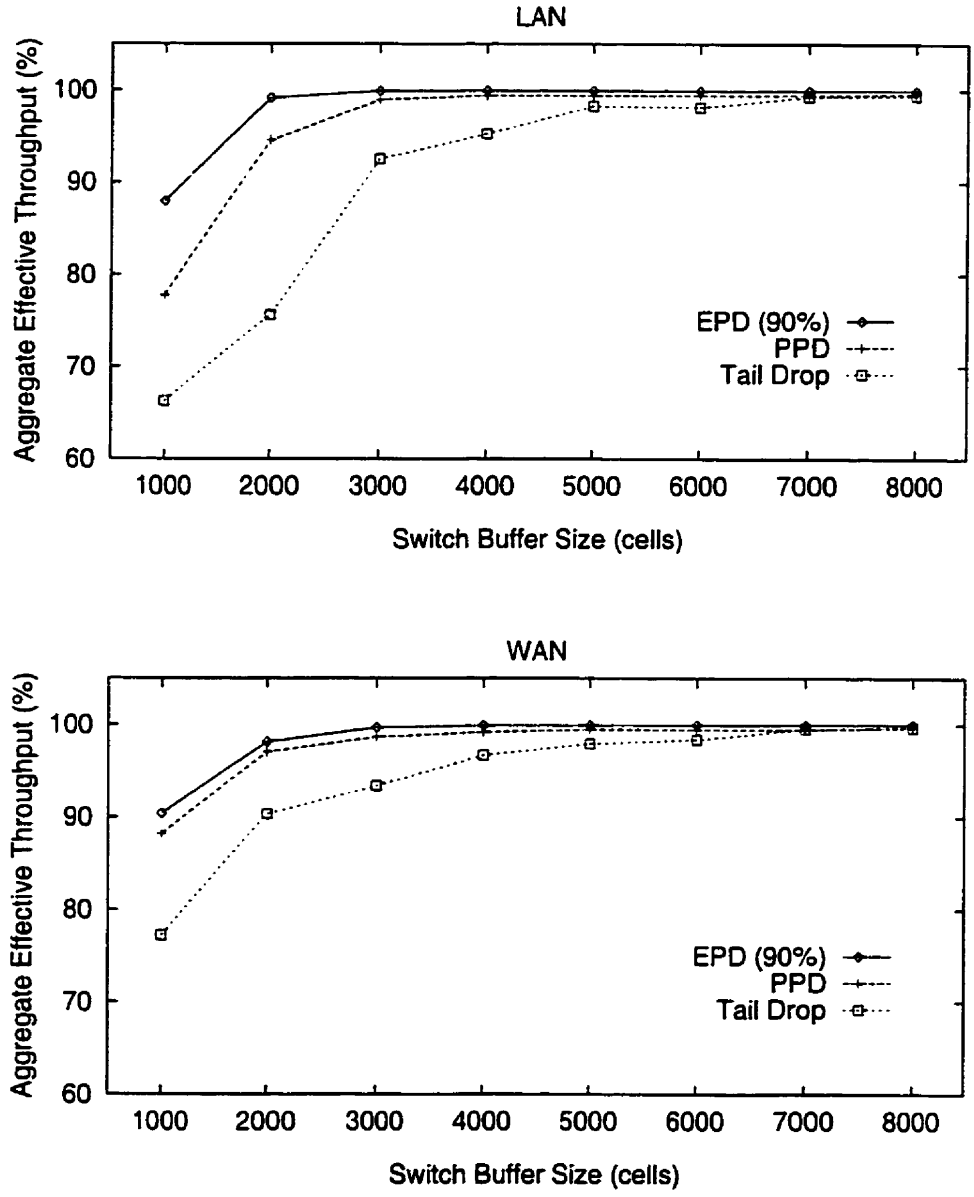

Figure 5.3: Aggregate Effective Throughput for LAN and WAN (TCP Packet Size = 9140 bytes)

Table 5.1: Fairness Index for LAN and WAN (TCP Packet Size = 9140 bytes)

Buffer Size	LAN			WAN		
(cells)	Tail Drop	PPD	EPD(90%)	Tail Drop	PPD	EPD(90%)
1000	0.99	0.97	0.92	0.97	0.99	0.98
2000	0.99	0.98	0.96	0.99	0.98	0.99
3000	0.98	0.98	0.98	0.98	0.99	0.99
4000	0.97	0.99	0.98	0.98	0.99	0.99
5000	0.99	0.99	0.99	0.99	0.99	0.99
6000	0.99	0.99	0.99	1.00	0.99	1.00
7000	0.99	0.99	0.99	0.99	0.99	0.99
8000	0.99	0.99	1.00	1.00	1.00	0.99

because more packets get through the network successfully. Results of CLR, PLR, and average sender window size (ASWS) are shown in Figure 5.4.

The average RTT in EPD is smaller than that in tail drop, because in EPD cells from damaged packets are dropped in the congested switch instead of the traffic sinks. So acknowledgements in EPD are received at a faster speed, which causes the sender window to open more quickly. Therefore the average sender window size before timeout in EPD is larger than that in tail drop.

From the above observation, the aggregate effective throughput and PLR are not strongly correlated. Higher aggregate effective throughput does not necessarily mean lower PLR. When the aggregate effective throughput is high, PLR may be high if more packets are transmitted over the network.

Another interesting observation is that although the CLR in EPD is much higher than that in tail drop, the PLR is just a little higher than that in tail drop. This is because tail drop drops cells from many different packets, while EPD intelligently

