The Effects of Traffic Shaping on Link Utilization and Congestion in ATM-based WANs

Benjamin J. Ewy
Joseph B. Evans
Gary J. Minden

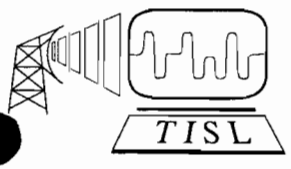
TISL Technical Report TISL-9770-29

Prepared for:

Defense Advanced Research Projects Agency/CSTO

Research on Gigabit Gateways
AARPA Order No. 8634
Issued by EDS/AVS under Contract #F19628-92-C-0080

January 1995



Telecommunications and Information Sciences Laboratory
The University of Kansas Center for Research, Inc.
2291 Irving Hill Drive Lawrence, Kansas 66045

Abstract

Networks based on the Transmission Control Protocol and the Internet Protocol are the heart of today's Internet and applications that use TCP/IP are the most prevalent form of "networked" applications available. Recent surveys indicate that TCP/IP is present in over 62% of all networks, well above the next most popular protocol.

The existing national network infrastructure will evolve to higher capacities by using ATM over SONET based backbones. Improving the understanding of TCP/IP performance in these networks is of fundamental importance. Testing done in gigabit/s testbeds showed that observed throughput was less than that expected given the link capacity, processing capabilities of the hosts involved, and individual host to host throughput tests.

In order to understand these problems a series of studies, including simulations, theoretical analysis, and experiments in a gigabit testbed were performed. Results indicate that the current TCP rate control mechanism alone is inadequate for congestion avoidance and control in wide-area gigabit networks. Buffer overflow in ATM switches due to congestion was found to be the major cause of poor performance. Further results show that TCP augmented by a source based ATM cell level pacing addresses these problems and allows the link to be more fully utilized.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Networks based on the Transmission Control Protocol and the Internet Protocol are the heart of today's Internet and applications that use TCP/IP are the most prevalent form of "networked" applications available. Fang et al [11] report on a survey that indicated that TCP/IP was present in over 62% of all networks, well above the next most popular protocol.

The behavior of the Transmission Control Protocol (TCP) [21] has been studied by a variety of authors [9, 11, 13, 25, 27, 32], however improving the understanding of TCP/IP performance in networks based on SONET and ATM is of fundamental importance to the evolution of the existing national infrastructure to higher capacities. The previous work served as a foundation for this study which used a number of analytical models, simulation studies, and experiments in a gigabit testbed to explore, predict, and improve the performance of TCP/IP applications in ATM over SONET networks.

Results indicate that the current TCP rate control mechanism alone is inadequate for congestion avoidance and control in wide-area gigabit networks. Further results show that TCP augmented by a source based ATM cell level pacing addresses these problems and allows the link to be more fully utilized.

1.1 Motivation and Organization

During the initial testing of the MAGIC terrain visualization application [23], observed throughput was less than that expected given the link capacity, processing capabilities of the hosts involved, and individual host to host throughput tests. This study is an attempt to explain these anomalies and find solutions to the throughput limitations.

This work is organized as follows. First, background on TCP/IP and its implementation, as well as issues relating to its performance in networks is presented. Background on ATM, SONET, and networks based on these standards are covered next. Performance studies of ATM LANs and WANs, both simulation based and experimental, are covered in Chapter 2. Chapter 3 focuses on analytical models of congestion problems in ATM networks. In Chapter 4, a method for reducing congestion based on ATM cell level pacing is presented, and experimental results are presented.

1.2 Background on TCP/IP

The Transmission Control Protocol (TCP) [21], is a connection-oriented layer 4 protocol which resides above the Internet Protocol (IP) [20]. Its purpose is to multiplex application data streams together, breaking the streams down into packets, and then guaranteeing the in-order and error-free reception of these packets, where the data is then demultiplexed and passed to the appropriate applications. Thanks to the success of TCP in performing these tasks, its widespread use insures that these activities must be done in environments that vary widely in transmission rates, delays, and error rates.

IP is an unreliable connectionless protocol which defines the Internet addressing scheme, routes datagrams to remote hosts, and performs fragmentation and reassembly of datagrams as needed.

TCP uses a sliding window scheme to provide flow control. At connection time, the receiver informs the transmitter of the space available to store incoming data. This amount is then used as the number of bytes that can be unaccounted for in the network at any one time. The receiver acknowledges the successful receipt of packets after they are checked for completeness and correctness. Acknowledgements (ACK's) are cumulative, and each one represents the last byte received. The original maximum window size for TCP was 64KB, but long delay paths, and increasing bandwidths inspired research in this area resulting in new options to TCP[14].

TCP uses timers to tell the transmitter when ACK's are overdue. TCP resends the entire packet that is outstanding when it expires. The timer is based on an estimated Round Trip Time (RTT) which TCP maintains internally. Packets can be unacknowledged for various reasons. They may be received but contain errors which will be referred to in this paper as incorrect packets, or portions of the packet may never arrive at the destination, which will be referred to as an incomplete packet. In both of these cases the packet is dropped, by the destination if incorrect, and by the network if incomplete.

TCP divides the buffers of data into segments before transmitting. The Maximum Segment Size (MSS) is another parameter set up at connection time. This value is based on the capabilities of the underlying transmission layers. For TCP/IP over Ethernet this value is 1500 bytes and for ATM based networks it is recommended that a MTU of 9180 bytes [3] be used.

TCP uses an algorithm known as slow start developed by Jacobson [13] to address serious congestions problems that were occurring in TCP/IP networks. When congestion occurs in a TCP network, delays grow dramatically. As delays increase, the expected Round Trip Time is exceeded, causing a timeout, and a corresponding retransmission. This resending of a window's worth of data into an already congested network worsens the problem. The problem can cycle until

everything slows to a standstill, an experience known as congestion collapse.

Jacobson's algorithm changes the window size to be the minimum of the advertised window and a *congestion window.* The congestion window starts out the same size as the advertised window and never exceeds that size. Every time a segment times out, the congestion window is cut in half and the retransmission timer is backed off exponentially. When the congestions is over, or for a new connection, the congestion window is set to the size of a single segment. When the segment is ACK'ed, the window is increased by one segment. This linear increase continues, adding an additional segment to the congestion window's size for every successfully transmitted segment, until an entire advertised windows worth of data is transmitted.

When the congestion window and slow start are used together, TCP quickly cuts the amount it is sending when congestion occurs and then slowly increases the amount of data sent back up to a full windows worth if no further time outs occur.

1.3 TCP/IP in Long Delay, Large Bandwidth Environments

TCP throughput is limited by both the transfer rate, and the round trip delay. The product of these two terms is how much data is needed to fully utilize a given connection. To achieve maximum utilization, the window size must be as large as this bandwidth-delay product. The original maximum window size of 64 KB became a limiting factor in coast to coast networks, and high bandwidth links.

To allow for larger window sizes Jacobson [14] proposed adding a TCP option that allows for a window scale. This scaling permits window sizes of up to 1 GB.

A second major problem that occurs in long delay - high bandwidth networks is the effect of losses on throughput. Expanding the window size to match the

requirements of these links results in a corresponding increase in the probability of a packet being either incomplete or incorrect in a given window. When a loss occurs and the congestion window collapses, the slow start mechanism prevents the full utilization of the link. The slow start is unnecessary in cases where the loss is due to infrequent link errors, and the penalty imposed on the throughput is so great that losses due to congestion must be carefully avoided.

Jacobsen [14] proposed the Fast Retransmit and Fast Recovery algorithm for preventing slow start with a single loss in a window, and further work is being done on selective acknowledgments which would allow recovery from multiple losses per window. Neither of these methods have been fully adopted by vendors of TCP/IP software, but their importance assures their integration into future systems.

Fast Retransmit and Fast Recovery work as follows. TCP is required to send a duplicate ACK containing the sequence number that was expected when it receives a segment out of order. When the source host receives the duplicate ack, it watches for three of them to decide if there was an out of order delivery or if the packet really was lost. When this happens the source immediately retransmits the missing segment, and cuts the window size in half. Slow start is not used. When the ACK for the missing data finally arrives, the window is increased back to the normal size. This results in much better performance that the standard tcp_slowtimo() routine that only retransmits packets every 500 ms.

1.4 Implementation of TCP/IP for Unix Systems

The performance of network I/O depends on the software implementation of the protocols. The following discussion is for the DEC OSF/1 implementation of TCP/IP [7], which is based on the 4.3 BSD-Reno distribution of Berkely Unix [17].

A socket layer is provided as an interface between applications and the protocol implementations for the network. A user process uses the send() call to interface with the socket. The kernel side of the socket interface uses the bcopy() routine to copy the user data into kernel space. The kernel buffers are known as mbufs. The copying of large amount of data can be a limiting factor on throughput, and so after the data is in kernel space, a pointer to the mbuf is passed down the protocol stack instead of actually moving the data. The bcopy() routine has also been optimized to work as fast as possible on the Alpha AXP architecture. The code was written in assembly language and deals with 64 bits of data at a time [7].

Once in kernel space the the tcp_output() routine partitions the buffer into segments based on the Maximum Segment Size negotiated at connection time. tcp_ouput() then fills the TCP header for each of these segments, performs the checksums, fills the IP header and then calls ip_output().

ip_output() inserts any options such as the window scale option for large window sizes, fills the remaining parts of the ip header, determines the route, and computes the IP header checksum.

Once the kernel is done with the data, it has to be copied across the system bus to the adapters buffers using DMA.

When the receiving host's interface receives the packet, it generates an interrupt which calls the device driver. The driver filters the data, sends it to the appropriate protocol queue and calls ipintr(). This routine removes a packet from the input queue, and checks to see that the packet is complete and that the header is error free. It then passes the data up to the tcp_input() routine.

The tcp_input() routine checks the TCP header and verifies the correctness of all of the data. It checks to see if the packet is within the window that is currently expected, and sends an acknowledgment accordingly. If everything is correct, the data is placed in the socket receive buffer.

The receive user client will use the listen() and read() functions to get the data out of the socket buffer where it can then be processed normally.

In addition to the routines used to send and receive data, there are a number of kernel variables that can be manipulated with programs such as netconfig by Jon Kay. The _tcp_recvspace variable is the maximum window size the host is willing to allow. The _tcp_mssdflt variable is the maximum segment size to be used on non-directly connected networks.

1.5 Asynchronous Transfer Mode (ATM)

Due to the non scalability of both Time Division Multiplexing techniques and the media access techniques, an asynchronous, statistically multiplexed, packet based scheme called asynchronous transfer mode (ATM) was developed for the implementation of the services offered by B-ISDN [22, 19].

ATM is a packet-oriented transfer method. It allows multiple logical connections to be multiplexed over a single physical channel. The information flow on each logical connection is organized into fixed-size packets called cells [22, 30]. As shown in Figure 1-1, ATM cells are 53 byte data units that consist of a 5 byte header and a 48 byte body.

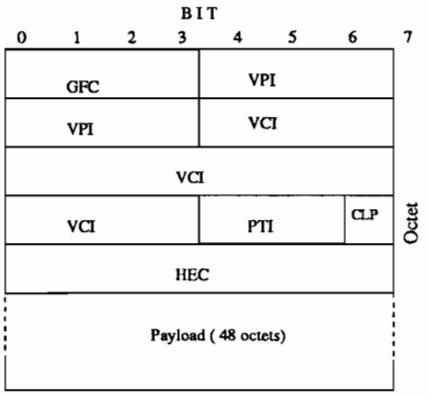


Figure 1-1: ATM Cell Format

The header contains a Virtual Circuit Identifier (VCI) and a Virtual Path

Identifier (VPI) which are used for routing by switches in the network. The PTI field contains the Payload Type Indicator used by ATM Adaptation Layers to indicate to the network the kind of traffic contained in the cell. The CLP field indicates the Cell Loss Priority used by the network to decide which cells can be dropped when congestion occurs. A cell with a lower priority is the first candidate for dropping. The HEC field contains the Header Error Code which is used by the network equipment to determine if any bits in the header have been corrupted during transmission. Corruption of header bits may lead to delivery of cells to a incorrect end user. The header also has a 4 bit GFC field that can be used for Generic Flow Control. The standards for use of these bits are presently being developed [12].

ATM was developed with the intention of serving a wide variety of traffic types. Two main classes of traffic are Constant Bit Rate (CBR) traffic, and Available Bit Rate (ABR) traffic. Video feeds or telephone conversations are examples of CBR traffic, while the traditional data transfer between computers is an example of ABR traffic. ATM Adaptation Layers exist to adapt higher layers of the protocol stack, such as IP, to ATM cells. There are several AALs defined, each of which is specialized for a particular type of traffic. An example may be a video source which has a constant rate of traffic, and a specific timing relationship that must be preserved. This type of traffic is served by AAL1. Throughout this paper, we will focus on traffic that is best served by AAL5 which is intended for packet type traffic consisting primarily of data. AAL5 [22] is responsible for segmenting and reassembling the packets passed to it. AAL5 also performs a CRC checksum of each packet, and appends the CRC at the end of the packet while marking the last cell in a packet with a one in the Payload Type Indicator field.

1.6 Traffic Shaping to Insure Quality of Service

Network resources can be overtaxed by customers causing congestion in networks if no constraints exist. These constraints require resource management schemes that allocate network resources such as buffers, ports, etc. One such resource that needs to be managed is the bandwidth allocated to a particular customer. Bandwidth management has many different forms and can be either statically determined or vary with time.

Bandwidth management may be applied on either the customer side or the network side of the ATM User-Network Interface (UNI) [12]. For example, customers are encouraged to produce traffic that does not violate the traffic contract with the network provider. The network side may police the customer traffic and penalize nonconforming traffic. To produce conforming traffic the customer may shape the traffic between the source and the network. Shaping or smoothing refers to queueing ATM cells and then releasing those cells so that the burstiness and rate of the source is controlled.

Static bandwidth management allocates a specific amount of bandwidth for a given connection. Dynamic bandwidth management changes the bandwidth allocated to a particular customer over time. This is accomplished by observing parameters such as the average rate or the average burst size of customer traffic and allocating available network resources to accommodate the traffic. The Quality of Service provided by the dynamic adjustment of the network resources among several streams is superior to that obtained by static assignment schemes [5].

1.7 Synchronous Optical NETwork (SONET)

The most widely accepted standard for fiber optic transmission in North America is SONET. This standard has been agreed to internationally by the ITU, and has

the benefit that transmission equipment from one manufacturer can communicate with receiving equipment from another manufacturer [4, 26]. Outside of North America, a largely compatible system called the Synchronous Digital Hierarchy (SDH) is used. A SONET frame contains two parts, the SONET Overhead and the SONET Payload.

Figure 1-2 shows a STS-1 SONET frame as a set of bytes arranged in rows. Transmission of this frame is done row by row, from left to right. The overhead is distributed throughout the frame rather than coming at the start. The SONET overhead is further broken into three parts:

- *Section*: Individual fiber transmission links between repeaters and links between repeaters and other equipment , such as multiplexers.
- *Line*: Fiber transmission lines connecting SONET equipment at a given rate in the SONET hierarchy, STS-n. A line may consist of several sections
- *Path*: The path over which traffic is sent end-to-end between originating and terminating equipment using the SONET format. A path may traverse several lines.

