TCP over ATM: Simulation Model and Performance Results

Remi J. Gurski Carey L. Williamson
Department of Computer Science
University of Saskatchewan

Abstract

While the Transmission Control Protocol (TCP) generally provides robust performance across many network environments, several researchers have recently identified the poor end-to-end performance achieved by TCP on Asynchronous Transfer Mode (ATM) networks. The performance problems arise for several reasons: (1) a size mismatch between TCP segments and ATM cells; (2) the simple protocols used for segmentation and reassembly in ATM, and for retransmission in TCP; (3) specific "optimizations" in TCP, made primarily for low-bandwidth Internet environments, which do not work well in high speed ATM networks; (4) features (or misfeatures) in most TCP protocol implementations; and (5) subtle interactions between all of these factors.

This paper describes a simulation model that we have constructed to study TCP performance on ATM networks, as well as a set of simulation experiments conducted using the model. The TCP model is detailed enough to recreate many of the performance problems identified by other researchers, as well as to evaluate potential solutions to the performance problems. The TCP model adds to the traffic modeling toolkit available in our existing ATM network simulator, and enables the further study of performance issues in TCP over ATM.

1 Introduction

The Transmission Control Protocol (TCP) [17] is often described as the "4-wheel drive" of transport-layer protocols, because of its ability to "go anywhere that you need to go". Because TCP is a general purpose protocol, it can operate successfully in any network environment: local area networks (LAN's), metropolitan area networks (MAN's), and wide area networks (WAN's). Furthermore, TCP typically offers reasonable end-to-end performance to users in these environments regardless of the bandwidth and error characteristics of the particular network technology used at the physical layer (e.g., Ethernet, FDDI, satellite, packet radio). The robustness of TCP has contributed to its success in the Internet environment, where it provides the reliable byte stream delivery required by many higher level network applications, such as file transfer (`ftp`), remote login (`telnet, rlogin`), electronic mail (`SMTP`), network news (`NNTP`), and the World Wide Web (`HTTP`).

There is one network environment, however, where TCP does not "go" as smoothly: high speed ATM (Asynchronous Transfer Mode) networks [14]. ATM is a connection-oriented low-layer network technology that relies on fast packet-switching of small fixed-size *cells*. Several recent studies [4, 10, 15, 18, 21] have identified the failings of TCP in ATM network environments. In fact, the study reported in [15] shows that the effective throughput achieved by TCP over a 140 Mbps ATM link can approach *zero* (0.16 Mbps, to be exact) under certain conditions in the configuration and operation of TCP.

These performance problems are largely due to the segmentation and reassembly process required for ATM networks. That is, an ATM Adaptation Layer (AAL) protocol is required to break up each TCP *segment* (e.g., 9180 bytes) into ATM cells (e.g., 192 cells) for transmission on the network, and to reassemble those cells into a TCP segment at the destination. Unfortunately, the reassembly process can succeed only if *all* the cells of a TCP segment are received. The loss of a single ATM cell in the network (which is supposed to be very rare) means that the entire TCP segment is effectively lost, and must be retransmitted in its entirety (i.e., 192 cells) by the sender. The performance of TCP over ATM in the presence of cell loss gets even worse when the TCP segment size is larger.

There are several other "optimizations" that have been made to TCP, primarily for the Internet environment, which serve to further degrade the performance of TCP on ATM networks. For example, the delayed acknowledgement strategy[1] and Nagle's algorithm [16] both interact in subtle ways with the socket buffer copy rules in most TCP implementations to produce very low TCP throughput [15].

The end result is quite ironic: the TCP protocol provides its *worst* end-to-end performance precisely on the *high speed* ATM networks that are now being deployed. While solutions for many of the performance problems are suggested in the literature [15, 18], the "TCP over ATM" anomaly remains as a curious and interesting phenomenon to study.

The purpose of this paper is to describe a simulation model that we have constructed to study the performance achieved by TCP on ATM networks. The model is detailed enough to recreate many of the per-

[1] Details about the TCP mechanisms mentioned here are discussed in Section 2.

formance problems identified by other researchers, as well as to evaluate potential solutions for the performance problems.

The TCP model has been designed and implemented as part of the ATM-TN (Asynchronous Transfer Mode Traffic and Network) network simulator developed by the TeleSim project [22]. Our TCP model, which has been validated against results reported in the literature and with measurements from the Wnet experimental ATM network in western Canada, allows an ATM-TN simulation user to faithfully model the behaviour of TCP on ATM networks. Thus, the TCP model provides a valuable tool for network planners, complementing the existing set of traffic models in ATM-TN [2].

The remainder of this paper describes the design and implementation of the TCP model, as well as a set of preliminary simulation experiments that have been conducted with the model. Section 2 provides the necessary background on the TCP transport protocol, its mechanisms, and how it operates in ATM networks. Section 3 describes the design and implementation of the TCP model in our simulation environment. Section 4 describes three simulation experiments conducted with the TCP model, identifying some of the performance problems with TCP over ATM. Finally, Section 5 concludes the paper.