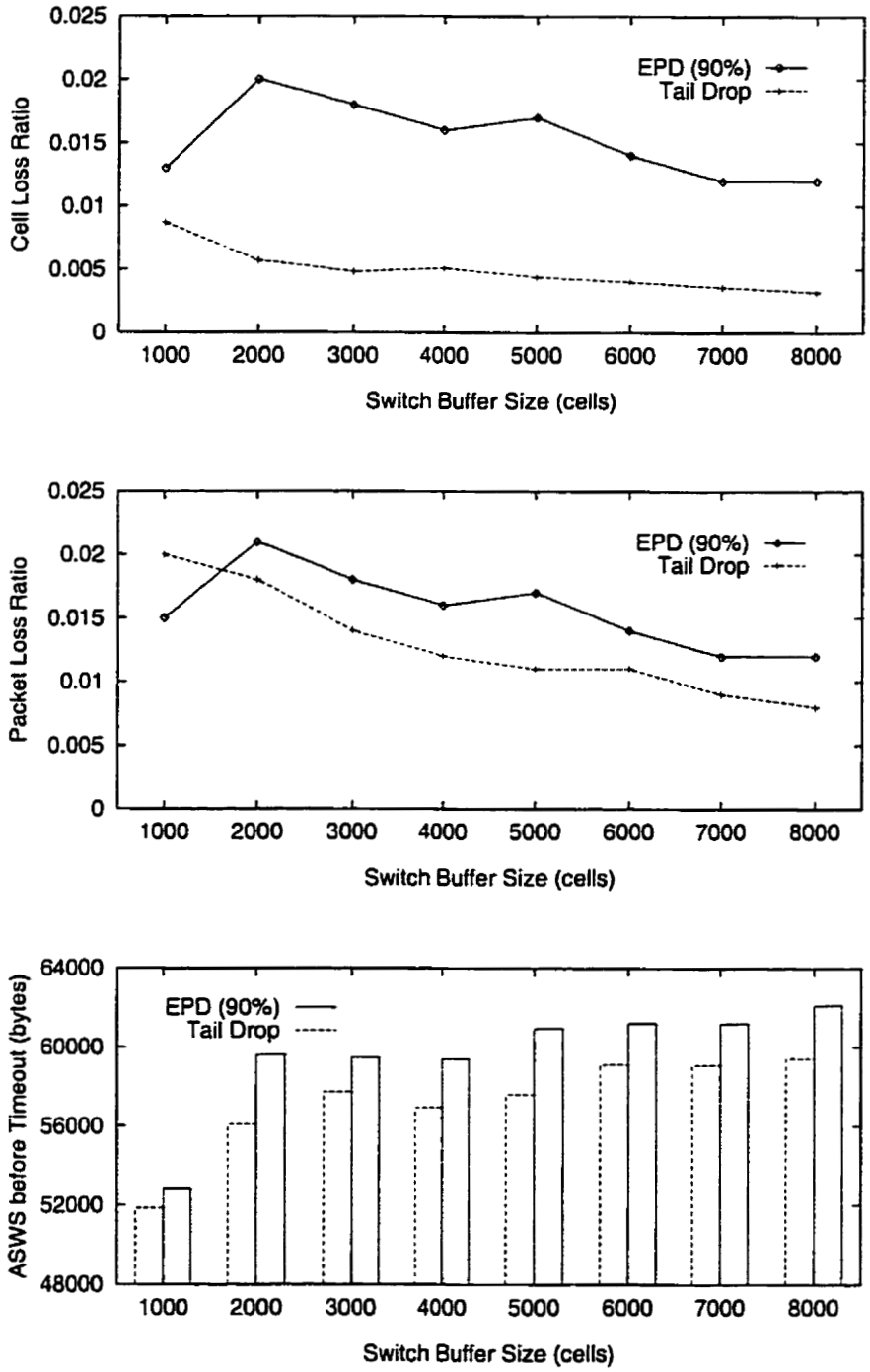

Figure 5.4: Cell Loss Ratio, Packet Loss Ratio, and Average Sender Window Size (ASWS) before Timeout (LAN, TCP Packet Size = 9140 bytes)

drops cells to minimize the number of incomplete packets.

Conclusion 2: When buffer size is small, throughput is sensitive to the combination of EPD threshold and packet size; when buffer size is large, these parameters make little difference to the throughput.

The simulation results of aggregate effective throughput of EPD with different thresholds and different packet sizes are shown in Figure 5.5. As seen from the figure, the aggregate effective throughput is sensitive to the different EPD thresholds and different packet sizes when the buffer size is small (e.g., less than 2000 cells in this case). We may choose different EPD thresholds for different buffer sizes and packet sizes to achieve good performance. When the buffer size is large (e.g., over 3000 cells in this case), the different EPD thresholds and the different TCP packet sizes make little difference to the throughput.

When the TCP packet size is 9140 bytes and the switch buffer size is 1000 cells, EPD with 70% threshold produces very low throughput. In this case, the average buffer occupancy is 42 cells, and the maximum buffer occupancy is 882 cells. This means the limited small buffer space is not fully utilized because of the low EPD threshold. So the aggregate effective throughput is low.

When the TCP packet size is 9140 and the switch buffer size is 1000 cells, the aggregate effective throughput at a 95% threshold is lower than that at a 90% threshold. This may indicate that in this case, 5% buffer space left for transmitting partial transmitted packets is not enough for EPD to gain better throughput.

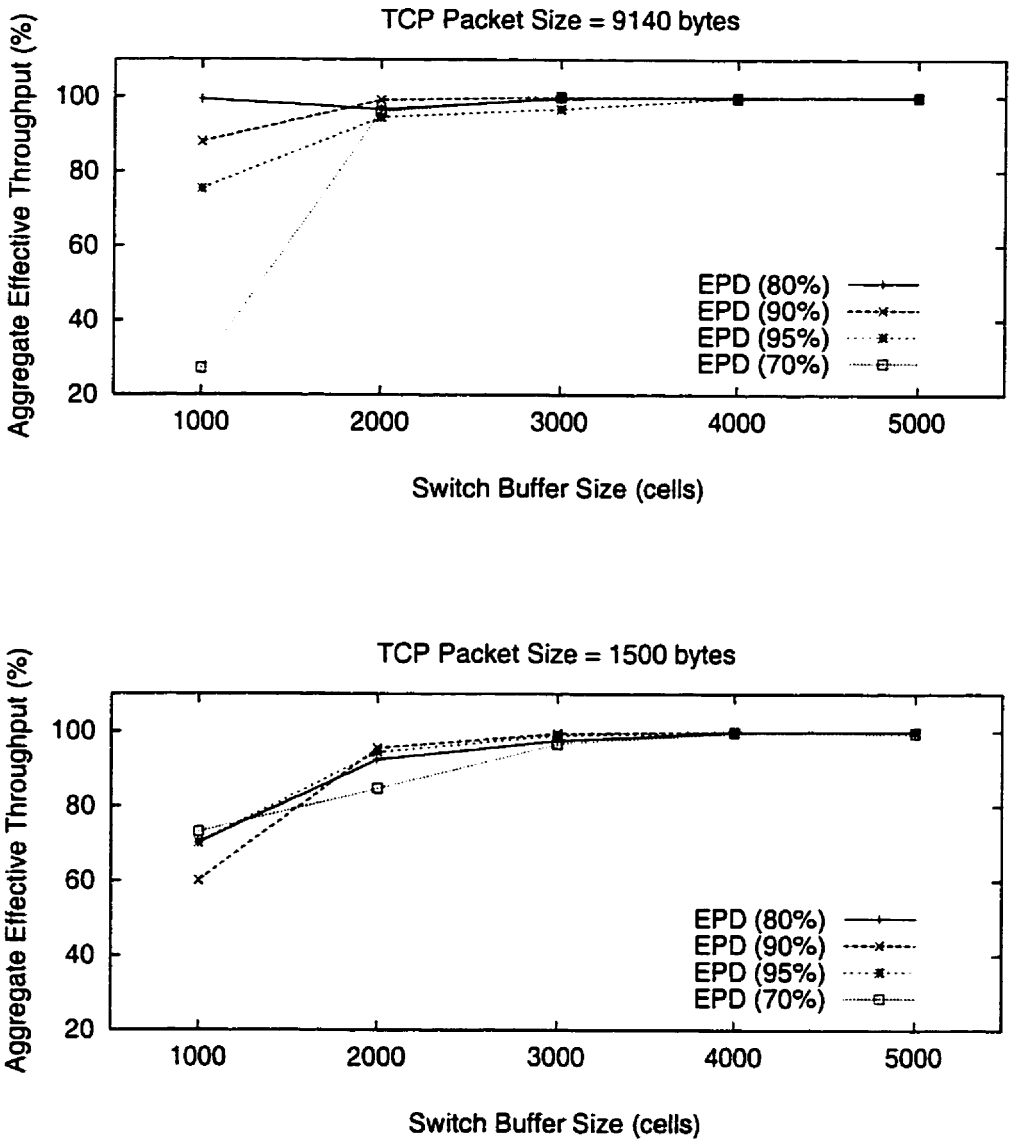

Figure 5.5: Aggregate Effective Throughput of EPD with Different Thresholds and Different TCP Packet Sizes (LAN)

5.4 Effect of Per-VC Queueing

Scenario 2 is used to study the effect of per-VC queueing. Five different TCP packet sizes are used, namely 512, 1500, 4352, 7000, and 9140 bytes. One conclusion can be drawn from the simulation results.

Conclusion 3: Per-VC queueing generally improves fairness in situations where fairness is poor without per-VC queueing.