The entire STS-1 frame consists of 810 bytes and repeats at 125 μs intervals thus yielding a nominal bit rate of 51.84 Mb/s. The Synchronous Payload Envelope (SPE) is the area reserved for transporting the data also called the payload, and contains 783 bytes per frame. Nine bytes in the SPE are overhead information that the system uses to verify the integrity of the payload and to supply end-to-end signaling. These nine bytes of overhead are referred to as Path overhead and they stay with the payload until it reaches the final destination. The remainder of the STS-1 frame consists of twenty-seven bytes, used for inter-network signaling and error checking. Nine bytes of the twenty-seven are used for the section portion of the network, that is, between optical repeaters. The remaining eighteen bytes are used for the line portion of the network, that is, between terminals. The

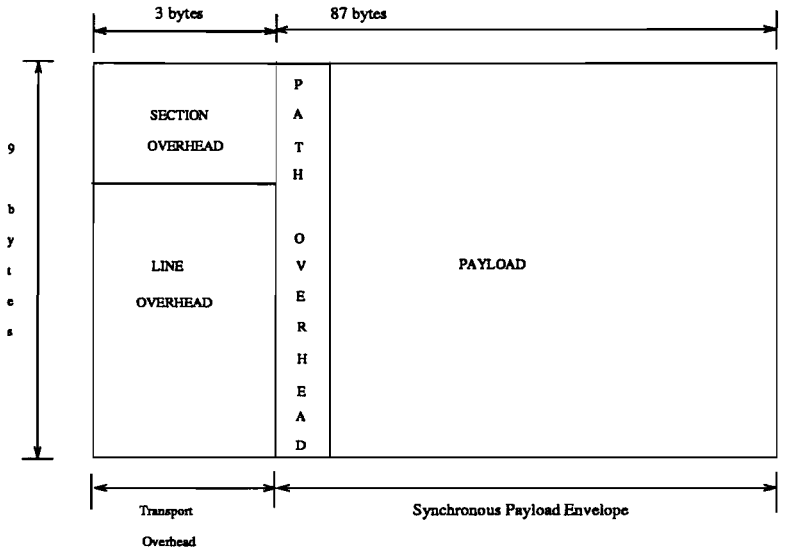


Figure 1-2: SONET STS-1 Frame Format

SONET format uses a pointer to do frequency and phase aligning of the payload. The SONET pointer contains the location of the beginning of the data payload which allows forward or backward movement of data within the frame [29].

STS-1 frames can be multiplexed into higher rate signals denoted by the terminology STS-N, where N denotes the number of STS-1's that have been byte multiplexed together. The optical signal equivalent to STS-N is given the designation Optical Carrier (OC)-N. Higher rate signals can be achieved by concatenation of individual STS-1 frames in a single STS-N frame. A SONET STS-3 frame is made of 3 concatenated STS-1 frames and is carried as a single stream. Figure 1-3 shows a SONET STS-3 frame [26]. It is a 2430 byte pattern that repeats every 125μs yielding a nominal bit rate of 155.52 Mb/s. Its two main components are the 81 byte Transport Overhead (TOH) and the 2349-byte Synchronous Payload Envelope (SPE). STS multiplexing is a process in which the transport overhead of the multiplexed signals are aligned the payload pointer, and in which the signals are sequentially byte-interleaved. This results in the appearance of the byte triplets in a STS-3 frame.

An STS-12 stream is at twelve time the basic SONET rate, that is 622 Mb/s.

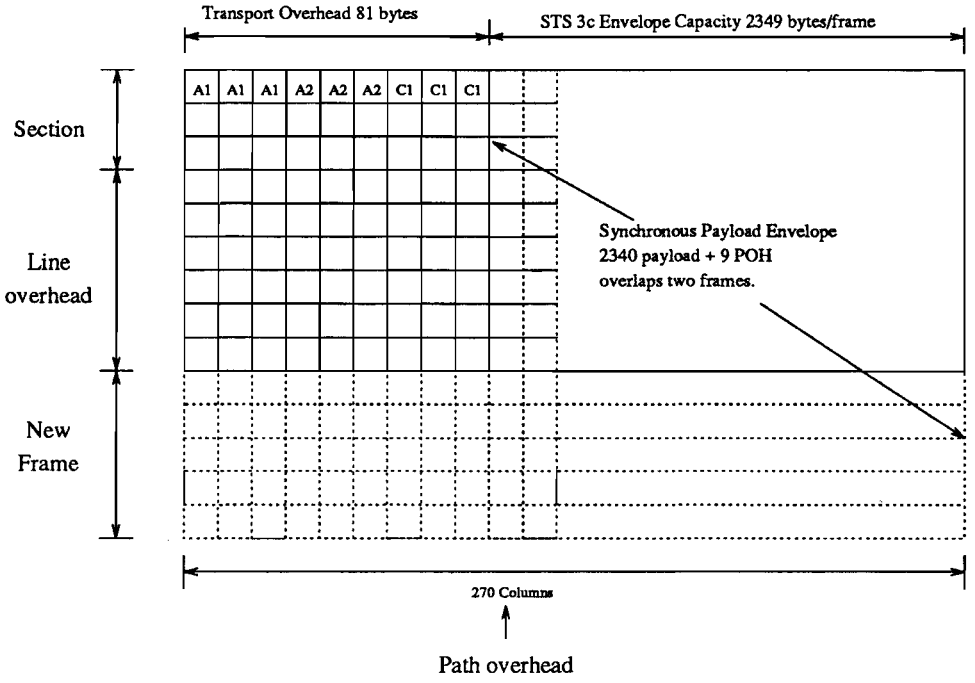


Figure 1-3: STS-3c Frame Structure

12 STS-1 or 4 STS-3 streams can be multiplexed together to form a single STS-12 stream. The STS-12 frame is a scaled up version a STS-3 frame [4, 29].

SONET frames can be used to carry ATM cells. The transmitting end assembles ATM cells into the payload portion of a frame. The receiving end of the SONET frame extracts cell from the payload and pass them up the protocol stack. Figure 1-4 shows the transport of ATM cells in a SONET frame.

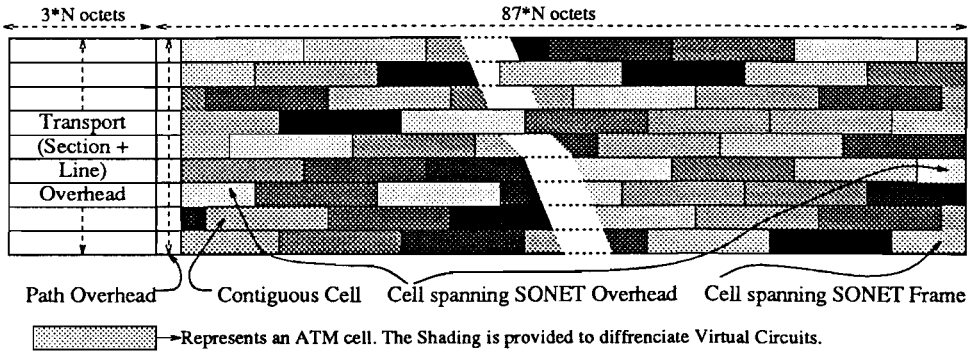


Figure 1-4: ATM Cells in a SONET Frame

1.8 ATM Local and Wide Area Networks

ATM networks consist of point to point links between hosts and switches, and switch to switch links. The hosts each have an ATM interface, and workgroup switches typically consist of between 10 and 64 host ports, as well as additional ports for connecting to other switches. Host links are typically between 51 and 155 Mb/s, while switch to switch links can be as large as 622 Mb/s.

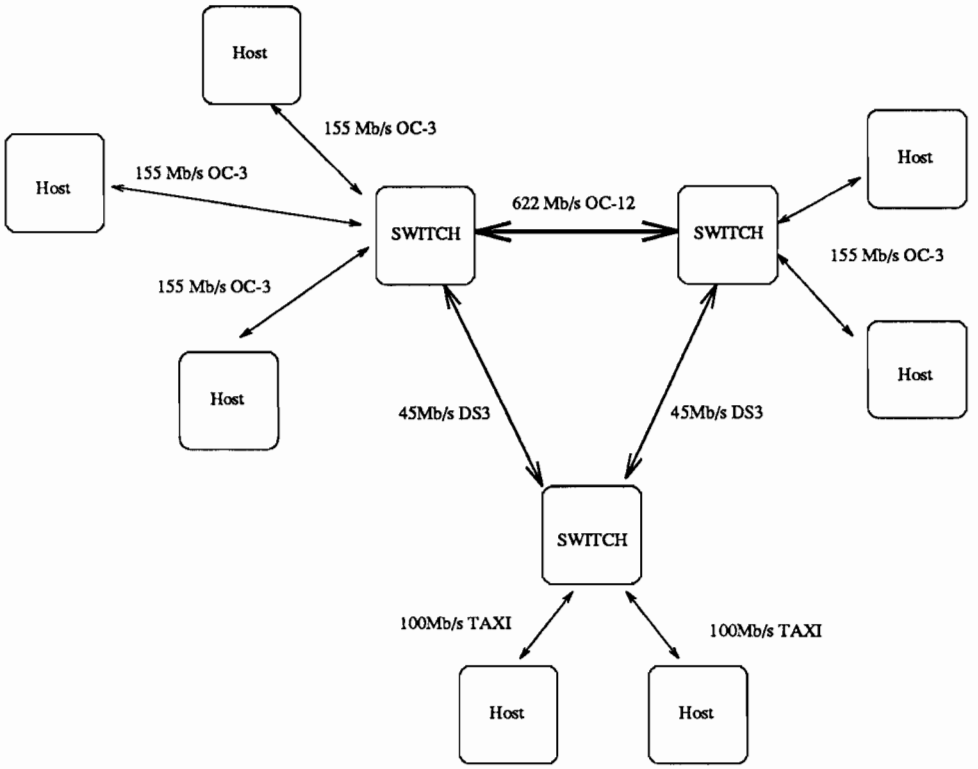


Figure 1-5: ATM Network

ATM switches that have ATM over SONET interfaces have an additional advantage. The switch to switch links can be carried over the standard SONET infrastructure that phone companies are installing. This allows voice, video, and data traffic to travel over the same networks.

Switches are available from many vendors and either have a common buffer memory for all of the ports, or a fixed space per port. This buffer space is a

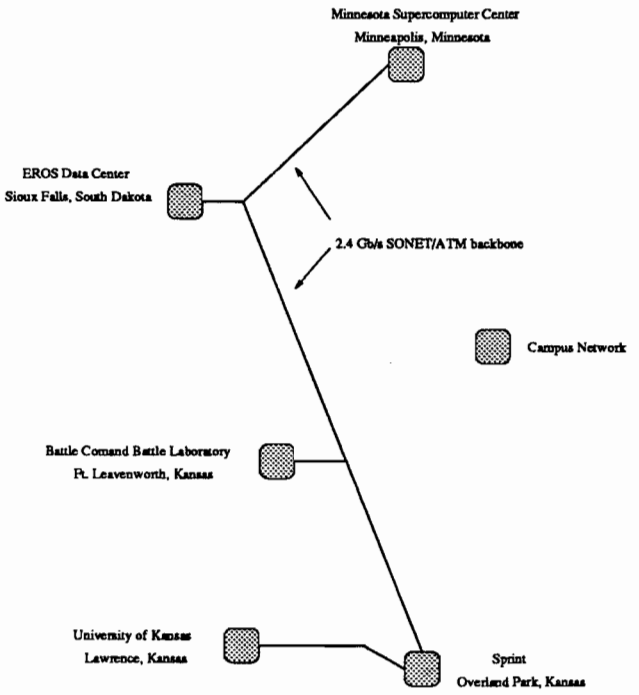


Figure 1-6: MAGIC Network

resource that must be carefully allocated. When an ATM cell arrives and there is no space for it in the buffers of a switch it is dropped from the network.

Switches can implement different policies for choosing what cells to drop. The Payload Type Indicator and Cell Loss Priority bits can be used to selectively drop traffic. Voice and video traffic can afford to have cells missing in the streams of traffic, but data traffic must arrive complete.

1.9 The MAGIC Network

The Multidimensional Applications and Gigabit Internetwork Consortium (MAGIC) is a group of industrial, academic, and government organizations participating in gigabit network research. The MAGIC backbone network operates at 2.4 Gb/s and each site on the network includes LANs or hosts communicating at gigabit per second rates. The MAGIC network is depicted in Figure 1-6.

The University of Kansas (KU) has deployed an experimental gigabit LAN called the AN2, provided by Digital Equipment Corporation and developed by Digital's Systems Research Center [2]. The hosts at KU communicate locally via the AN2 switches, and with remote sites via the MAGIC backbone. Interoperability has been demonstrated between the Digital AN2 switch, Digital OTTO host interfaces, the Fore Systems ASX-100 switch, a variety of Fore Systems host interfaces, and Northern Telecom network termination equipment.

The KU LAN contains Digital DEC 3000 AXP and DECStation-5000 hosts equipped with OTTO AN2 host adapter boards (which conform to standard OC-3c SONET/ATM specifications) attached to the AN2 switch. The ATM interfaces are installed on the Turbochannel bus of the Digital workstations. This interface includes several features which make it interesting for network studies. The interface supports credit-based flow control, which can be used with similarly equipped interfaces or an AN2 switch. This flow control guarantees no cell loss within a local area AN2 network. The OTTO interfaces implement AAL5 [28], which was used exclusively in these tests, in hardware. The flow control can be selected and controlled on a per VC basis.

The switches used at other sites on the MAGIC network included Fore Systems ASX-100 ATM switches equipped with a mixture of 155 Mb/s SONET OC-3c ports and 100 Mb/s TAXI ports. The ASX-100 versions used for our experiments were equipped with 256 cell output buffers per port.

The other host interfaces were Fore Systems 100 Mb/s TAXI interfaces on Sun SPARCstation-10 and SGI Onyx workstations. These interfaces implement most AAL5 functionality in hardware.

Chapter 2

TCP/IP in ATM Networks

From the discussion on TCP/IP, its implementation, and its use, it can be seen that throughput is limited by a number of factors.

- Bandwidth of the link
- TCP Window Size
- Delays of the link
- Bandwidth of the host bus
- Processing demands of protocol overhead

Cavanaugh [6] quantifies the amount of protocol overhead involved with using TCP/IP in an ATM over SONET network and shows that 5.76 Mb/s are used by SONET overhead in an OC-3 network. With an IP MTU of 9180 bytes, he shows that the AAL5, IP, and TCP overheads consume another 15.247 Mb/s leaving a maximum total of of 134.513 Mb/s available to the application.

This chapter will focus on various studies, both simulation based and experimental, of TCP/IP in ATM networks and how they add additional constraints to the throughput of a given TCP/IP application.

2.1 Performance Studies for LANs

2.1.1 Previous Work

There have been many studies of ATM performance in LANs recently [11, 16, 25]. These studies have shown that the TCP congestion mechanism is not sufficient for adequate performance in ATM LANs. During periods of congestion ATM switches drop cells. The incomplete packets that arrive at the destination force timeouts and retransmission. Not only is the bandwidth wasted because the whole segment needs to be retransmitted, but the original incomplete packet wastes network resources while being delivered.

Early work by Romanow et al [25] showed that TCP/IP over ATM links can utilize as little as 40% of the link bandwidth. Keung et al [16] showed that for systems that have not implemented the Fast Retransmit Fast Recovery Algorithm the throughput can be degraded by as much as 64% with just a 1% cell loss probability. This showed an even more pessimistic view of performance than the early work.

Simulations [25] show that smaller switch buffers, larger TCP windows, and larger TCP segment sizes can each reduce the throughput of TCP over ATM by increasing the possibility that one or more cells will be discarded per segment. It will be seen in the next section that increasing the TCP segment size is necessary to minimize overhead due to protocol processing. Furthermore, window sizes need to be at least as large as the delay bandwidth product to fully utilize expensive network links.

Fang et al [11] showed that increasing the amount of buffers for each VC so that an entire window full of cells can be held eliminates congestion, but for a 128K window of data you need

$$\frac{131072 \text{ bytes}}{48 \frac{\text{bytes}}{\text{cells}}} = 2,731 \text{ cells.} \quad (2.1)$$

2,731 cells of storage for every active VC in a switch is an expensive solution.

Two other factors that influence throughput in ATM networks are ACK compression, and inaccurate Round Trip Time estimates. ACK compression can occur if the return link of a TCP connection meets congestion. The ACKs on this path will bunch up behind the congested area and be received all at once, causing a large burst of data to be sent when they are received. Bursty traffic can also cause inaccurate RTT estimations for TCP. Premature retransmissions due to erroneous timer values can further degrade the performance.

2.1.2 LAN Experiments

To increase the understanding of various factors in the performance of TCP/IP in an ATM based LAN, a number of experiments were performed. Two DEC Alpha workstations with OTTO OC-3 interfaces were tied directly together to investigate performance without constraints from any switches.

The primary software tool used for these experiments was ttcp, a public domain tool originally created at the US Army Ballistics Research Lab and modified by several other authors. This application writes TCP packets from local memory to memory on a remote host as quickly as the intervening operating systems, interfaces, and networks allow.