2 Background: TCP and ATM

This section provides some background on the protocols and issues involved in transmitting TCP over ATM networks.

2.1 TCP/ATM Protocol Stack

An example of a TCP/ATM *protocol stack* [20] is shown in Figure 1. This protocol stack shows the application interfacing to TCP, which is the end-to-end protocol. After processing by several other levels, data is eventually transmitted over an ATM network. This process is referred to as TCP over ATM, or TCP/ATM.

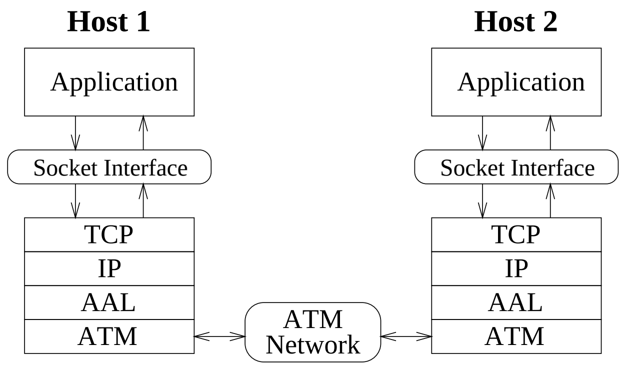

Figure 1: TCP over ATM protocol stack

In Figure 1, the lowest layers, Asynchronous Transfer Mode (ATM) and ATM Adaptation Layer (AAL), are used together to prepare data for transmission over an ATM network. The Transmission Control Protocol (TCP) and the Internet Protocol (IP) comprise the popular TCP/IP protocol suite [5, 19, 23]. TCP/IP provides reliable transmission and addressing for many applications including World Wide Web, `telnet`, and `ftp`. These applications interface with TCP, which transfers data to IP, which in turn uses the underlying network driver (such as AAL/ATM) to transmit and receive data.

The basic unit of data exchange in TCP/IP is the *segment* (sometimes called a *packet*), which consists of zero or more bytes of user data prefixed by TCP and IP headers. The total length of these combined headers is 40 bytes, though each may include additional protocol options. Hosts along the network path use these headers for routing (IP) and end-to-end control (TCP). ATM uses a basic unit of a *cell*, which is of fixed length (53 bytes) and includes routing (switching) information. Conversion between segments and cells is performed by the AAL.

The following subsections provide more detail on the TCP/ATM protocol stack depicted in Figure 1. The discussion begins with the lowest layer (ATM), and proceeds upward through the protocol stack to the application layer.

2.2 ATM

Asynchronous Transfer Mode (ATM) [14] is a method by which fixed-length units called *cells* are individually relayed to a destination. Each ATM cell is 53 bytes in length, which includes a 5 byte header. The header includes two addressing fields: a Virtual Path Identifier (VPI) and a Virtual Channel Identifier (VCI). Other header fields include the payload type and cell loss priority. An error-checking code protects the header from errors.

ATM is able to support a wide assortment of services such as voice, video, and data all over the same network. The fixed size of cells and the simple header layout facilitate very fast hardware switching, making ATM ideal for high speed fiber optic networks.

ATM is a connection-oriented technology. A connection (called a virtual channel) must be explicitly set up end-to-end before any data cells are transmitted. Once set up, the same path is used for all cells of the connection, with each cell carrying the same VPI/VCI. The ATM specification requires that cells be delivered in order on each VCI, as there is no sequence numbering or retransmission provided at the ATM cell level.

2.3 AAL

The ATM Adaptation Layer handles the task of dividing larger data units into 48-byte pieces for transmission within ATM cells. There are several different AAL types available to support the wide range of traffic carried over ATM. The most commonly used AAL for TCP data is AAL5 [1].

The AAL5 protocol breaks large data units (e.g., TCP segments) into as many cells as necessary, adds 0 to 47 bytes of padding to the end of the data, and then inserts an 8 byte trailer in the last cell. The trailer contains the total length of data in the segment as well as an error-checking code. A single bit carried in each ATM cell header indicates which cell contains the trailer (i.e., which cell is the last cell of the segment).

At the receiver, incoming cell data is buffered until the final cell of a segment is received, at which time the length and error codes are checked. If the data is intact, it is passed to the next higher protocol layer (in this case, IP).

2.4 IP

The Internet Protocol (IP) is a connectionless network layer protocol [20]. The IP layer is responsible for routing individual *datagram* packets to their destinations on a "best-effort" basis. End-to-end error control is left to a higher layer, such as TCP.

IP also supports *fragmentation*, in which it divides a large TCP segment into smaller IP datagrams suitable for transmission over the underlying network interface. A network interface usually has a maximum length of data that it can carry as a single unit, called the Maximum Transmission Unit (MTU). The recommended MTU for AAL5 is 9180 bytes [3]; IP fragments any segments longer than this into several datagrams. Since this fragmentation process incurs significant processing overhead, TCP attempts to keep all segment lengths smaller than the MTU.