In scenario 2, source 1 is farther away from switch 1 than source 2, so source 1 has a longer RTT than source 2. The duration of source 2's slow start phase is shorter since window adjustment is done on the RTT basis. Source 2 sends more data than source 1. Thus, source 2 gets more throughput than source 1. The simulation results for different TCP packet sizes are shown in Figure 5.6.

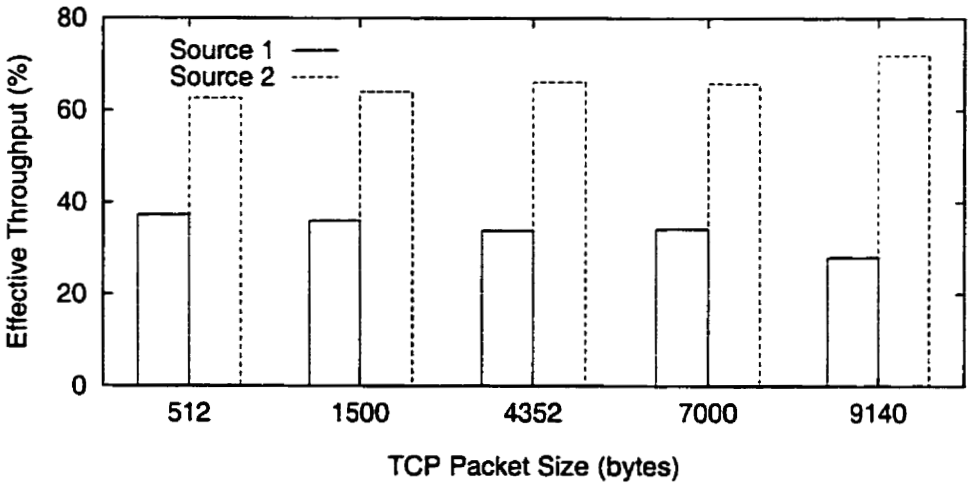

Figure 5.6: Effective Throughput (No Per-VC Queueing)

As shown in Figure 5.6, source 2 takes up nearly two-thirds of the total bandwidth, so fairness is very poor. The purpose of per-VC queueing is to improve

fairness. In per-VC queueing, there is one queue for each VC. So the source which sends data faster only increases its own queue length. All the queues from different sources are serviced in turn. The results of the simulation using per-VC queueing are shown in Figure 5.7.

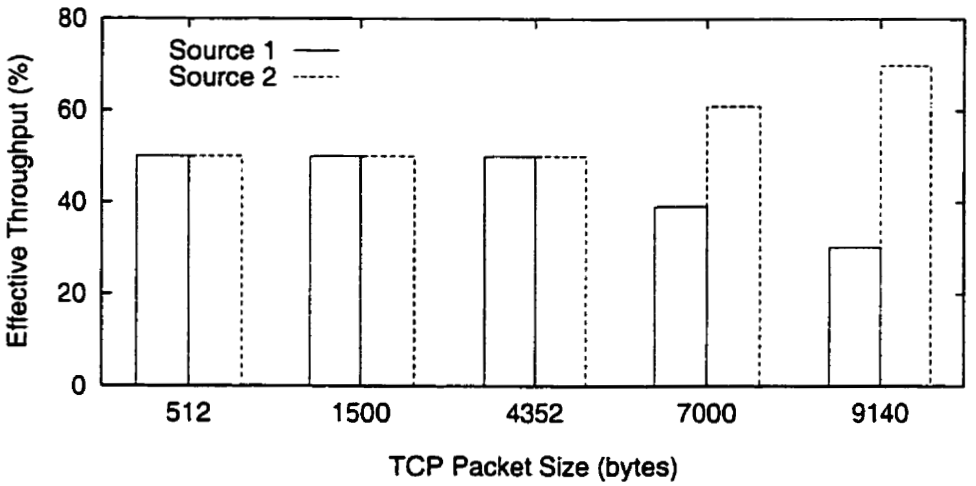

Figure 5.7: Effective Throughput (Per-VC Queueing)

Per-VC queueing improves the fairness ideally when the TCP packet size is 512, 1500 and 4352 bytes. It slightly improves the fairness when the TCP packet size is 7000 and 9140 bytes. We try to explain this from the RTT point of view. Per-VC queueing indirectly decreases the RTT of the faraway source, so the RTT difference between source 1 and source 2 is decreased, and fairness is improved. The estimated average minimum RTT are shown in Table 5.2 and 5.3. From the estimated results, when the packet size is 512, 1500, and 4352 bytes, the RTT difference decreases to zero in per-VC queueing. When the packet size is 7000 and 9140 bytes, the RTT difference is less than if per-VC queueing is not used, but is still prominent.

Table 5.2: Average Minimum Round-Trip Time (No Per-VC Queueing)

TCP Packet Size (bytes)	512	1500	4352	7000	9140
Source 1 Cell Delay in the Switch (ms)	3.0	2.5	1.8	1.2	0.8
Source 2 Cell Delay in the Switch (ms)	3.0	2.5	1.8	1.3	0.7
Source 1 Propagation Delay (ms)	2.2	2.2	2.2	2.2	2.2
Source 2 Propagation Delay (ms)	0.0	0.0	0.0	0.0	0.0
Each Source Cell Transmission Delay (ms)	0.0	0.0	0.0	0.0	0.0
The Number of Cells in a TCP Packet	12	33	92	147	192
Source 1 Minimum RTT (ms)	62.4	155.1	368.0	499.8	576.0
Source 2 Minimum RTT (ms)	36.0	82.5	165.6	191.1	134.4
RTT Difference (ms)	26.4	72.6	202.4	308.7	441.6

Table 5.3: Average Minimum Round-Trip Time (Per-VC Queueing)

TCP Packet Size (bytes)	512	1500	4352	7000	9140
Source 1 Cell Delay in the Switch (ms)	1.6	1.0	0.3	0.4	0.5
Source 2 Cell Delay in the Switch (ms)	3.8	3.2	2.5	1.3	0.7
Source 1 Propagation Delay (ms)	2.2	2.2	2.2	2.2	2.2
Source 2 Propagation Delay (ms)	0.0	0.0	0.0	0.0	0.0
Each Source Cell Transmission Delay (ms)	0.0	0.0	0.0	0.0	0.0
The Number of Cells in a TCP Packet	12	33	92	147	192
Source 1 Minimum RTT (ms)	45.6	105.6	230.0	382.2	518.4
Source 2 Minimum RTT (ms)	45.6	105.6	230.0	191.1	134.4
RTT Difference (ms)	0.0	0.0	0.0	191.1	384.0

5.5 Effect of Early Packet Discard and Per-VC Queueing

Much research has been carried out to study EPD [9, 10, 41] and to study per-VC queueing [12, 13, 38]. No known research has studied the performance of the combination of these two algorithms. In this section. we examine the effect of EPD and per-VC queueing. Scenario 2 is used, and an encouraging conclusion is drawn.

Conclusion 4: When EPD and per-VC queueing are used together, typically both throughput and fairness can be improved.

The simulation results of aggregate effective throughput for LAN and WAN when the TCP packet sizes are 9140 and 1500 bytes are shown in Figure 5.8 and 5.9. Table 5.4 and 5.5 show the fairness index results. From these results, we can see that in most cases. the combination of per-VC queueing and EPD improves throughput as well as fairness. This is because EPD improves throughput and at the same time per-VC queueing improves fairness.