Delays and throughput were examined for various MTU sizes and window sizes. The best performance seen was using a MTU of 16640 bytes and a window size of 128KB. In this configuration the throughput was 128.5 Mb/s, very close to the theoretical maximum for OC-3. Using the MTU of 9180 as recommended [3], the maximum throughput was found to be 111.5 Mb/s. Figures 2-1 and 2-2 show the relationship between MTU size and throughput in the ATM LAN environment.

At the smaller MTU sizes the overhead processing of all of the necessary segments is the limiting factor, not the window size. As the MTU size increases, the larger windows improve performance. It is clear that in the LAN environment, even with a relatively fast processor like the Alpha, the protocol processing for segments of 9180 bytes is limiting the overall performance.

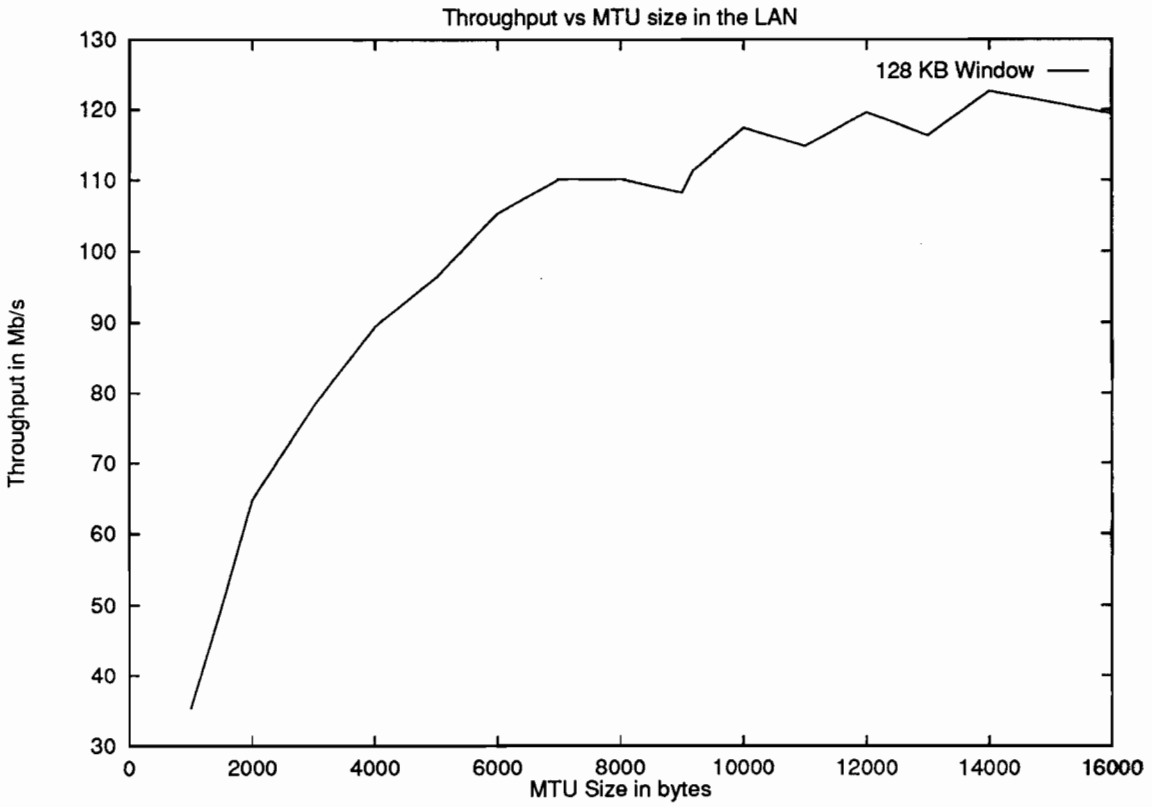


Figure 2-1: Throughput for various MTU's, with a 128KB window

Figure 2-3 plots the average delay in milliseconds versus the size of a segment sent. This shows the nearly linear increase in delay that can be expected copying the longer segments.

These studies show that in addition to the limits discussed before, window size and MTU size affect the throughput. The speed of the processor affects how much the MTU size changes the overall delays.

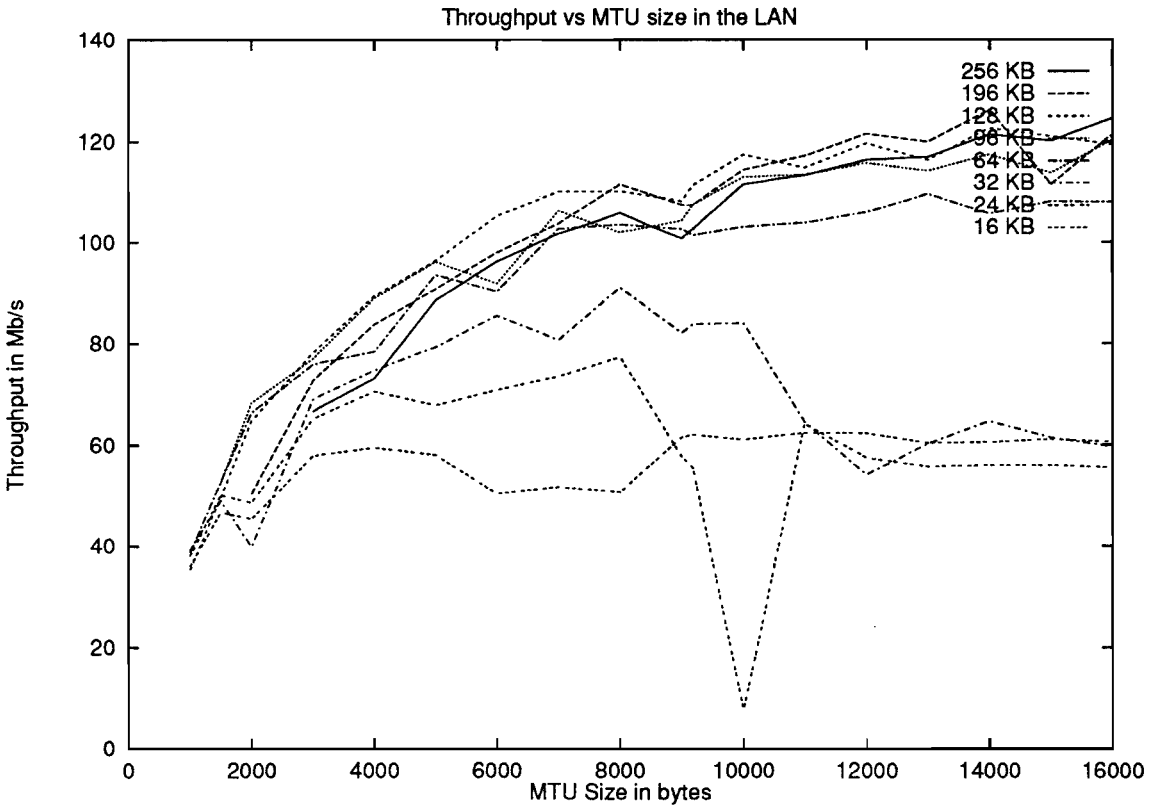


Figure 2-2: Throughput for various MTU's and windows

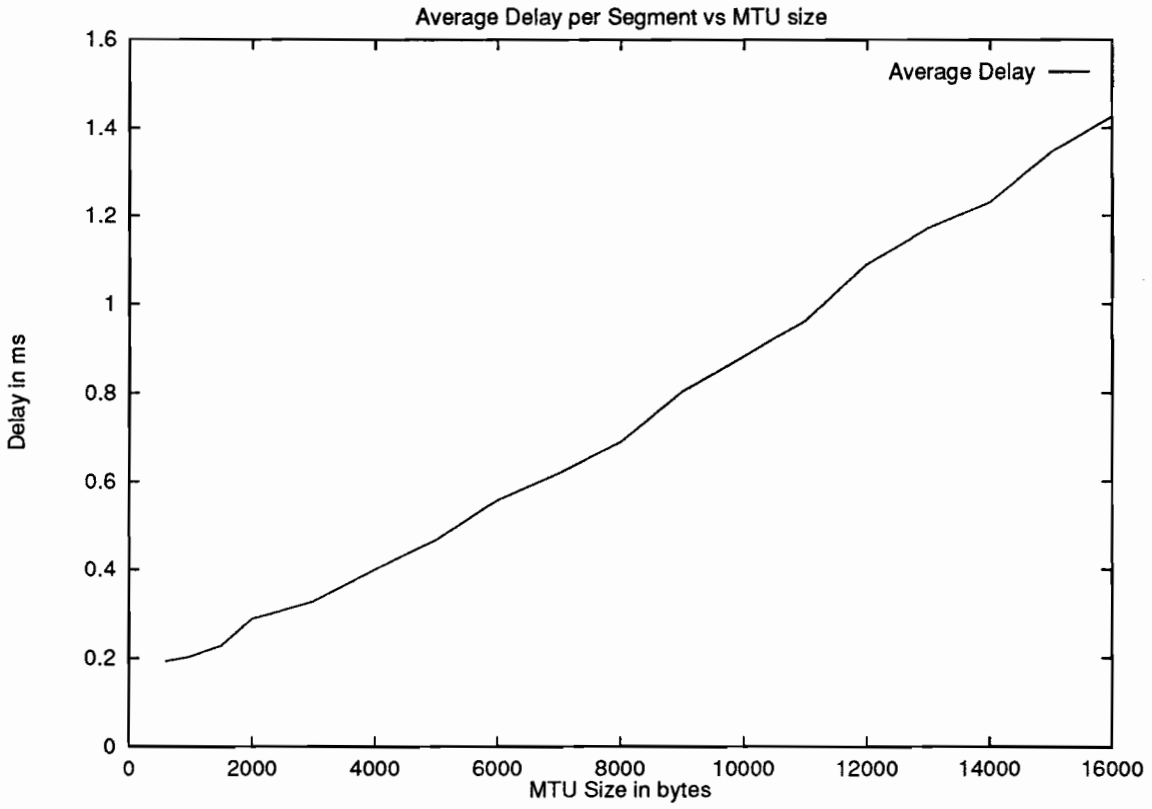


Figure 2-3: Average delay for various segment sizes

2.1.3 Simulations of LANs

Simulations of local area networks using ATM were performed to test models of ATM components. The simulator used was the MIT Network Simulator (NetSim) [18]. Additional components were added to NetSim by researchers at Sandia National Laboratories [11] to allow ATM based networks to be modelled. The simulator contained blocks for host adapters, point to point links, and ATM switches. There is also a model of the BSD 4.3 TCP implementation including the fast retransmit and recovery algorithm as used by our experimental hosts. Simulation parameters were based on previous work by Sandia [11] and compared with our experimental cases to validate the model.

A 250 us overhead delay was used for TCP processing, 200 ns delays were assigned for AAL5 segmentation and reassembly, and a switching delay of 4 us was used in the switch. 155 Mb/s OC-3 links were used in the simulation with 500 ns of delay in each link for our Local Area Network model. The TCP segment size was set at 9180 bytes and a variety of window sizes were used. Figure 2-4 shows the throughput achieved for a variety of different window sizes using MTUs of 16000 and 9180 bytes. Figure 2-5 compares the simulation results to the experimental results from our LAN shown previously. With the close match of the experimental measurements and the simulations we felt confidence in our model of the ATM network.

2.1.4 Improving performance for LANs

In the absence of strict flow control, several alternative have arisen for improving throughput. Minimizing cell loss due to congestion is one obvious goal. Many switch designs currently employ a shared buffer space for queueing cells. Making the buffer as large as possible is an obvious but expensive choice. Fang et al [11] demonstrated selective packet dropping schemes in ATM switches. These schemes did not forward cells from a packet that had a cell dropped due to congestion.

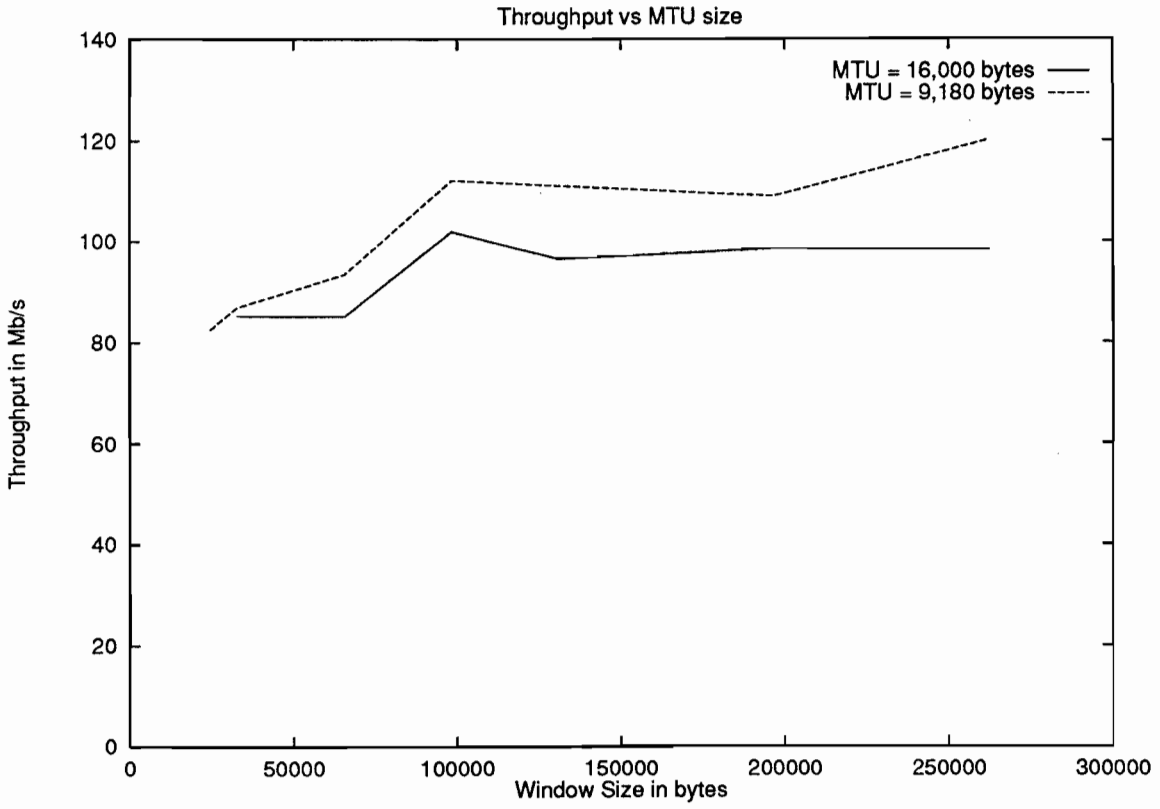


Figure 2-4: Throughput for various MTU's and window sizes in LAN Simulation

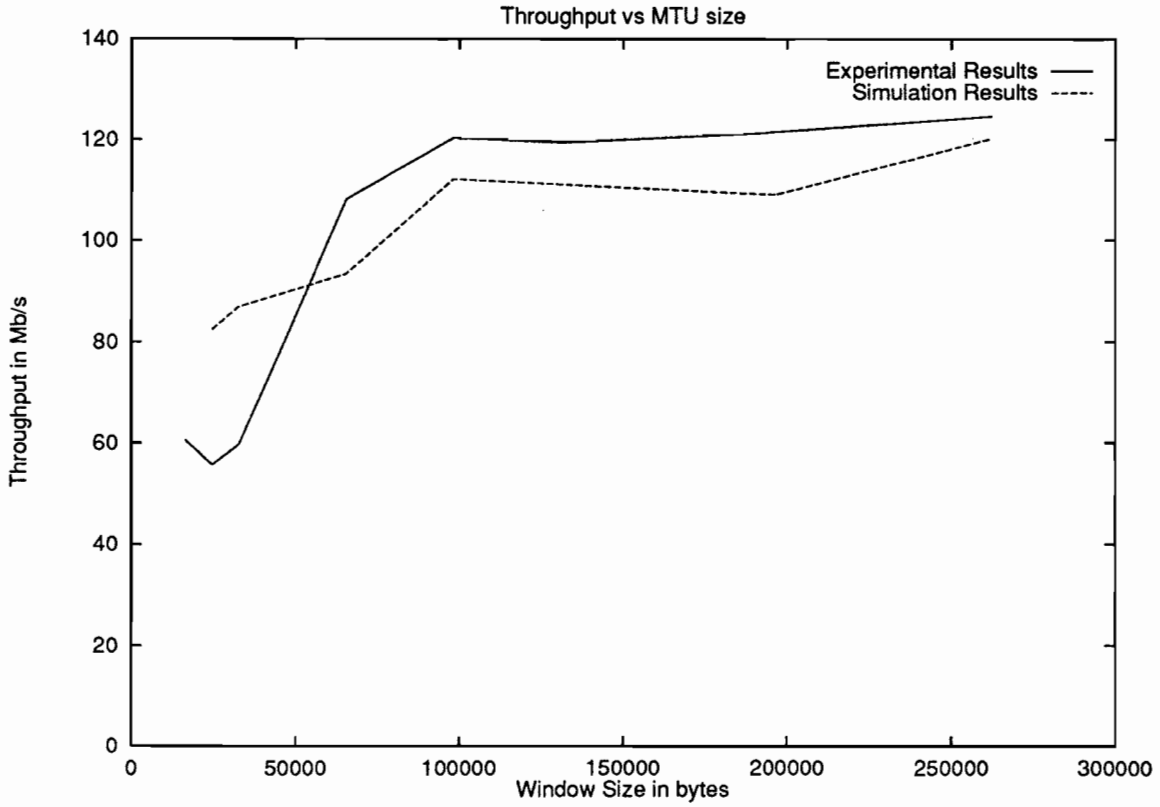


Figure 2-5: Comparison of LAN experimental and simulation results, MTU=16000 bytes

This technique helps remove congestion in downstream switches by removing data that is going to have to be retransmitted anyway. Chapman et al. [8] show how various flow control algorithms reduce the needed amount of buffers by allowing the switch to more effectively act like a statistical multiplexer. Traffic shaping can be used to meet the requirements of reducing burst sizes. TCP sends a segment as a burst of cells when an acknowledgment is received. Shaping these bursts so that they are more evenly distributed reduces the likelihood of congestion.