In Figure 5.9, the aggregate effective throughput of tail drop for LAN when the switch buffer size is 1000 cells is close to 100%. The fairness index at this point is low at 0.1, which means only one traffic source takes up all the bandwidth. What seems to be happening is that one source is monopolizing use of the buffer so that no other source is able to get a packet in. Due to the design of TCP congestion control, sources that get packets through successfully are trying to push more through, while those who lost packets get timeouts and reduce their rates of sending. In this case, one source ends up winning all the bandwidth. We tried different start times for some of the traffic sources and found different traffic sources gained all the bandwidth each time. So it is a state that happens coincidentally.

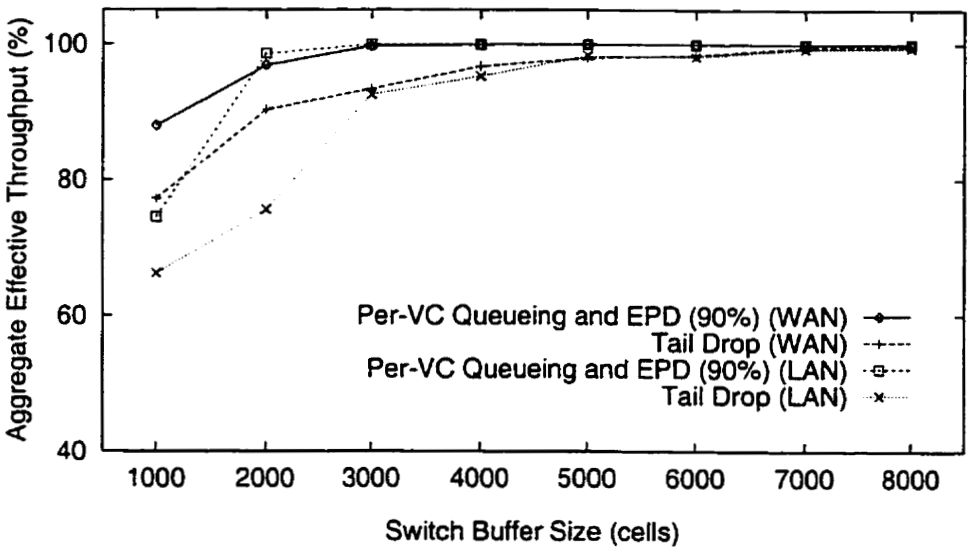

Figure 5.8: Aggregate Effective Throughput (TCP Packet Size = 9140 bytes)

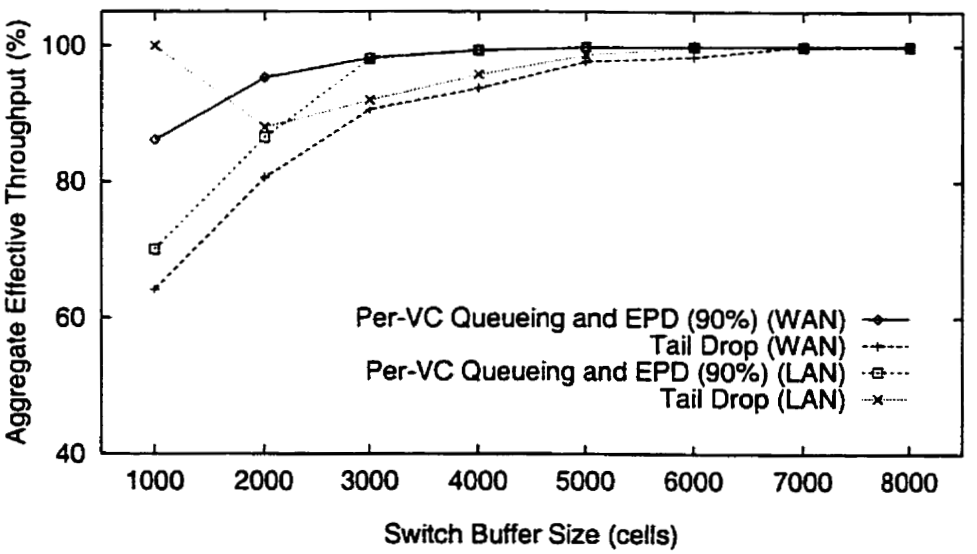

Figure 5.9: Aggregate Effective Throughput (TCP Packet Size = 1500 bytes)

Table 5.4: Fairness Index (TCP Packet Size = 9140 bytes)

Buffer Size (cells)	LAN		WAN	
	Tail Drop	Per-VC Queueing and EPD (90%)	Tail Drop	Per-VC Queueing and EPD (90%)
1000	0.99	0.98	0.97	1.00
2000	0.99	0.98	0.99	0.99
3000	0.98	0.98	0.98	0.99
4000	0.97	0.98	0.98	0.99
5000	0.99	0.99	0.99	0.99
6000	0.99	0.99	1.00	1.00
7000	0.99	0.99	0.99	0.99
8000	0.99	0.99	1.00	1.00

Table 5.5: Fairness Index (TCP Packet Size = 1500 bytes)

Buffer Size (cells)	LAN		WAN	
	Tail Drop	Per-VC Queueing and EPD (90%)	Tail Drop	Per-VC Queueing and EPD (90%)
1000	0.10	0.98	0.98	0.99
2000	0.98	0.99	0.99	0.99
3000	0.98	1.00	0.96	0.98
4000	0.95	0.99	0.99	0.99
5000	0.98	0.99	0.85	1.00
6000	0.98	0.99	0.83	0.98
7000	0.96	1.00	0.81	0.98
8000	0.89	0.98	0.74	0.98

Special Case: When the TCP packet size is 512 bytes, the results do not follow conclusion 4.

When EPD and per-VC queueing are used together, typically both throughput and fairness can be improved. However, from all the cases we examined (i.e., 512, 1500, 4352, and 9140 byte TCP packet sizes), we found when the TCP packet size was 512 bytes, the results did not follow the conclusion. The simulation results for the packet size of 512 bytes are shown in Figure 5.10 and Table 5.6.

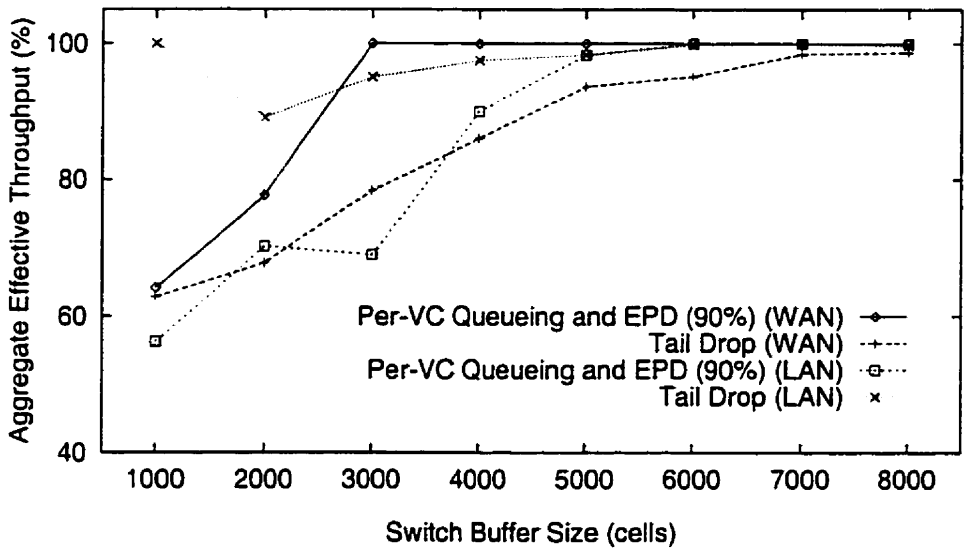

Figure 5.10: Aggregate Effective Throughput (TCP Packet Size = 512 bytes)