2.2 Performance of ATM WANs

As was discussed before, large segments are needed to reduce overhead processing, and large windows sizes are needed to fully utilize links with large bandwidth delay products. Villamizar et al [31] show that current DS3 based WANs such as NSFNET effectively use TCP in a high bandwidth-delay product environment, but points out the problems that have not been addressed for ATM WANs, primarily congestion avoidance for ABR traffic.

Evans [10] showed the relationship between queue size, delay in the WAN, bandwidth mismatches, and the maximum obtainable throughput. This type of situation is common when interfacing ATM LANs to typical wide area networks such as 45 Mb/s T3 connections or 100 Mb/s TAXI interfaces. Figure 2-6 shows

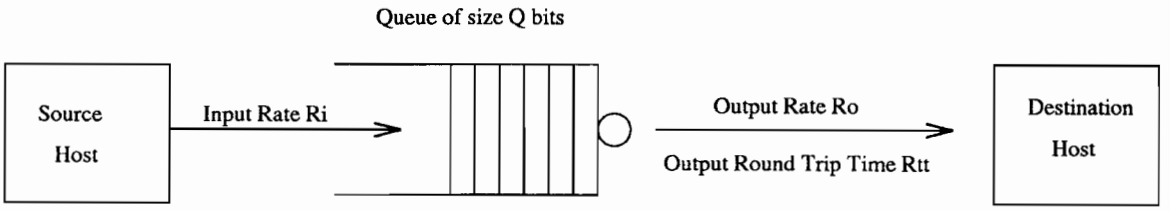


Figure 2-6: Rate Mismatch Model

the model. The objective is to find the queue size necessary in the switch to prevent an overflow of the queue due to the rate mismatch, while still allowing a full windows worth of cells to be transmitted. The number of segments it takes

to overflow a queue can be found as follows. If the input capacity is Ci and the output rate is Co, a queue of size Q bits will fill in the following period

$$\text{time to fill a queue} = \frac{Q}{(Ci - Co)}. \tag{2.2}$$

The number of bits that leave the queue during the time it takes the queue to fill is the output rate times the queue fill time.

$$\text{Bits leaving queue} = Co * \frac{Q}{Ci - Co}. \tag{2.3}$$

The total number of bits that it takes to overflow the queue is then the size of the queue plus the number of bits leaving the queue during the queue fill time.

$$\text{Bits to overflow queue} = Q + \frac{Q * Co}{Ci - Co}. \tag{2.4}$$

We would like to have this number larger than the window size needed to fully utilize the link. If the optimal window size is set to the bandwidth of the output link (Co) times the latency of the link (Rtt), we can combine the two equations to get the size of the queue needed for a given set of link rates and latencies, that is,

$$\text{Bits to overflow queue} = Q(1 + \frac{Co}{Ci - Co}) \tag{2.5}$$

$$\text{Optimal window size} = Co * Rtt \tag{2.6}$$

$$Q(1 + \frac{Co}{Ci - Co}) = Co * Rtt \tag{2.7}$$

$$Q = \frac{Co * Rtt}{\frac{Ci-Co}{Ci-Co} + \frac{Co}{Ci-Co}} \tag{2.8}$$

$$Q = \frac{Co * Rtt * (Ci - Co)}{Ci} \tag{2.9}$$

Queue sizes for several combinations of link rates and round trip times are

shown in Figures 2-7, 2-8, and 2-9. The results take into account the overhead bandwidth taken up by higher level protocols [6]. 18 ms is the round trip time for the longest connection on the MAGIC network which reaches from KU to the Minnesota Supercomputer Center. 50 ms is the round trip delay a from KU to Washington D.C. and 100 ms is the cross country round trip delay.

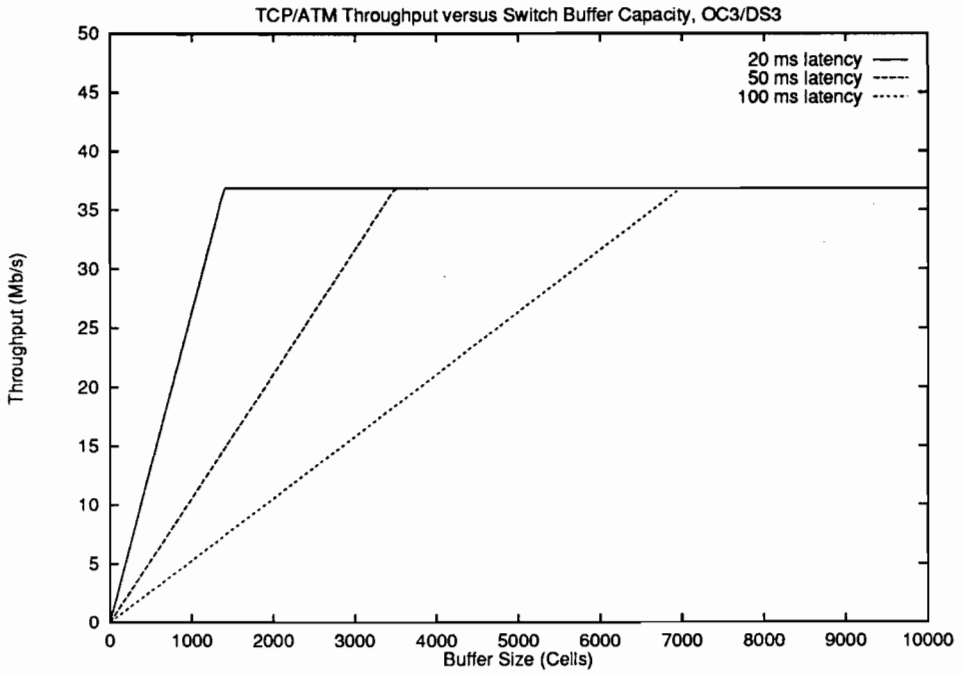


Figure 2-7: WAN Throughput with OC-3 to DS3 Mismatch

Equation 2.9 can be rewritten for the maximum output rate with a given queue size and input and output capacities as

$$Ro = \frac{Q}{Rtt} * (1 + \frac{Co}{Ci - Co}). \tag{2.10}$$

2.2.1 Experiments with ATM WANs

Several experiments were performed using the MAGIC network to serve as a baseline for TCP performance over the long delay paths in MAGIC. The experiments used a variety of host interfaces, link speeds, and round trip times.

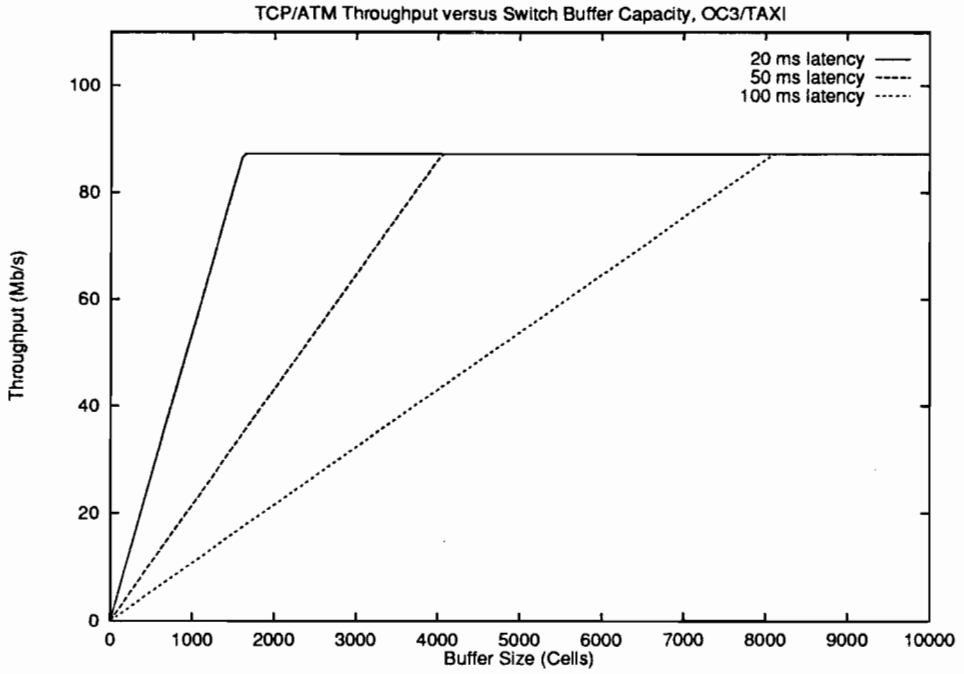


Figure 2-8: WAN Throughput with OC-3 to TAXI Mismatch

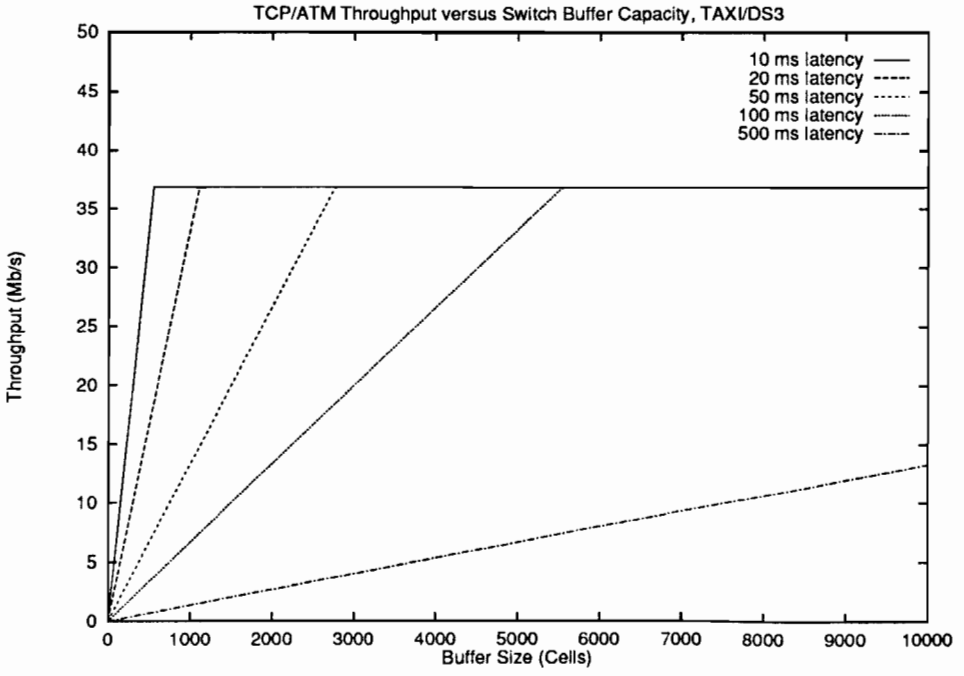


Figure 2-9: WAN Throughput with TAXI to DS3 Mismatch

2.2.1.1 Experiment 1

Figure 2-10 shows the first case. Two Alphas were connected together over a 1100km round trip link. Figure 2-11 shows the throughput for various combinations of window size and MTU size. It is clear from the figure that the window size plays a much greater factor in determining throughputs in the WAN. The plots are very linear for the smaller window sizes showing the lack of effect the segment size has in the WAN.

Figure 2-12 shows a comparison between the throughput seen in the LAN and in the WAN test cases for the same segment size of 9180 bytes. In the LAN the throughput varies more with segment size than in window size, but the long delay path in the WAN overshadows the segment processing overhead.

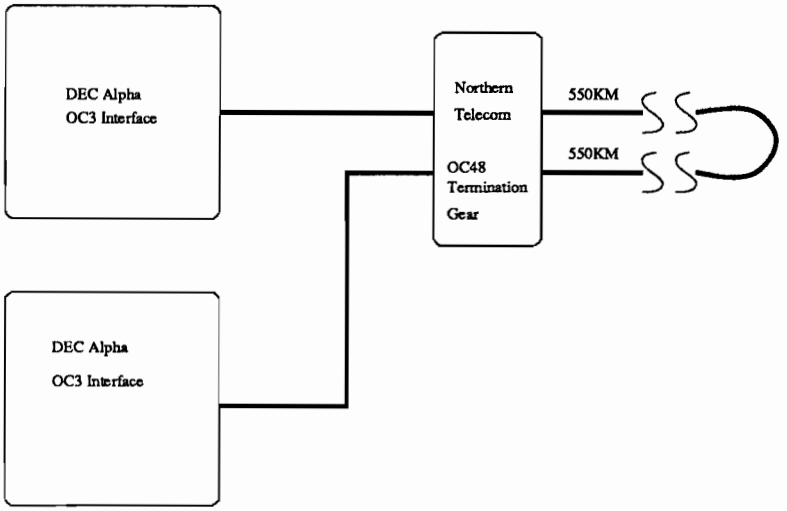


Figure 2-10: WAN test case with no congestion

2.2.1.2 Experiment 2

The second experiment involved a DEC Alpha with an OC3 card transmitting to a Sun Sparc 10 with a 100 Mb/s TAXI interface over a distance of 500 km. The round trip time was 8.8 ms. Figure 2-13 shows the test setup. The rate mismatch caused by going from a 155Mb/s OC3 link to a 100Mb/s TAXI link introduced

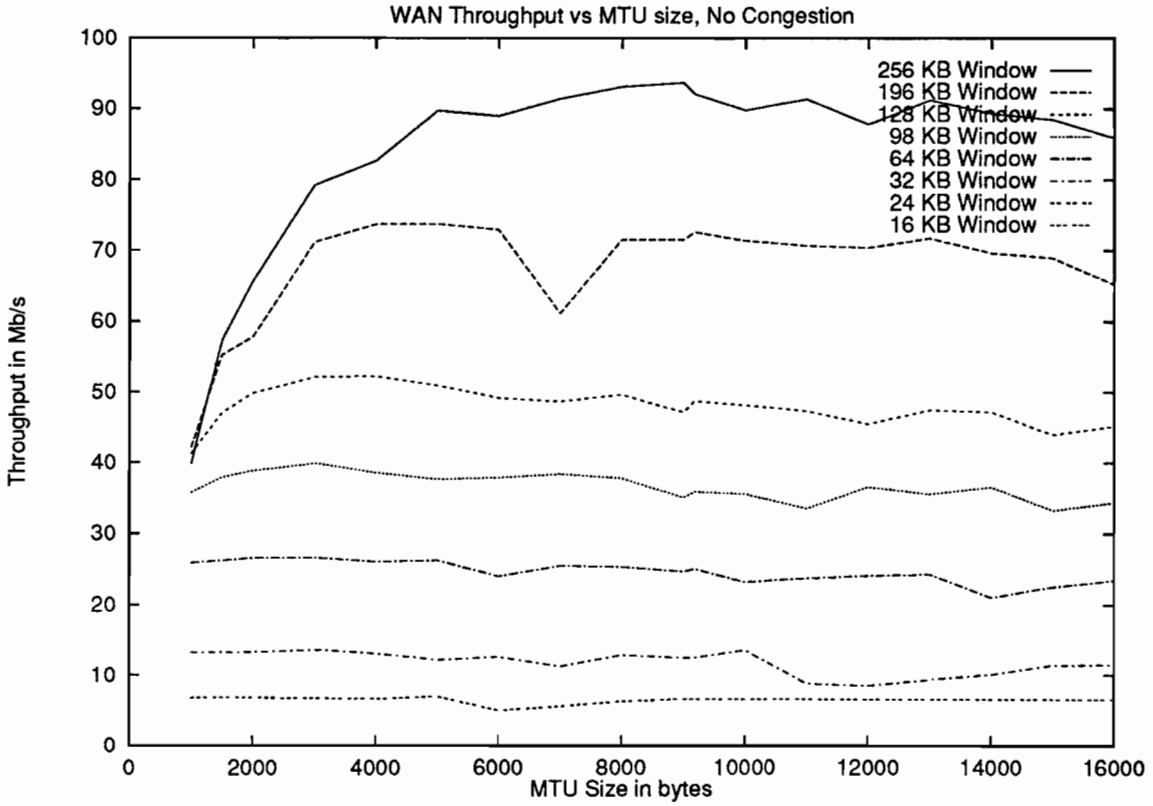


Figure 2-11: Throughput for various MTU's and windows in the WAN

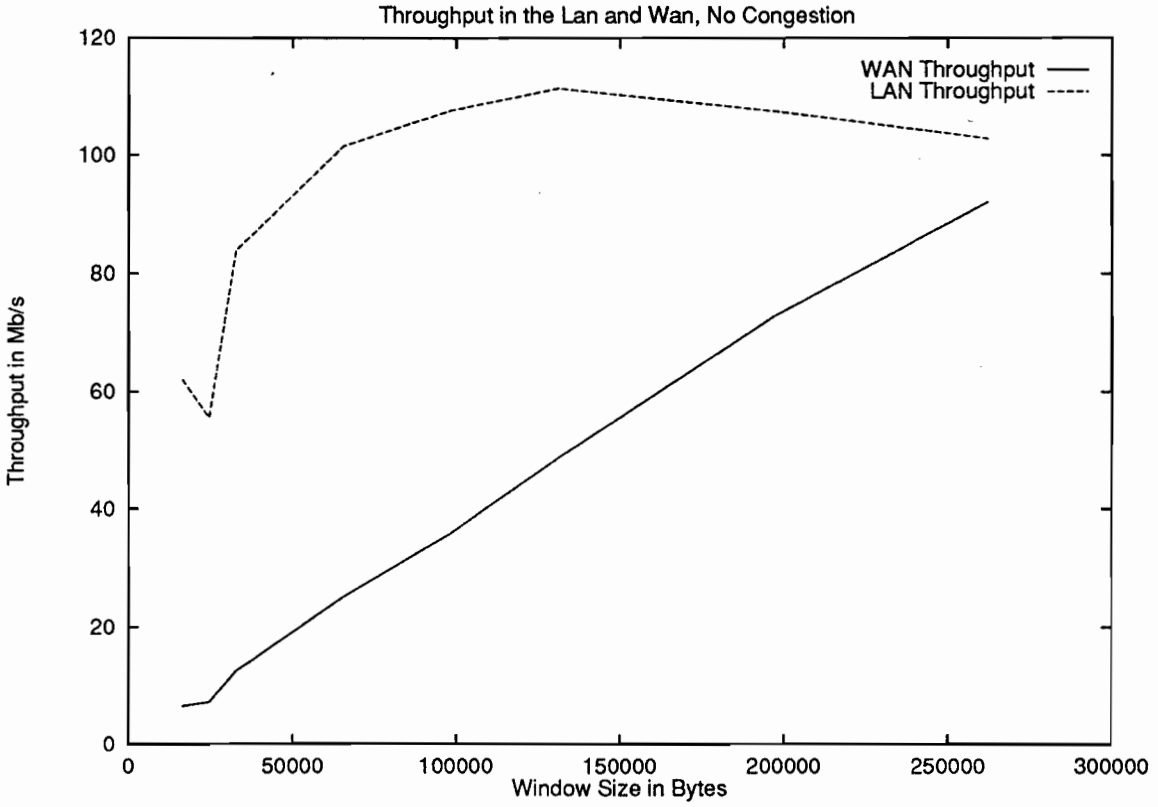


Figure 2-12: Throughput vs window size, MTU = 9180 bytes

congestion in the FORE switch. Figure 2-14 shows the throughput for various window sizes and mtu sizes.

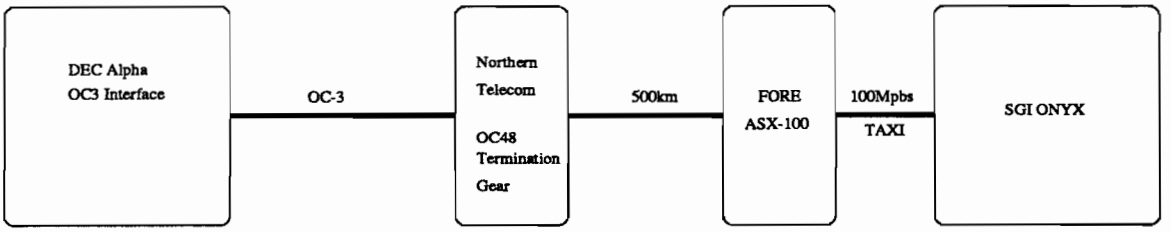


Figure 2-13: WAN test case with no congestion

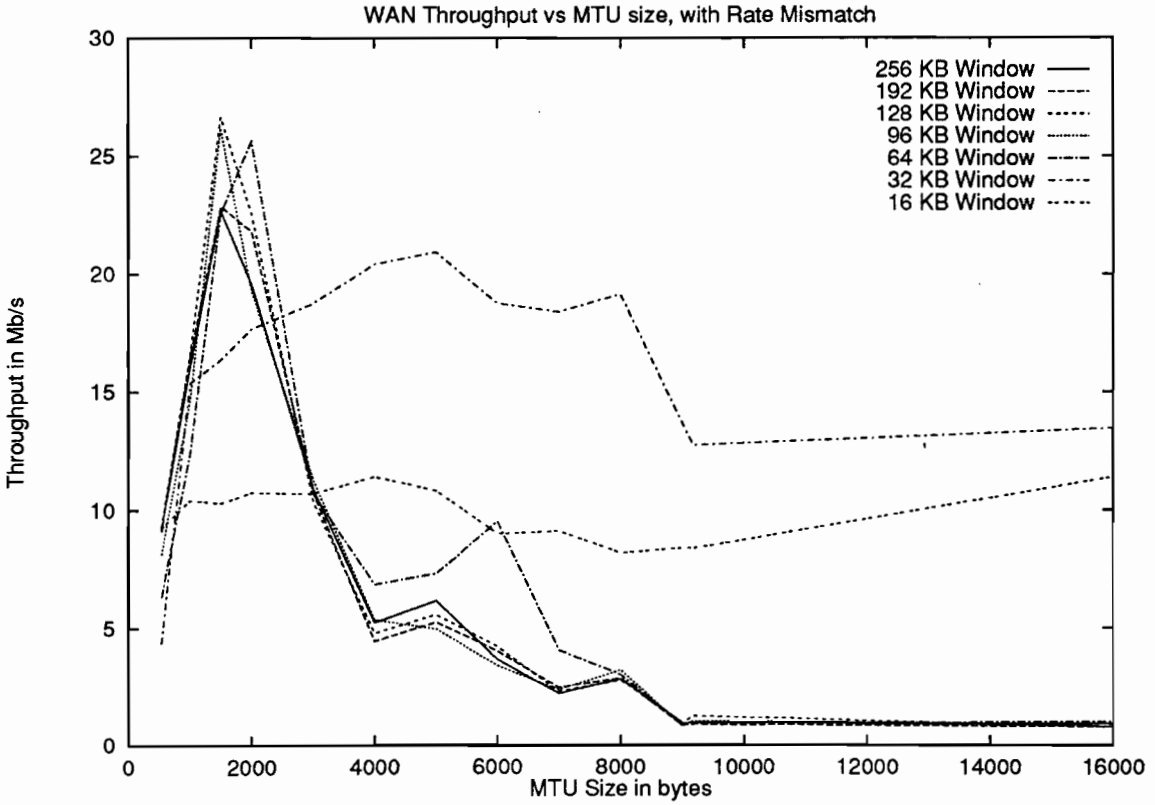


Figure 2-14: Throughput for various MTU's and windows in the WAN with rate mismatch

2.2.1.3 Experiment 3

The third set of experiments involved using two DEC Alphas transmitting to an SGI Onyx. Congestion was due not only to the rate mismatch from the OC-3 to

the TAXI link, but also because two hosts were sending traffic at the same time. Figure 2-15 illustrates the test setup, and Figure 2-16 shows the throughput for various segment and window sizes.

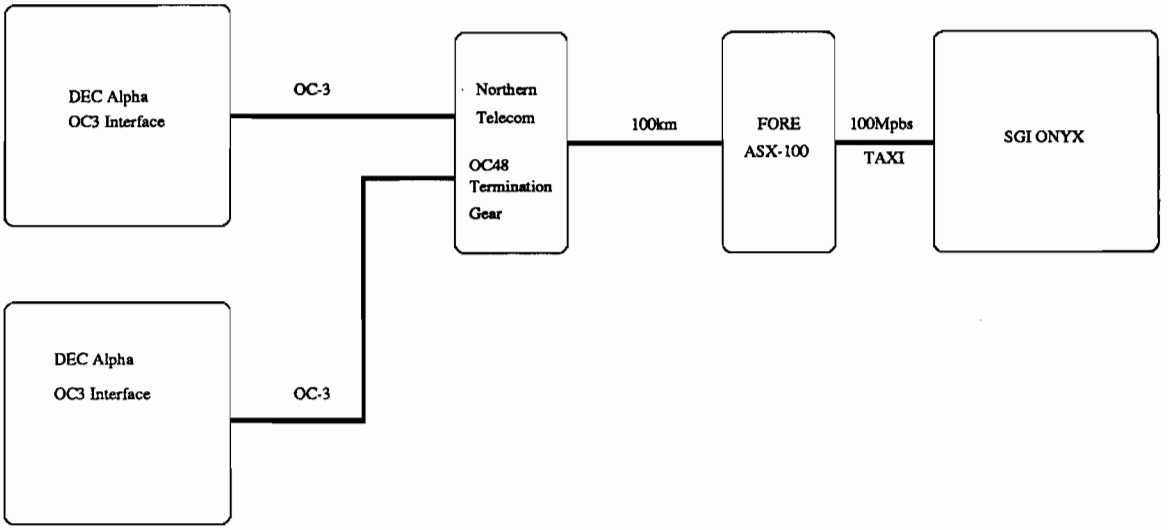


Figure 2-15: WAN test case with congestion

In this case it can be beneficial to use smaller segments to help reduce burst sizes, even though this incurs additional overhead processing delays. The overall performance can be improved by using smaller windows or smaller segment sizes. The link utilization is very low, however, varying between 0.2 and 0.3.

Figures 2-14 and 2-16 show a dramatic drop off for window sizes larger than 32 KB. The FORE ASX-100 switch used in these experiments has a per port buffer of 256 cells. Using that queue size, and the rate mismatches shown in Figure 2-15 we can solve for the number of bytes needed to overflow the port queue on the FORE switch. Solving equation 2.4 for this case shows that a window greater than 34,630 bytes will results in an overflow. This matches our experimental results superbly.

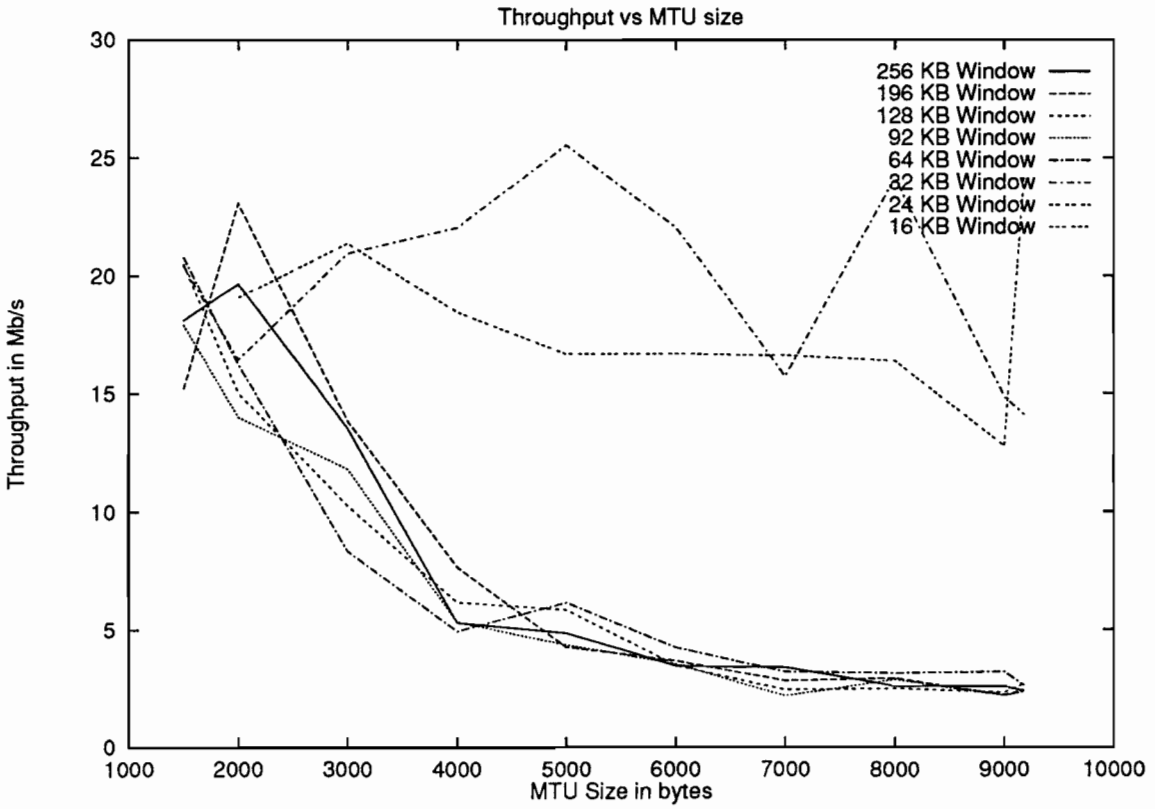


Figure 2-16: Throughput vs window size for various MTUs

2.2.2 Simulations of WANs

Netsim was used to simulate the case seen in experiment 1. Figure 2-17 shows the experimental results seen previously compared with the throughputs found in the simulation.