In LAN, the combination of EPD and per-VC queueing improves fairness, but it decreases throughput. In WAN, the combination of EPD and per-VC queueing improves throughput, but it decreases fairness. The results may have something to do with the small TCP packet size. Simulations of other small TCP packet sizes are desired in order to confirm this hypothesis. Due to the time limit of this thesis, we only study the 512 byte TCP packet size. However, performance of other TCP

Table 5.6: Fairness Index (TCP Packet Size = 512 bytes)

Buffer Size (cells)	LAN		WAN	
	Tail Drop	Per-VC Queueing and EPD (90%)	Tail Drop	Per-VC Queueing and EPD (90%)
1000	0.10	0.98	0.96	0.93
2000	0.96	0.96	0.97	0.87
3000	0.98	0.99	0.96	0.36
4000	0.96	0.98	0.96	0.37
5000	0.94	0.99	0.95	0.37
6000	0.95	1.00	0.95	0.51
7000	0.96	1.00	0.97	0.49
8000	0.96	1.00	0.94	0.48

packet sizes needs to be explored in future work. When the TCP packet size is 512 bytes, it only takes 12 cells to transmit a packet. The fragmentation with the tail drop scheme is not serious. Since the aggregate effective throughput in LAN is already high at more than 90%, EPD with 90% threshold may not improve the throughput. Per-VC queueing also does not improve fairness when the packet size is 512 bytes. Table 5.7 and Figure 5.11 and 5.12 are used to illustrate this observation.

As seen in Table 5.7, per-VC queueing improves throughput, but it decreases fairness. Consider per-VC queueing when the buffer size is 3000 cells. The aggregate effective throughput is 100%, and the fairness index is 0.39. Figure 5.11 and 5.12 show the effective throughput for each traffic source and the congestion window size after 3 seconds simulation run to help us understand what is happening. We see three sources get most of throughput, while others only get a small proportion of the total throughput. This may be caused by TCP congestion control. The congestion window size for source 1, 3, and 8 remains at 64K bytes after 3 seconds simulation

run, while the congestion window size for other sources remains at a low value.

Table 5.7: Aggregate Effective Throughput and Fairness Index for Tail Drop (WAN, TCP Packet Size = 512 bytes)

Buffer Size	No Per-VC Queueing		Per-VC Queueing	
(cells)	Throughput (%)	Fairness	Throughput(%)	Fairness
1000	62.86	0.96	100.0	0.24
2000	67.83	0.97	87.88	0.48
3000	78.42	0.96	**100.0**	**0.39**
4000	86.09	0.96	100.0	0.40
5000	93.65	0.95	100.0	0.39
6000	95.15	0.95	100.0	0.53
7000	98.46	0.97	100.0	0.51
8000	98.69	0.94	100.0	0.40

5.6 Effect of Buffer Management

Buffer management decides which packet gets into the buffer. It can be used with a scheduling scheme to provide rate guarantees to individual flows. This section studies the effect of the dynamic buffer allocation scheme presented in Section 3.4. Scenario 1 is used. Both LAN and WAN are tested, and different TCP packet sizes are used. Two conclusions are drawn from the simulation results.

Conclusion 5: When buffer size is small and effective throughput is low, per-VC queueing with buffer management does improve the throughput.

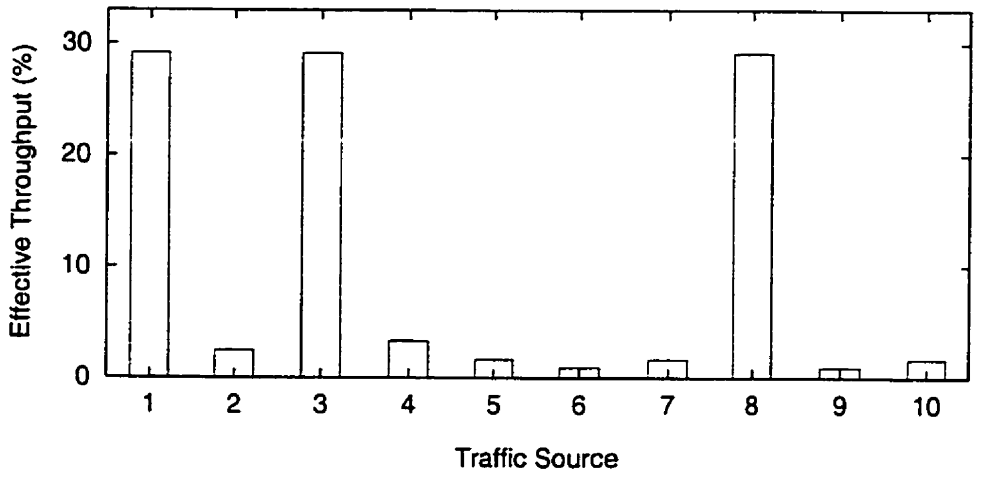

Figure 5.11: Effective Throughput (WAN, TCP Packet Size = 512 bytes, Buffer Size = 3000 cells)

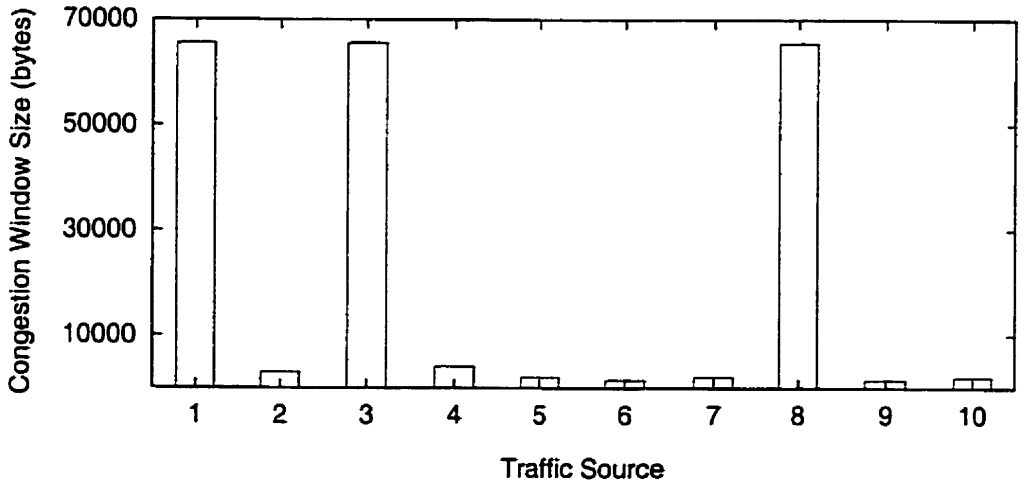

Figure 5.12: Congestion Window Size after 3 Seconds Simulation Run (WAN, TCP Packet Size = 512 bytes, Buffer Size = 3000 cells)

In Figure 5.13, Per-VC (max=0.5) represents per-VC queueing where at most 50% of the total buffer space can be used by a VC. Per-VC (max=0.5, min=0.1) represents per-VC queueing where at most 50% of the total buffer space can be used by a VC and at least 10% of the total buffer space is guaranteed to a VC. As shown in the figure, per-VC queueing with buffer management improves throughput when the switch buffer size is small (i.e., 1000 and 2000 cells) and the aggregate effective throughput is low.