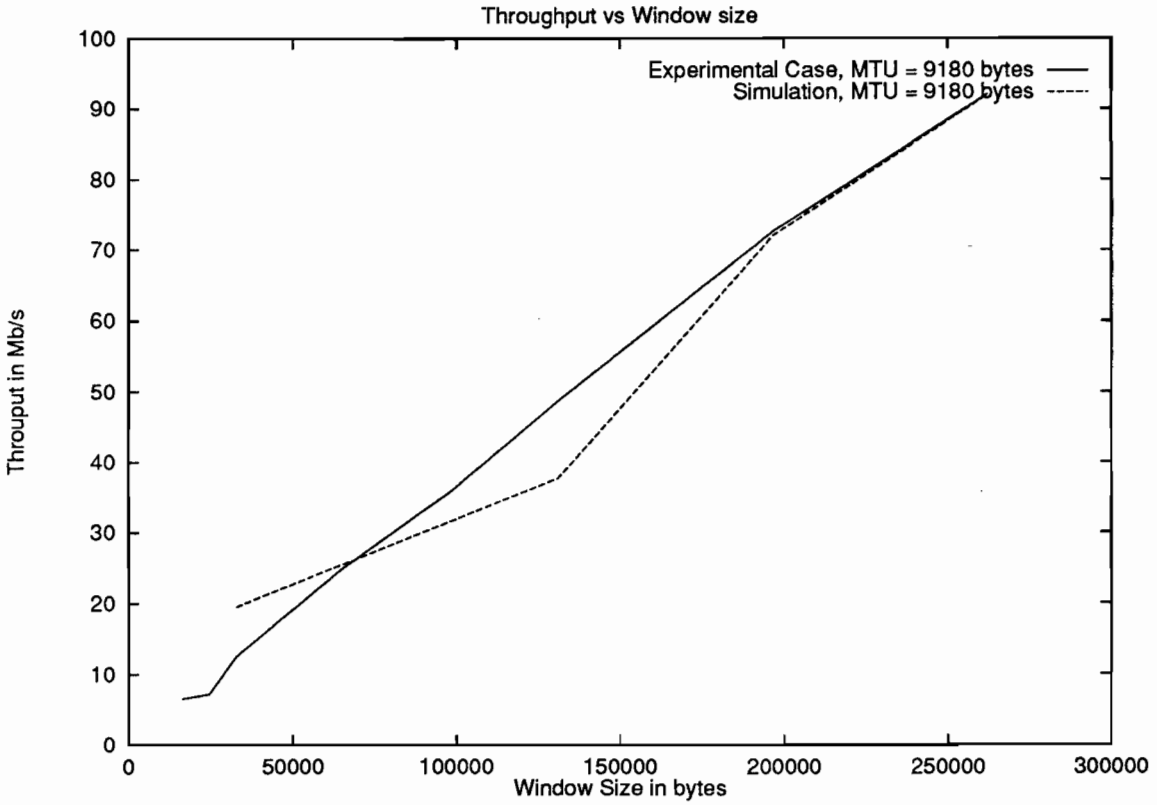


Figure 2-17: Throughput vs window size for a MTU of 9180 bytes

The close match again serves as a reassurance that we are modelling the events correctly.

In this chapter we have shown that several factors particular to ATM networks influence the performance of TCP/IP based applications. Cell loss is the most important parameter of an ATM network. The cell loss rate is influenced primarily by congestion in the network. Congestion can be the result of hot spots in the traffic patterns as well as rate mismatches in link speed. Additional constraints on the maximum throughput include losses from the overhead being transmitted,

a 13.5% reduction for TCP/IP using AAL5 over SONET, and the time needed to process this overhead.

Chapter 3

Analytical Models of ATM Switch Buffers

TCP/IP throughput in ATM systems is strongly influenced by the cell loss rate [11]. Simulations indicate that cell loss rates lower than 1% are needed to provide reasonable throughput. The relationship between segment size and buffer space needed in a switch for a given loss rate is important, as illustrated by the results in the previous chapter. In order to understand this relationship, we assume a given cell loss rate and use several models to find the number of cell buffers needed to support that rate.

3.1 ATM Queueing Model

ATM switches route each cell according to its output port address and transmits the cell to its destination. A large number of alternative switching architectures have been described in literature, e.g. [1, 15] and some of them have become commercial products, or are the focus of current research. One of the basic functions of a switching element is the buffering of cells destined to the same port. ATM based networks operate in a slotted time scheme, and it can be considered that

all cell transitions, cell arrivals and cell departures occur at the boundaries of these time slots. The different services expected to be used over ATM networks have widely different traffic patterns in terms of mean holding time, bit rates, burstiness, and other characteristics.

In order to analyze the behavior of an ATM queue, analytical models must be developed. The simplest model is known as an M/M/1/N model. In a M/M/1/N model, arrivals to the queue are assumed to be independent exponentially distributed random events. Departures are also Poisson processes. There is assumed to be a single server emptying the queue, and the queue has a finite capacity N.

A discrete time analog of the M/M/1/N model is the Geom/Geom/1/N. It is assumed that the interarrival time and interdeparture times between cells are geometrically distributed. Geom/Geom/1/N models are simple discrete Poisson processes. While Poisson processes are well understood and allow relatively simple computations, they are memoryless and therefore do not accurately model the burstiness and correlation characteristics of many traffic sources.

There are other queueing models extensively used in the analysis of ATM networks. The On/Off traffic model is often used in modeling the queueing behavior of ATM switches. Instead of assuming an iid Bernoulli process for the cell arrival process, On/Off traffic models take into account the statistical fluctuations in the call set up time and cell generation for individual calls. In the On/Off traffic model, a call, during its holding time, is described to evolve between 2 states, the On state (Active state) and the Off state (Silent state), which are both governed by a discrete Markov chain. When in the On state, the source generates a cell at a constant rate. This constant rate is called the peak rate of the On/Off source.

Recent work by Hzhu [33] showed that many different models including Geom/Geom/1/N, On/Off sources, and real trace data from a Bellcore video source, predict the same relationship between the buffer size and the cell loss rate. Because of these results, the more general models will be explored in this

work.

3.2 M/M/1/N

An M/M/1/N model assumes the interarrival and service times are independent exponentially distributed random variables. There is a single server servicing the queue, and the queue has finite size N. If a state is defined as have the number n items in a queue then the state transition probabilities can be expressed as

$$P_n = (\frac{\lambda}{\mu})^n P_0, \quad 0 \leq n \leq N \tag{3.1}$$

with

$$P_O = \frac{1}{\sum_{n=0}^{N} (\frac{\lambda}{\mu})^n} \tag{3.2}$$

where P_n is the probability of being in state n. If we solve for the probability that there are N in the queue where N is the size of the buffer we get the blocking probability:

$$P_b = \frac{1 - \frac{\lambda}{\mu}}{1 - (\frac{\lambda}{\mu})^{N+1}} (\frac{\lambda}{\mu})^N \tag{3.3}$$

A program was written that used this equation to solve for the blocking probability for an entire range of loads, buffer sizes and segment sizes. A search was done through this solution space and curves were plotted for various probabilities of blocking. The analysis was done at the TCP level where a segment was the smallest unit and the buffer held N segments. The results were then expressed in terms of cells.

Figure 3-1 shows the load attainable for various segment sizes, and buffer sizes for a loss rate of 1%. It can be seen that sending smaller segments allows higher possible utilization for a given buffer size, as expected. It also is seen that relatively large buffers, greater than 2000 cells, are needed to fully utilize a link

with 9180 byte segments.

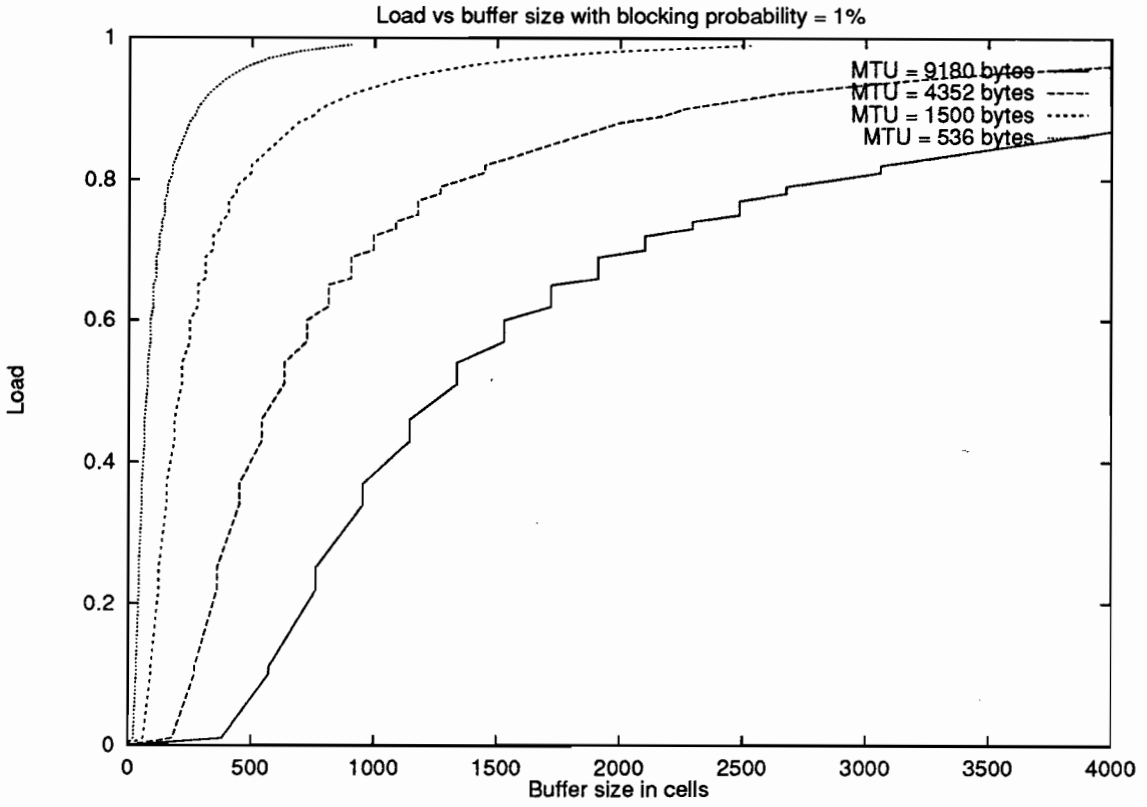


Figure 3-1: Maximum utilization for various segment sizes, blocking prob = 1%

Figure 3-2 shows that if a 5% cell loss rate can be tolerated the required buffer size for a given utilization and segment size is significantly lower.

3.3 Geom/Geom/1/N

In a discrete time queueing network, time is slotted. Traffic governed by a discrete Poisson process has a geometrically distributed interarrival time. Assuming a geometrically distributed message length and a single server queue with finite buffer size, the queueing system is said to be a Geom/Geom/1/N model, where N is the buffer size. In a non cut-through system, when a customer arrives to an empty queue they must wait until the next slot for service. The equilibrium state probabilities for such a system can be expressed as [24],

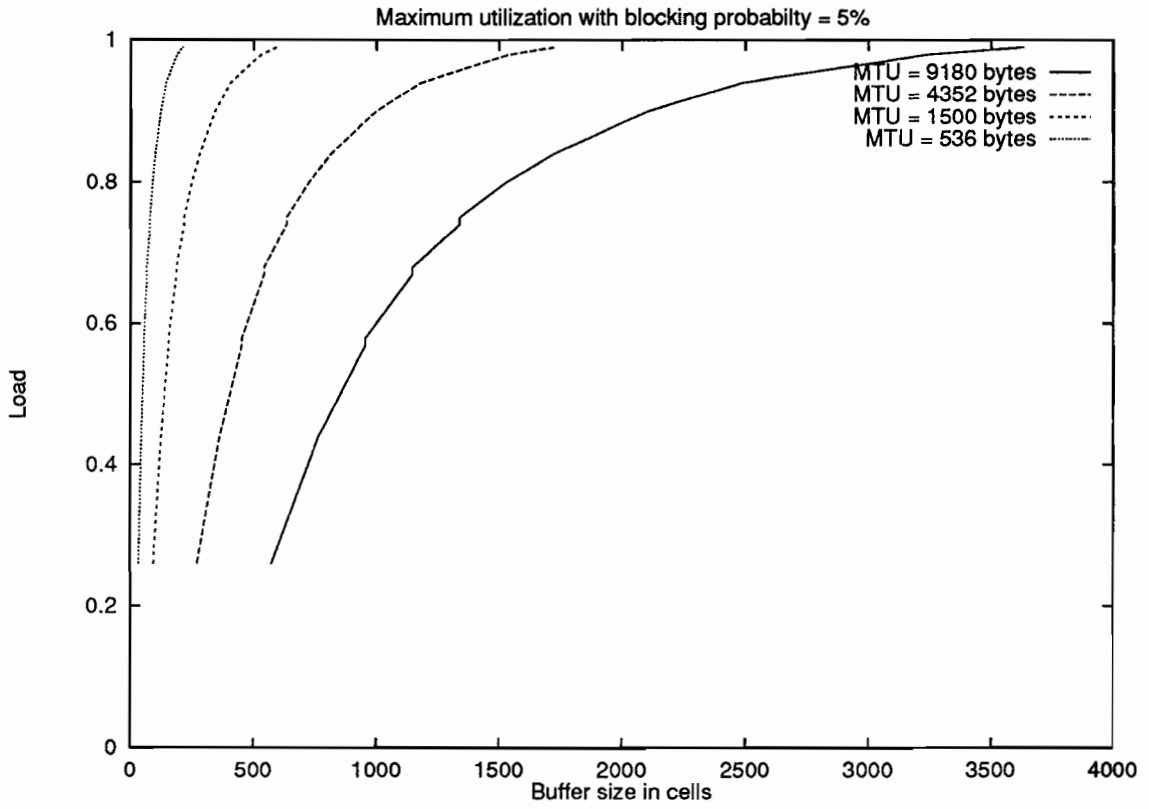


Figure 3-2: Maximum utilization for various segment sizes, blocking prob = 5%

$$P_n = (\frac{p^n(1-s)^{n-1}}{s^n(1-p)^n})P_0, \quad n = 1, 2, ...N, \tag{3.4}$$

where p is the arrival probability of a customer during a time slot and s is the service completion probability of a customer during a slot. Since the interarrival time and interdeparture time are both geometrically distributed, the mean interarrival time is $(1-p)/p$ and the mean interdeparture time is $(1-s)/s$. Therefore the mean traffic load ρ is

$$\rho = \frac{p(1-s)}{s(1-p)} \tag{3.5}$$

Solving for P_0 through the conservation of probability relationship results in

$$P_0 = \frac{1-s}{\frac{1-\rho^{N+1}}{1-\rho} - s} \tag{3.6}$$

Now consider the blocking probability,

$$\begin{aligned} P_b &= Prob(Arriving\ Customer\ is\ Blocked), \\ &= Prob(Queue\ Full) \times Prob(No\ Service\ Completion), \\ &= P_N(1-s). \end{aligned} \tag{3.7}$$

From Equation 3.4, 3.6 and 3.7, we can get the blocking probability for a Geom/Geom/1/N system as follows:

$$P_b = \frac{(1-\rho)\rho^N(1-s)}{1-\rho^{N+1}-(1-\rho)s} \tag{3.8}$$

Figure 3-3 shows the loads that can be supported with a cell loss probability no greater than 1% for several different MTU sizes and buffers sizes.