The PLR and the average sender window size before timeout shown in Figure 5.13 help us understand the throughput improvement. When the switch buffer size is 1000 cells, per-VC queueing with buffer management tends to push more data through the network although it has a high PLR; so the aggregate effective throughput is improved. When the switch buffer size is 2000 cells, per-VC queueing with buffer management has a low PLR, which improves the aggregate effective throughput.

Table 5.8 shows the results for a different TCP packet size, 512 bytes. We can see per-VC queueing with buffer management improves throughput. At the same time, it improves fairness. This is explained in Conclusion 6.

Table 5.8: Aggregate Effective Throughput and Fairness Index for Tail Drop (LAN, TCP Packet Size = 512 bytes)

Buffer Size (cells)	1000		2000	
	Throughput (%)	Fairness	Throughput(%)	Fairness
No Per-VC Queueing	100.0	0.10	89.13	0.96
Per-VC Queueing	100.0	0.10	99.33	1.00
Per-VC (max=0.5)	100.0	1.00	100.0	1.00

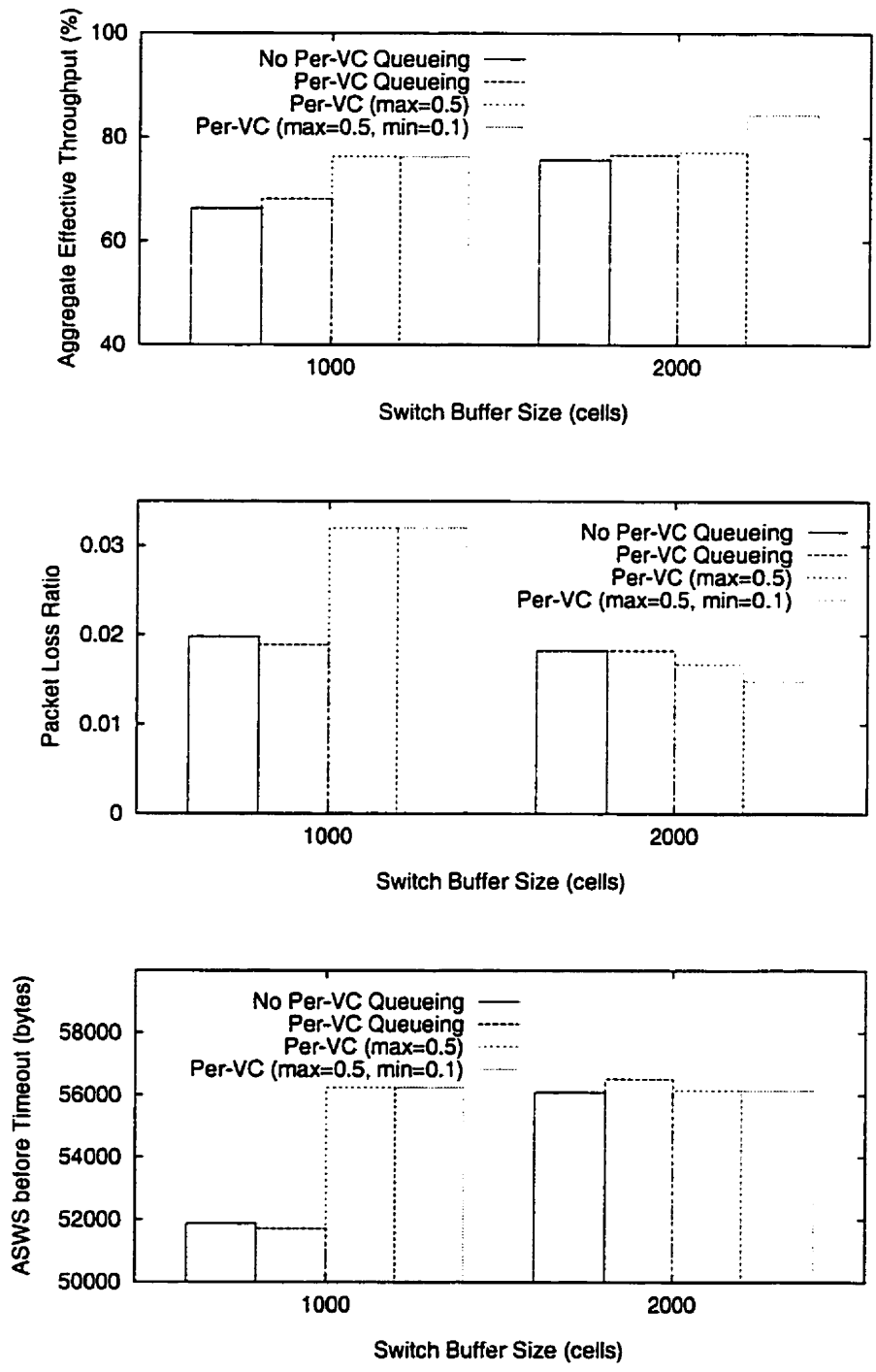

Figure 5.13: Aggregate Effective Throughput, Packet Loss Ratio, and Average Sender Window Size (ASWS) before Timeout (LAN, TCP Packet Size = 9140 bytes)

Conclusion 6: When buffer size is small and fairness is very bad, per-VC queueing with buffer management does improve fairness.

As seen in Table 5.9, when the switch buffer size is 1000 cells, the aggregate effective throughput of tail drop for 512-byte TCP packets and 1500-byte TCP packets is 100%. The fairness index is low at 0.1, which means one traffic source hogs all the bandwidth. Per-VC queueing without buffer management cannot help to improve fairness in this case, because all the buffer space is occupied by that greedy traffic source. Per-VC queueing with buffer management is expected to improve fairness since the maximum buffer size limits the amount of buffer space that can be used by a single traffic source. Thus, the greedy source can be prevented from taking up all the bandwidth. Fairness is improved.

Table 5.9: Aggregate Effective Throughput and Fairness Index for Tail Drop (LAN, Buffer Size = 1000 cells)

Packet Size (bytes)	512		1500	
	Throughput (%)	Fairness	Throughput(%)	Fairness
No Per-VC Queueing	100.0	0.10	100.0	0.10
Per-VC Queueing	100.0	0.10	100.0	0.10
Per-VC (max=0.5)	100.0	1.00	79.30	0.97
Per-VC (max=0.4)	100.0	1.00	84.99	0.95

When the TCP packet size is 512 bytes, per-VC queueing with a maximum buffer size of 500 cells and 400 cells improves the fairness index to 1. When the TCP packet size is 1500, the fairness index is close to 1.

5.7 Summary

This chapter studies the performance of TCP over ATM. The simulation scenarios and performance metrics are presented first. We then examine the effects of EPD, per-VC queueing, EPD and per-VC queueing, and buffer management. Experimental results are presented and analyzed. Six conclusions are drawn from our simulation results:

1. EPD improves throughput, with little impact on fairness.
2. When buffer size is small, throughput is sensitive to the combination of EPD threshold and packet size; when buffer size is large, these parameters make little difference to the throughput.
3. Per-VC queueing generally improves fairness in situations where fairness is poor without per-VC queueing.
4. When EPD and per-VC queueing are used together, typically both throughput and fairness can be improved.
5. When buffer size is small and effective throughput is low, per-VC queueing with buffer management does improve the throughput.
6. When buffer size is small and fairness is very bad, per-VC queueing with buffer management does improve fairness.

In short, EPD improves throughput, and per-VC queueing improves fairness. Buffer management improves the performance of per-VC queueing, both in throughput and in fairness. When switch buffer size is small, PPD and EPD may be used to

improve throughput. When switch buffer size is large, throughput is already high. So packet-level discard may not be necessary.

The next chapter presents the thesis conclusions and discusses possible future work.

Chapter 6

Conclusions and Future Work

This final chapter contains a summary of the thesis and a list of major conclusions and suggestions. Some suggestions for future work are also presented.

6.1 Summary

The research carried out in this thesis achieved the three main goals of the thesis. First, a simulation model for several congestion control algorithms was designed and implemented. Second, performance of the different congestion control algorithms was evaluated. Third, suggestions were provided to improve the performance of TCP over ATM.

TCP is a connection oriented protocol that provides a reliable byte stream transmission between two hosts. IP is a connectionless network layer protocol that provides unreliable data transmission. ATM is a connection-oriented technology which is often used as underlying network technology. TCP over IP over ATM can be very inefficient. When an ATM network experiences congestion, the performance of TCP over ATM is very poor due to the length mismatch between TCP packets and ATM cells. Several congestion control algorithms were proposed to improve the performance of ATM networks. Per-VC queueing, buffer management, and packet-level discard were studied in this thesis. These schemes were designed and implemented as part of the ATM-TN simulator, a simulator designed for ATM network design

and analysis.

A simple output queueing switch with one FIFO queue for each output port cannot provide fair allocation of bandwidth to different traffic sources. A greedy source may take up all the bandwidth, preventing other sources from getting their share of the bandwidth. Per-VC queueing was proposed to solve this problem. It provides different queues for different VCs. The greedy source only increases its own queue length. With fair queueing, all the queues are serviced in turn. Thus, all the traffic sources are given the same priority. WFQ associates each traffic source with a weight. A source with a larger weight is serviced faster than a source with a smaller weight. Therefore, different priorities can be provided to different traffic sources through the use of different weights. The thesis compared the results from FIFO queueing and per-VC queueing. WFQ was also tested.

Buffer management decides if an arriving packet should be allowed entry into the buffer. It can be applied with any scheduling scheme in order to provide rate guarantees. A dynamic buffer allocation scheme to be used with per-VC queueing was investigated and validated in this thesis.

The main reason for the low throughput of TCP over ATM when an ATM network experiences congestion is fragmentation. TCP packets are broken into 53-byte ATM cells for transmission over the ATM network. Those cells are reassembled into the TCP packets at the destination. One cell loss causes the whole packet to be retransmitted later. The tail drop scheme which discards any incoming cells after buffer overflow results in low throughput. There is a strong correlation between TCP packet size and throughput. The larger the TCP packet size, the larger the number of wasted cells that the congested link transmits when a single cell from a packet is

dropped, and the lower the throughput. Packet-level discard was proposed to discard cells intelligently to maximize throughput: PPD discards the tail part of incomplete packets to improve throughput; EPD discards entire packets before buffer overflow to improve throughput to an optimal level. The results from tail drop, PPD, and EPD were compared in the thesis.

6.2 Thesis Contributions

This thesis provided a detailed study of per-VC queueing, buffer management, and EPD. The main contributions of this thesis are as follows:

- Designed, implemented, and validated several congestion control mechanisms, including per-VC queueing, fair queueing, buffer management, and packet-level discard in the ATM-TN simulator.

- Studied the performance of per-VC queueing on TCP over ATM. Previous research studied the performance of per-VC queueing over packet networks [12, 13]. However, no published research examined the performance of per-VC queueing on TCP over ATM. This thesis studied the performance of per-VC queueing on TCP over ATM.

- Investigated a simple buffer management scheme, a dynamic buffer allocation scheme with two parameters, to be used with per-VC queueing.

- Examined both the throughput and the fairness performance of EPD. Previous research focused on the throughput performance of EPD. Since fairness is also an important performance metric in a congested network, we also studied

the fairness performance. This thesis also studied the effect of different EDP thresholds, which was seldom studied in former research.

- Studied the performance of the combination of EPD and per-VC queueing on TCP over ATM. No known research was carried out in this area before.

Several conclusions and suggestions drawn from the research are presented in the following section.

6.3 Conclusions and Suggestions

The following conclusions and suggestions can be drawn from the experience in the design, implementation, and experimentation of per-VC queueing, fair queueing, buffer management, and packet-level discard.

1. Per-VC queueing improves fairness. It generally improves fairness in situations where fairness is poor without per-VC queueing.

2. Weighted fair queueing provides different priorities to different traffic sources. Different priorities are provided through the use of different weights. Sources with larger weights are serviced faster than sources with smaller weights.

3. Buffer management can improve the performance of per-VC queueing. When buffer size is small and effective throughput is low, per-VC queueing with buffer management can improve the throughput. When buffer size is small and fairness is very bad, per-VC queueing with buffer management can improve fairness.

4. PPD and EPD improve throughput, with little impact on fairness. PPD has a higher throughput than tail drop. EPD has a higher throughput than PPD. Fairness is not affected.

5. EPD threshold affects the performance of TCP over ATM. When buffer size is small, throughput is sensitive to different EPD thresholds and different packet sizes. In this case, the EPD threshold should be tuned to achieve high throughput.

6. When EPD and per-VC queueing are used together, typically both throughput and fairness can be improved. Since EPD improves throughput and per-VC queueing improves fairness, the combination of these two improves both throughput and fairness.

7. Throughput has a strong correlation to the TCP packet size. When the packet size is large, throughput is poor. In this case, PPD and EPD can greatly improve the throughput. Throughput is greater with smaller packet sizes. When the packet size is very small, throughput is high, and PPD and EPD may be not needed.

8. Switch buffer size plays an important role in the performance of TCP over ATM. When the buffer size is small, throughput is poor, and sometimes fairness is poor too. In this case, PPD and EPD can improve the throughput, and buffer management can improve fairness. Throughput increases when buffer size increases. When the buffer size is large, the performance of TCP over ATM is good, and we don't need to use PPD or EPD.

6.4 Future Work

Future work can be carried out in two directions: 1) Enhance the simulation model (i.e., ATM-TN). 2) Further study the performance of TCP over ATM.

6.4.1 Simulation Model Enhancement

The simulation model we have built provides several schemes related to per-VC queueing, buffer management, fair queueing, and packet-level discard. We could further enhance our simulation model by adding the following functions:

1. Per-VC buffer management to be used with FIFO queueing. One disadvantage with FIFO queueing is unfair allocation of bandwidth. Since all the traffic sources are merged into one FIFO queue, a traffic source sending packets at a high rate can get a large fraction of the bandwidth. If we control the buffer space that a traffic source can use, then fairness can be improved. The implementation of per-VC buffer management may be easier than that of per-VC queueing. If per-VC buffer management can achieve the same performance as per-VC queueing, then we don't need to use per-VC queueing.

2. Weighted Round Robin (WRR) [31] to be used with per-VC queueing. Since round robin allocates bandwidth fairly when all the traffic sources have equal weights, WRR, which allocates bandwidth to VCs in proportion to the prescribed weights, may have similar behavior to WFQ. If so, we may use WRR instead of WFQ because it is much easier to implement WRR than WFQ.

6.4.2 Further Performance Study

In this thesis, we provide a detailed study of the performance of several congestion control schemes on TCP over ATM, including per-VC queueing, fair queueing, buffer management, and packet-level discard. The study could be continued in the following two areas:

1. Performance of WFQ on TCP over ATM. In this thesis, we studied the performance of WFQ using Poisson traffic. WFQ can provide different priorities to difference Poisson sources. Does WFQ perform well on TCP over ATM? A study may be carried out to explore the performance of WFQ on TCP over ATM.