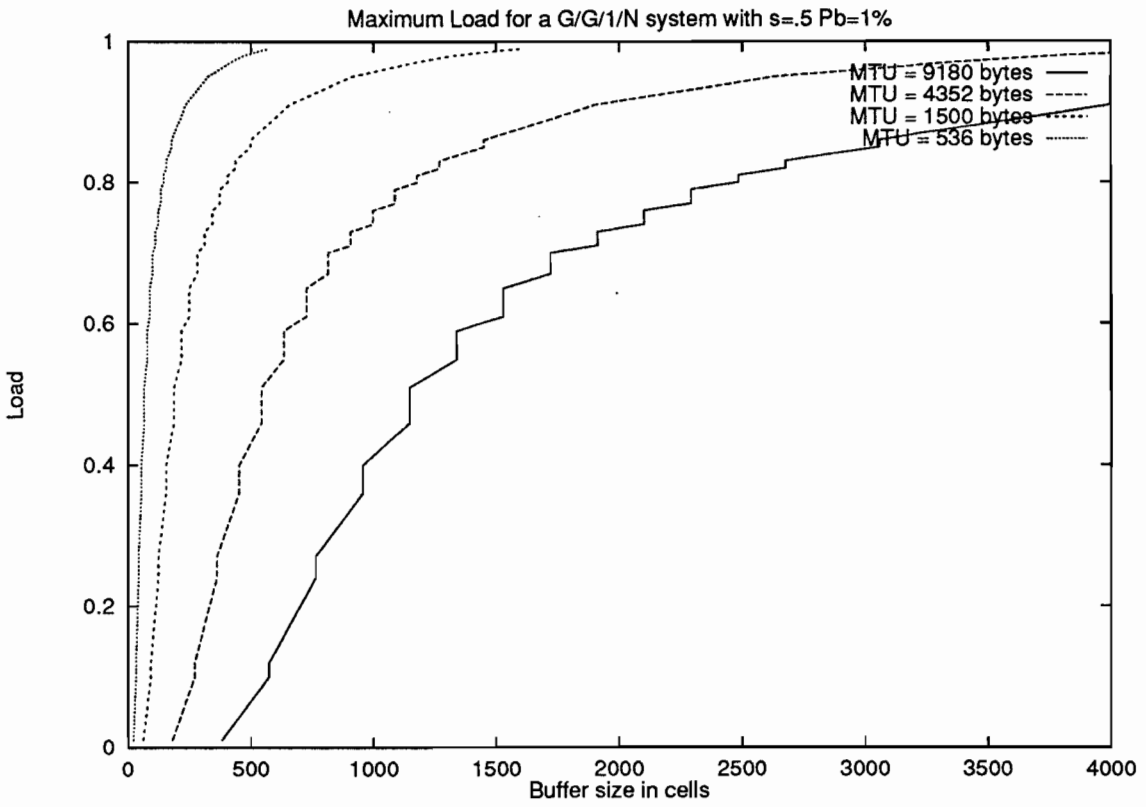


Figure 3-3: Maximum utilization for various segment sizes with blocking probability = 1%

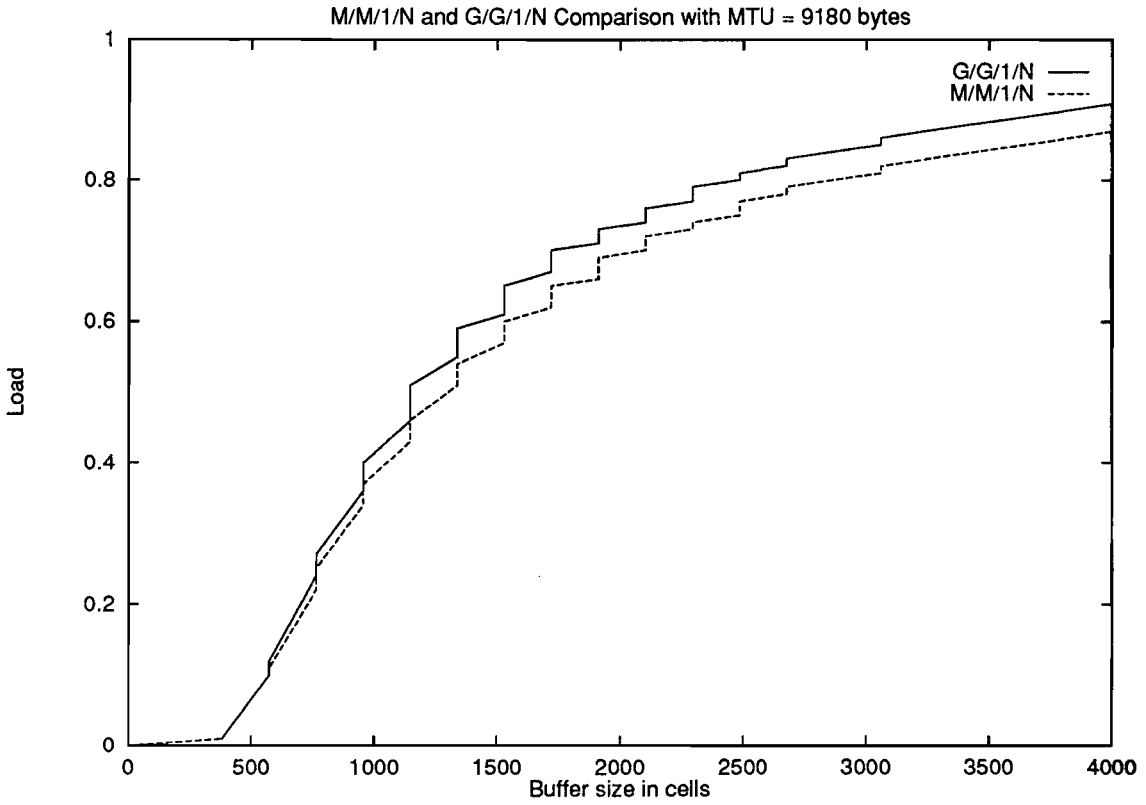


Figure 3-4: Comparison of M/M/1/N and Geom/Geom/1/N models

Figure 3-4 compares the Geom/Geom/1/N model's results with those from the M/M/1/N model with a cell loss probability of 1%. The Geom/Geom/1/N model predicts a slightly smaller buffer than the M/M/1/N model, as it assumes a fewer sources contending for a given output port.

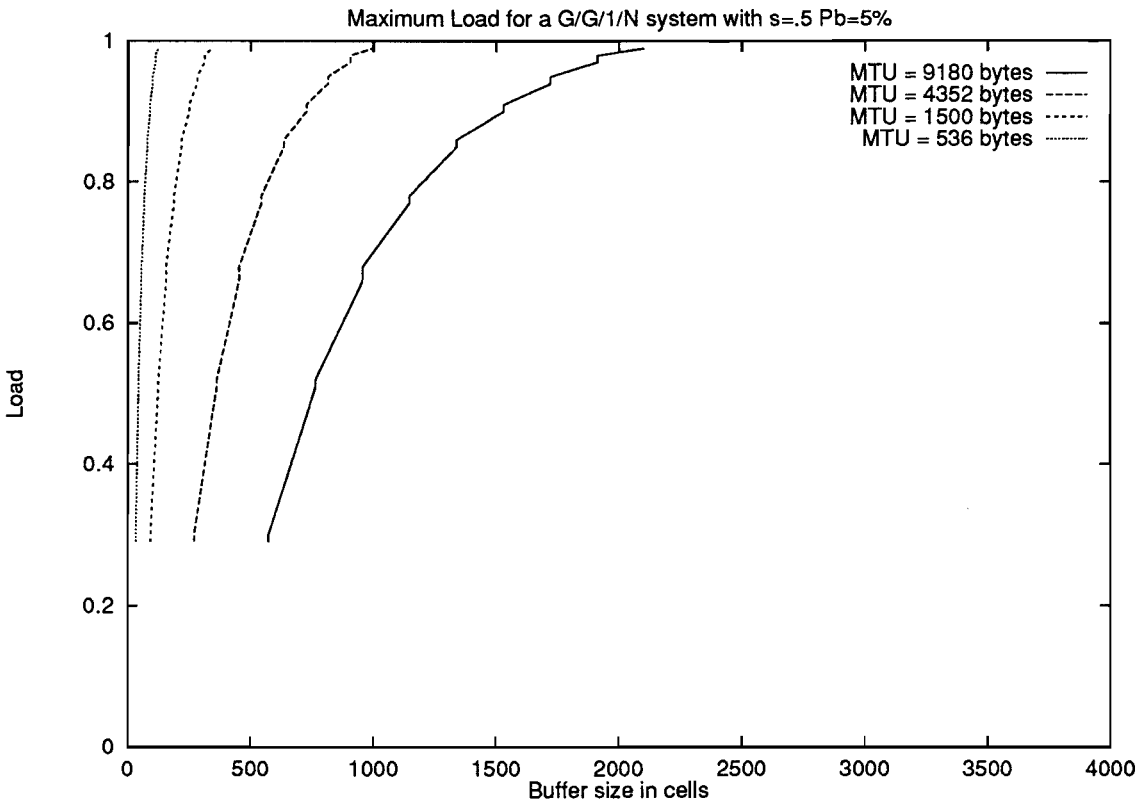


Figure 3-5: Maximum utilization for various segment sizes with blocking probability = 5%

Figure 3-5 illustrates that much less buffer space is needed for a given MTU size if a cell loss rate of 5% can be tolerated.

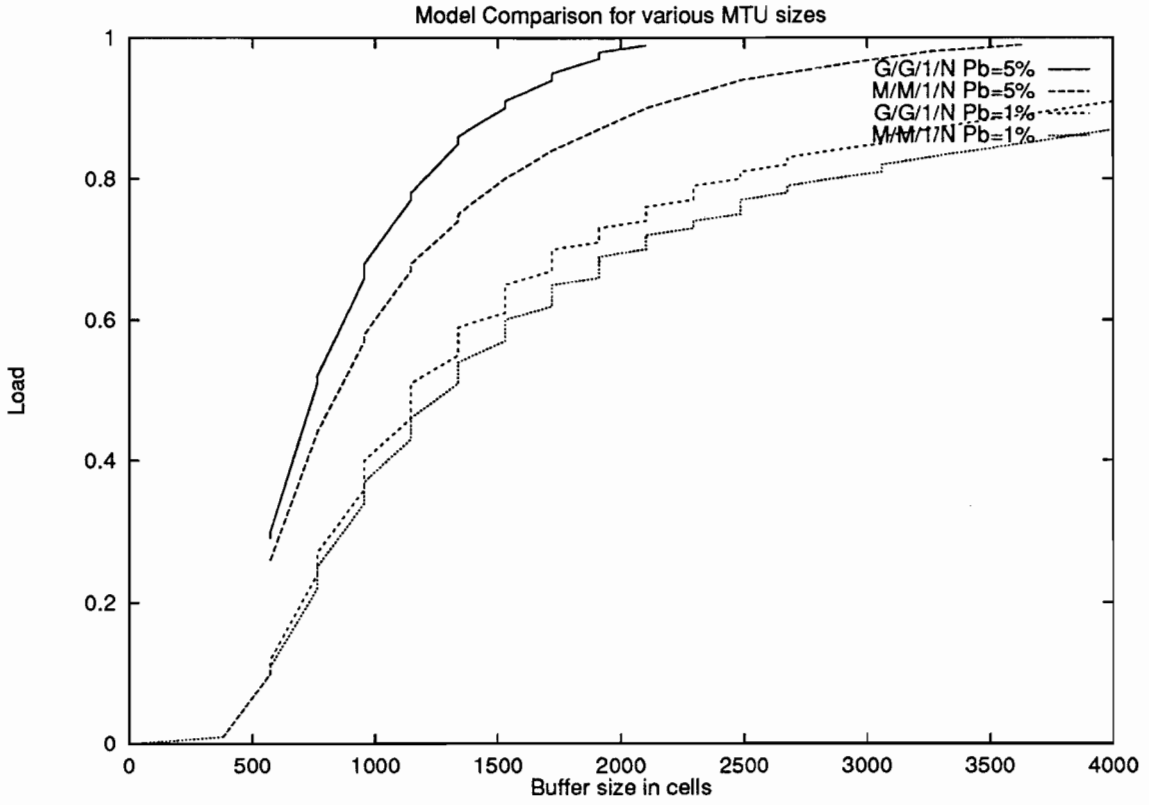


Figure 3-6: Maximum utilization for various segment sizes and loss rates

Figure 3-6 compares both the Geom/Geom/1/N and the M/M/1/N models for cell loss probabilities of 1% and 5%.

This chapter focused on several analytical models to explore the size of the buffers needed to fully utilize network links. TCP/IP was shown in the previous chapter to have inadequate control if the cell loss rate was higher than 1%. The results from this chapter show that without some ATM level mechanism to keep cell loss low, large buffers on the order of several thousand cells per port are needed to fully utilize the network.

Chapter 4

Source Based Cell Level Pacing

From the previous chapters it can be seen that TCP/IP's rate control mechanism does not have the granularity to perform optimally when losses occur at the ATM cell level do to buffer overflow. Figure 4-1 indicates that the effectiveness of TCP's rate control improves as the segment size decreases relative to the available buffers. Figure 4-2 illustrates the effect of window-based pacing, that is, when the window size is sufficiently small, the throughput is limited by the round-trip delay, rather than switch congestion. While both of these cases suggest methods of improving throughput, neither allows the link to be fully utilized.

A more appropriate mechanism for improving TCP/IP's control in ATM networks is a cell level pacing. This pacing is done at the source host of each data stream, and is a function of the host interface to the ATM network. Cell level pacing is implemented as a transmission schedule, in which a time slot is allocated to a particular VC. If no traffic needs to be sent on that VC during that time slot, the slot is available for use by other VCs. For example, VC_i in Figure 4-3 will use up to half of the available bandwidth if there is any traffic to send on that VC. When ABR traffic is paced it has the effect of allocating a fixed upper bound on the bandwidth available to the customer, and it distributes the traffic uniformly in time, preventing bursts.