2. Performance of per-VC queueing and EPD on TCP over ATM. In Chapter 5, we studied the performance of per-VC queueing and EPD on TCP over ATM. In most cases, the combination of per-VC queueing and EPD improves throughput as well as fairness. However, there is a special case (i.e., TCP packet size = 512 bytes) in which simulation results do not follow the conclusion. More work needs to be done to discover the performance of per-VC queueing and EPD on TCP over ATM. TCP congestion control may be studied in detail to further understand the performance.

Bibliography

[1] ANSI. AAL5 - a new high speed data transfer AAL. ANSI T1S1.5 91-449, November 1991.

[2] G. Armitage and K. Adams. Packet reassembly during cell loss. *IEEE Network Magazine*, 7(5):26–34, September 1993.

[3] J. Banks, J. S. Carson, and B. L. Nelson. *Discrete-Event System Simulation.* Prentice Hall, second edition, 1996.

[4] J. C. R. Bennett and H. Zhang. WF^2Q: Worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM'96*, pages 120–128, San Francisco, CA, March 1996.

[5] A. Bianco. Performance of the TCP protocol over ATM networks. In *Proceedings of the 3rd International Conference on Computer Communications and Networks*, pages 170–177, San Francisco, CA, September 1994.

[6] R. Braden. Requirements for internet hosts – communication layers. RFC 1122, October 1989.

[7] K. M. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, September 1979.

[8] K. M. Chandy and J. Misra. Asynchronous distributed simulation via a sequence of parallel computations. *Communications of the ACM*, 24(11):198–205,

November 1981.

[9] K. Cheon and S. S. Panwar. Early selective packet discard for alternating resource access of TCP over ATM-UBR. In *Proceedings of the 22nd IEEE Annual Conference on Computer Networks*, pages 306–316, Minneapolis, November 1997.

[10] K. Cheon and S. S. Panwar. On the performance of ATM-UBR with early selective packet discard. In *Proceedings of IEEE ICC'98*, Atlanta, June 1998.

[11] D. Clark. Window and acknowledgment strategy in TCP. RFC 813, July 1982.

[12] J. Davin and A. Heybey. A simulation study of fair queueing and policy enforcement. *ACM Computer Communication Review*, 20(5):23–29, October 1990.

[13] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking: Research and Experience*, 1:3–26, September 1990.

[14] P. A. Fishwick. *Simulation Model Design and Execution: Building Digital Worlds*. Prentice Hall, 1994.

[15] ATM Forum. The ATM forum traffic management specification version 4.0, April 1996.

[16] R. M. Fujimoto. Parallel simulation. *Communications of the ACM*, 33(10):31–53, October 1990.

[17] P. Gburzynski, T. Ono-Tesfaye, and S. Ramaswamy. *ATM-TN Network Model Design*. WurcNet Inc, Calgary, January 1995.

[18] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM'94*, pages 636–646, Toronto, CA, June 1994.

[19] F. Gomes, J. Cleary, S. Franks, B. Unger, and Z. Xiao. SimKit: A high performance logical process simulation class library in C++. In *Proceedings of the 1995 Winter Simulation Conference*, pages 706–713, Arlington, VA, December 1995.

[20] P. Goyal, H. M. Vin, and H. Chen. Start-time fair queuing: A scheduling algorithm for integrated services. In *Proceedings of ACM SIGCOMM'96*, pages 157–168, Palo Alto, CA, August 1996.

[21] R. Goyal, R. Jain, S. Kalyanaraman, and S. Fahmy. UBR+: Improving performance of TCP over ATM-UBR service. In *Proceedings of IEEE ICC'97*, volume 2, pages 1042–1048, Montreal, June 1997.

[22] R. Guerin, S. Kamat, V. Peris, and R. Rajan. Scalable QoS provision through buffer management. In *Proceedings of ACM SIGCOMM'98*, pages 29–40, Vancouver, British Columbia, September 1998.

[23] R. J. Gurski and C. L. Williamson. TCP over ATM: Simulation model and performance results. In *Proceedings of the 15th Annual IEEE International Phoenix Conference on Computers and Communications*, pages 328–335, Phoenix, AZ, March 1996.

[24] M. Hassan. Impact of cell loss on the efficiency of TCP/IP over ATM. In *Proceedings of the 3rd International Conference on Computer Communications and Networks*, pages 165–169, San Francisco, CA, September 1994.

[25] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM'88*, pages 314–329, Stanford, CA, August 1988.

[26] V. Jacobson. Modified TCP congestion avoidance algorithm. end2end-interest mailing list, April 1990.

[27] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, May 1992.

[28] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report TR-301, September 1984.

[29] D. R. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, July 1985.

[30] F. Kamoun and L. Kleinrock. Analysis of shared finite storage in a computer network node environment under general traffic conditions. *IEEE Transactions on Communications*, COM-28(7):992–1003, July 1980.

[31] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. *IEEE Journal on Selected Areas in Communications*, 9(8):1265–1279, October 1991.

[32] B. Kercheval. *TCP/IP over ATM: A No-Nonsense Internetworking Guide*. Prentice Hall, 1998.

[33] D. E. McDysan and D. L. Spohn. *ATM: Theory and Application*. McGraw-Hill, 1994.

[34] R. Van Melle. ATM-TN switch validation: Scheduling algorithms & buffer management, December 1998.

[35] S. Minzer. Broadband ISDN and asynchronous transfer mode ATM. *IEEE Communications Magazine*, 27(9):17–24, September 1989.

[36] K. Moldeklev and P. Gunningberg. How a large ATM MTU causes deadlocks in TCP data transfers. *IEEE/ACM Transactions on Networking*, 3(4):409–422, August 1995.

[37] J. Nagle. Congestion control in IP/TCP internetworks. RFC 896, January 1984.

[38] J. Nagle. On packet switches with infinite storage. *IEEE Transactions on Communications*, 35(4):435–438, April 1987.

[39] R. O. Onvural. *Asynchronous Transfer Mode Networks: Performance Issues.* Artech House, 1994.

[40] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control: The single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[41] A. Romanow and S. Floyd. Dynamics of TCP traffic over ATM networks. *IEEE Journal on Selected Areas in Communications*, 13(4):633–641, May 1995.

[42] W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols.* Addison-Wesley Publishing Company, 1994.

[43] D. Stiliadis and A. Verma. Design and analysis of frame-based fair queueing: A new traffic scheduling algorithm for packet-switched networks. In *Proceedings of ACM SIGMETRICS'96*, pages 104–115, Philadelphia, PA, May 1996.

[44] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, third edition, 1996.

[45] C. Tipper and J. Daigle. ATM cell delay and loss for best-effort TCP in the presence of isochronous traffic. *IEEE Journal on Selected Areas in Communications*, 13(8):1457–1464, October 1995.

[46] F. Tobagi. Fast packet switch architectures for integrated services digital networks. In *Proceedings of the IEEE*, volume 78, pages 133–167, January 1990.

[47] B. Unger, F. Gomes, Z. Xiao, P. Gburzynski, T. Ono-Tesfaye, S. Ramaswamy, C. Williamson, and A. Covington. A high fidelity ATM traffic and network simulator. In *Proceedings of the 1995 Winter Simulation Conference*, pages 996–1003, Arlington, VA, December 1995.

[48] Mark Williams. An introduction to ISDN, B-ISDN, and ATM. http://www.uq.edu.au/~miw/lecture/handout/intro_atm.htm.

[49] C. Williamson. *ATM-TN Traffic Model User's Manual*. WurcNet Inc, Calgary, September 1995.

[50] WurcNet Inc, Calgary. *ATM-TN System Release 1.0*, January 1995.