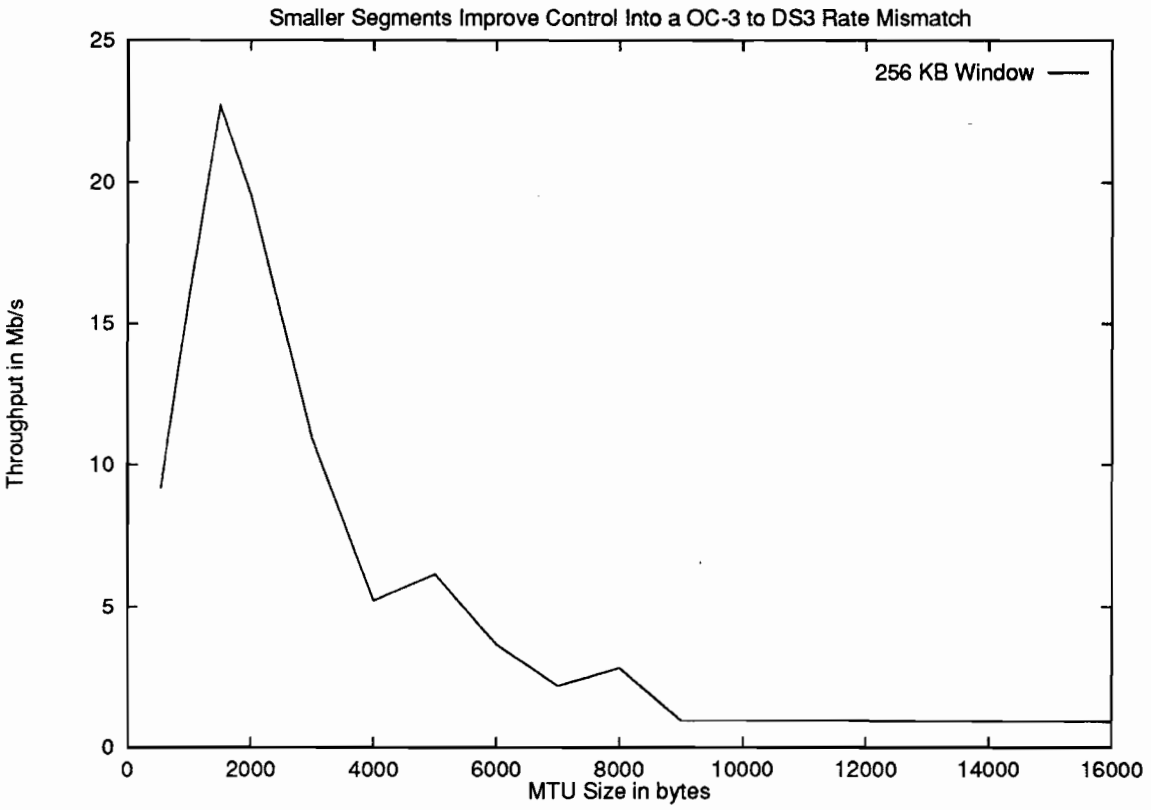


Figure 4-1: TCP's control improvements for smaller segments

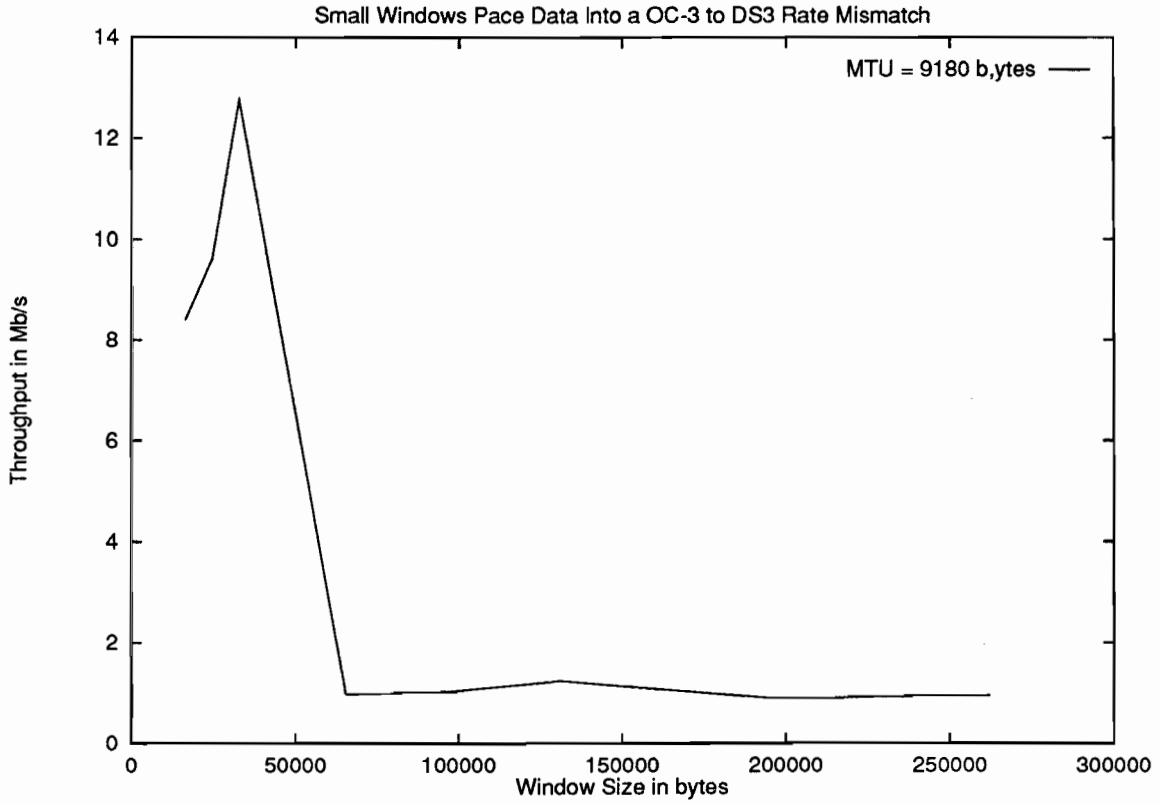


Figure 4-2: Window Pacing

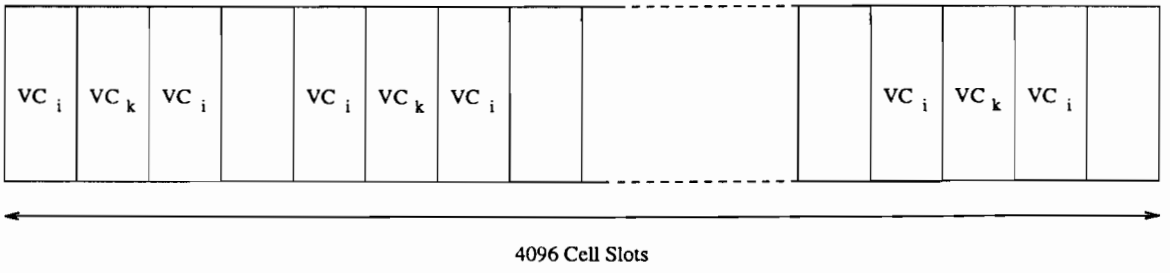


Figure 4-3: Pacing Schedule

Table 4.1: Throughput with Rate Mismatches in an ATM WAN

No Pacing	Pacing
0.87 Mb/s	68.20 Mb/s

The Digital OTTO OC-3 network interfaces have pacing implemented on a per VC basis. These interfaces allowed us to experiment with different rates to keep from overflowing switches downstream. Several experiments are detailed in the next section.

4.1 Experiments

4.1.1 Experiment 1

First, cell level pacing was used to try and prevent the buffer overruns seen when rate mismatches occur in the network. A Digital DEC 3000 AXP (OC-3c) in Lawrence, Kansas transmits to a SPARCstation-10 (TAXI) at EDC in Sioux Falls, South Dakota (600 km), that is, a single host transmits to another host with a 155 Mb/s to 100 Mb/s bandwidth constriction in the path. The experimental conditions in this case included 128 KB TCP windows and 64 KB ttcp write buffers. The ATM pacing in this case was fixed at a bandwidth of 70 Mb/s.

The results are shown in Table 4.1. They demonstrate that cell-level pacing successfully compensated for rate mismatches in the ATM network, while TCP rate control alone was ineffective.

4.1.2 Experiment 2

Second, cell level pacing was used to try and improve performance when multiple high bandwidth TCP sources are multiplexed together at switching points.

Two Digital DEC 3000 AXPs (OC-3c) in Lawrence, Kansas transmit to a

Table 4.2: Combined Throughput with Switch Congestion in an ATM WAN

No Pacing	Pacing
1.66 Mb/s	52.36 Mb/s

SPARCstation-10 (TAXI) in South Dakota (600 km), to evaluate the effect of two hosts causing contention on the switch port to a third host. The experimental conditions included 128 KB TCP windows, 64 KB write buffers in ttcp, and ATM pacing at 35 Mb/s for each source.

The results of Experiment 2 are shown in Table 4.2. These indicate that pacing is very effective in improving throughput.

4.2 Future Work

Continued work is necessary is several areas. The effect of both rate based and credit based flow control algorithms on the traffic characteristics should be investigated. Additional simulation models of the cell level pacing are another area for future research. This will allow the testing of network scenarios that have not been experimentally tried. Making the pacing schedule on the OTTO interfaces respond to external signals such as congestions alarms and pacing requests from network policers is another promising area of future research.

4.3 Conclusions

A study of TCP/IP over ATM based networks was performed with the intention of understanding bottlenecks and improving performance. Buffer overflow in ATM switches due to congestion was found to be the major cause of poor performance. Congestion could be the result of "hot spots" where many streams were multiplexed onto a single link, or from bandwidth mismatches which result from the

many different interface rates found both in new networks, and legacy networks.

TCP's rate control mechanism was found to be inadequate for dealing with events such as cell losses due to congestion. Controlling the size of bursts, and the average rate of sources so that the queue overflow were prevented becomes a necessary task. Using small segment sizes limits the size of bursts and prevents cell loss, but processing overhead limits overall utilization to between 20% and 30% of the total link capacity. Small windows can be used to pace the data as well, but on WAN links, the windows must be large to fully utilize the link.

A cell level mechanism is needed to shape the traffic so that TCP's rate mechanism is adequate. An ideal solution is to use credit based flow control to guarantee zero cell loss. Rate based flow control is an area of ongoing research. Cell level pacing was presented as a solution for dealing with rate mismatches in heterogeneous networks with various link rates. It was shown the pacing allowed the switch buffers to be utilized much more efficiently, providing 80% link utilization in cases that could only utilize 30% of the link with TCP's rate mechanism alone.

Appendix A

C Source Code for Queue Models

```
/*     Markov/Markov/1/N Search Program



*/

#include "stdio.h"
#include "math.h"

struct triplet
{
struct triplet *next;
int buffersize;
double utilization;
double poverflow;
};


struct triplet *create_link(int buffer,double util,double pover);

struct triplet *head = NULL;
char *malloc();
```

```
print_list(space)
struct triplet *space;
{
 while (space != NULL)
{
printf("Buffersize = %d\n", space->buffersize);
printf("Util = %f\n", space->utilization);
printf("Pover = %f\n", space->poverflow);
space = space->next;
}
}
search_list(space, value)
struct triplet *space;
double value;
{
double previousputilization;
previousputilization= 1.0;

 while (space != NULL)
{
if (space->poverflow <= value+.002 && space->poverflow >= value-.002){
if (space->utilization < previousputilization) {
printf("%d %f\n", space->buffersize, space->utilization);
/* printf("Util = %f\n", space->utilization);
printf("Pover = %f\n", space->poverflow);*/
previousputilization = space->utilization;
}
}
space = space->next;
}
}

add_triplet(buffer,util,pover,space)
struct triplet *space;
int buffer;
double util;
double pover;
{
struct triplet *next_link, *newlink;
newlink = create_link(buffer,util,pover);
next_link = space->next;
```

```
space->next = newlink;
newlink->next = next_link;

}

struct triplet *create_link(buffer,util,pover)
int buffer;
double util;
double pover;
{
 struct triplet *newlink;
 newlink = (struct triplet *)malloc(sizeof(*newlink));
newlink-> next=NULL;
newlink->buffersize=buffer;
newlink->utilization=util;
newlink->poverflow=pover;
return(newlink);
}

void main(void)
{
struct triplet *myspace;
double value = .05;
int buffers;
int cellbuffers;
double utils;
double perror;
double temp;
double temp1;
double temp2;
myspace = create_link(0,0.0,0.0);
for (cellbuffers= 10; cellbuffers < 10000; ){
for (utils = 1; utils < 100; utils++){
temp2 = utils/100;
buffers = (48*cellbuffers)/536;
temp = pow(temp2,buffers+1);
temp1 = pow(temp2,buffers);
perror = ((1 - temp2)/(1 - (temp))*(temp1));
add_triplet(cellbuffers,temp2,perror,myspace);
}
cellbuffers= cellbuffers + 1;
}
```

```
search_list(myspace,value);
}
```

```
/*    Geometric/Geometric/1/N Search Program



*/
#include "stdio.h"
#include "math.h"

struct triplet
{
struct triplet *next;
int buffersize;
double utilization;
double poverflow;
};


struct triplet *create_link(int buffer,double util,double pover);

struct triplet *head = NULL;
char *malloc();

print_list(space)
struct triplet *space;
{
 while (space != NULL)
{
printf("Buffersize = %d\n", space->buffersize);
printf("Util = %f\n", space->utilization);
printf("Pover = %f\n", space->poverflow);
space = space->next;
}
}
search_list(space, value)
struct triplet *space;
double value;
{
double previousputilization;
previousputilization= 1.0;

 while (space != NULL)
```

```
{
if (space->poverflow <= value+.002 && space->poverflow >= value-.002){
    if (space->utilization < previousputilization) {

printf("%d %f\n", space->buffersize, space->utilization);
        previousputilization = space->utilization;
                }
}
space = space->next;
}
}

add_triplet(buffer,util,pover,space)
struct triplet *space;
int buffer;
double util;
double pover;
{
struct triplet *next_link, *newlink;
newlink = create_link(buffer,util,pover);
next_link = space->next;
space->next = newlink;
newlink->next = next_link;

}

struct triplet *create_link(buffer,util,pover)
int buffer;
double util;
double pover;
{
 struct triplet *newlink;
 newlink = (struct triplet *)malloc(sizeof(*newlink));
newlink-> next=NULL;
newlink->buffersize=buffer;
newlink->utilization=util;
newlink->poverflow=pover;
return(newlink);
}

void main(void)
{
```

```
struct triplet *myspace;
double value = .05;
int buffers;
int cellbuffers;
double utils;
double perror;
double temp;
double temp1;
double temp2;
double s = .4666;
myspace = create_link(0,0.0,0.0);
for (cellbuffers= 10; cellbuffers < 10000; ){
for (utils = 1; utils < 100; utils++){
temp2 = utils/100;
buffers = (48*cellbuffers)/536;
temp = pow(temp2,buffers+1);
temp1 = pow(temp2,buffers);
perror = (((1 - temp2)* temp1 * (1-s))/(1-temp-((1 - temp2)*s)));
add_triplet(cellbuffers,temp2,perror,myspace);
}
cellbuffers= cellbuffers + 1;
}
search_list(myspace,value);
}
```

Bibliography

[1] H. Ahmadi and Denzel. A survey of modern high-performance switch techniques. *IEEE J. Select. Areas Commun.*, 7(7), 1989.

[2] T. Anderson, C. Owicki, J. Sax, and C. Thacker. High speed switch scheduling for local area networks. In *Proc. ACM Int. Conf. Arch. Supp. Prog. Lang. and Op. Sys.*, Boston, Oct 1992.

[3] R. Atkinson. Default IP MTU for use over ATM Adaption Layer 5 (AAL5). IETF draft-ietf-atm-mtu-07.txt, IETF, Feb 1994.

[4] R. Ballart and Y. Ching. SONET: Now it's the standard optical network. *IEEE Communications Magazine*, 27(3):8–15, March 1989.

[5] C. Braun. Performance evaluation of dynamic and static bandwidth management methods for ATM networks. Master's thesis, University of Kansas, September 1994.

[6] J. Cavanaugh. Protocol overhead in IP/ATM networks. In *MSCI 1994-08-12*, 1994.

[7] C. Chang et al. High-performance TCP/IP and UDP/IP networking in DEC OSF/1 for Alpha AXP. *Digital Technical Journal*, Winter 1993.

[8] A. Chapman, H. T. Kung, and T. Blackwell. Use of flow control for effective statistical multiplexing and notes on implementation. In *ATM Forum Contribution 94-0085*, 1994.

[9] D. E. Comer. *Internetworking with TCP/IP, Volume I.* Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[10] J. Evans. Relationship between queue size, delays, and bandwidth mismatches. personal communication, 1994.

[11] C. Fang, H. Chen, and J. Hutchins. Simulation analysis of TCP performance in congested ATM LAN using DEC's flow control scheme and two selective cell-drop schemes. In *ATM Forum Contribution 94-0119*, 1994.

[12] ATM Forum. ATM User-Network Interface Specification, Version 2.0. ATM Forum, June 1992.

[13] V. Jacobson. Congestion avoidance and control. In *Proc. of SIGCOMM '88.* ACM, Aug 1988.

[14] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. Internet Working Group Request for Comments 1323, IETF, May 1992.

[15] M. Karol and M. Hluchyi. Queueing in high performance packet switching. *IEEE J. Select. Areas Commun.*, 6(9), 1988.

[16] S. Keung and K. Siu. Degradation in tcp performance under cell loss. In *ATM Forum Contribution 94-0490*, 1994.

[17] S. Leffler, M. McKusick, M. Karels, and J. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System.* Addison-Wesley, Readings, Massachusetts, 1994.

[18] D. Martin. *Network Simulator User's Manual.* MIT, 1988.

[19] S. Minzer. Broadband ISDN and asynchronous transfer mode (ATM). *IEEE Communications Magazine*, 27(9), September 1989.

[20] J. Postel. Internet Protocol. Internet Working Group Request for Comments 791, USC/Information Sciences Institute, Marina del Rey, California, Sept 1981.

[21] J. Postel. Transmission Control Protocol. Internet Working Group Request for Comments 793, USC/Information Sciences Institute, Marina del Rey, California, Sept 1981.

[22] M. Prycker. *Asynchronous Transfer Mode: Solution for B-ISDN.* Ellis Horwood, 1991.

[23] I. Richer. The MAGIC project. In *Proc. Fourth Gigabit Testbed Workshop*, Reston, Virginia, June 1993.

[24] Thomas G. Robertazzi. *Computer Networks and Systems, Queueing Theory and Performance Evaluation.* Springer - Verlag, 1994.

[25] A. Romanow and S. Floyd. Dynamics of TCP traffic over ATM networks. In *Proc. of SIGCOMM '94.* ACM, Aug 1994.

[26] W. Stallings. *ISDN and Broadband ISDN.* Macmillan Publishing Company, 1992.

[27] W. R. Stevens. *TCP/IP Illustrated, Volume 1.* Addison-Wesley, Readings, Massachusetts, 1994.

[28] ANSI Committee T1 Contribution T1S1.5/91-449. AAL 5 – A New High Speed Data Transfer AAL. Bellcore Technical Reference Issue 2, IBM et al, Dallas, Texas, Nov 1991.

[29] Bellcore Technical Reference TR-NWT-000253. Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria. Bellcore Technical Reference Issue 2, Bellcore, Dec 1991.

[30] J. Turner. Design of a broadcast packet switching network. *IEEE Trans. Comm.*, COM-36(6):734–743, June 1988.

[31] Curtis Villamizar and Cheng Song. High performance tcp in ansnet. submitted for publication, October 1994.

[32] L. Zhang. Why TCP timers don't work well. In *Proc. of SIGCOMM '86.* ACM, Aug 1986.

[33] H. Zhu. In-service monitoring and estimation of cell loss rate qos in atm network. Master's thesis, University of Kansas, December 1994.