2.5 TCP

TCP is a transport-layer protocol designed to provide reliable, error-free transmission of data between two hosts [17]. TCP uses IP as a means of transporting an application's data across a potentially unreliable network, and adds facilities to ensure that all transmitted data arrives error-free at the receiving host and is reassembled in order. In contrast to IP, TCP is a connection-oriented protocol; hosts negotiate a connection which lasts for the duration of the data transfer and is then closed.

The service that TCP provides is commonly called *reliable byte stream* delivery. An application presents some data, which is placed in a send buffer. When sufficient data is available, TCP creates a *segment* of user data from the buffer, prepends a TCP header, and sends the segment to IP for transmission. The receiver accepts data units from IP and places the segments in a receive buffer, where they can be retrieved by the application. The longest permitted length of segment data is called the Maximum Segment Size (MSS), which is usually set to the MTU of the network interface minus the size of the TCP/IP headers. If possible, TCP makes all segments equal to the MSS; the exceptions are when the network interface is idle, the application requests immediate delivery of a small amount of data, or the receiver cannot accept a full-sized segment due to a temporary lack of buffer space.

Reliable data transmission in TCP is accomplished through a variety of techniques. The TCP header includes a *checksum* field that can be used to detect transmission errors within a segment. Also, TCP assigns a *sequence number* to each byte of application data, and each protocol header contains the sequence number of its first data byte as well as the length of the data segment. In this scheme, a receiver can detect when a segment has been lost or received out of order by comparing the received sequence number with its expected sequence number.

When a segment arrives in order, the TCP receiver sends an *acknowledgement* (ACK) back to the sender, specifying which segment (byte) to send next. If an out-of-order segment arrives, the receiver sends an acknowledgement specifying that the missing segment(s) should be sent again. The sender maintains a *retransmit timer* for each segment that it sends; if an acknowledgement has not been received for a segment when the timer expires, the sender retransmits that segment. Acknowledgements may be *piggybacked* on data transmissions, so that a TCP/IP packet contains both an ACK for received data (in one direction) and a segment of transmitted data (in the other direction). Receivers are permitted to delay (up to 200 milliseconds) the sending of acknowledgements until additional segments arrive, so that multiple TCP segments can be ACKed at once.

In addition to error control, TCP provides flow control. To accomplish this, a *sliding window* method is used. The sender maintains a *send window* corresponding to the number of bytes (and thus segments) that it can transmit and have outstanding (i.e., unacknowledged) at one time. As segments are acknowledged, the window "slides" and new segments can be transmitted. In this manner, the source can keep the network full of segments without waiting for immediate acknowledgements, which may be subject to long round trip time delays. The receiver also indicates the window size that it can receive (subject to buffer limits), and the sender can use this information to adjust its send window accordingly.

Congestion control is also present in TCP. TCP attempts to estimate how busy the network is and adjust accordingly by maintaining a *congestion window*, in bytes. The maximum number of bytes that a sender can have outstanding is the lesser of the send window and the congestion window. A standard algorithm that makes use of the congestion window is called *slow-start* [11]. When a segment is lost, the send window is set to one segment and increased by one segment as each successfully delivered segment is acknowledged. When the congestion window approaches its previous size (i.e., half the size at which the loss occurred), TCP opens the congestion window more slowly (i.e., one segment per round trip time). This is known as the *congestion avoidance* phase.

Additional enhancements have been made to TCP congestion control in recent implementations. The *fast retransmit* algorithm retransmits a segment when a source receives several duplicate ACKs for the same segment, implying that a segment was lost. The *fast recovery* algorithm avoids performing slow-start after a fast retransmit. Numerous enhancements have also been made to TCP for high speed, long distance networks (e.g., timestamps, larger send windows [12]).

2.6 Application/Socket Interface

In 4.2BSD and later versions, applications communicate with the TCP protocol through network *sockets*. The socket interface includes facilities for making network connections, buffering data, and writing and receiving data streams.

A diagram of protocol processing of user data from the application level to the ATM level is given in Figure 2. Raw data from the application is written, through the socket interface, to the TCP layer. TCP and IP divide the data into TCP/IP segments, each of which contains an MSS length of data. The AAL breaks these segments into 48-byte pieces for inclusion in ATM cells. The last cell includes some padding and the AAL5 trailer.

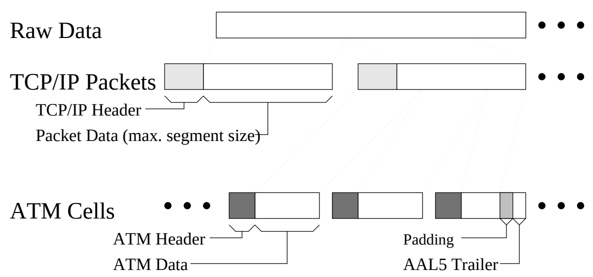

Figure 2: Segmentation of application data for TCP/ATM using AAL5

2.7 Problems with TCP over ATM

TCP/IP traffic is usually carried over *packet-switched* networks, where a packet (segment) is the basic unit. Routing and buffering are performed on the packet level. If the network experiences congestion and switch buffers become full, an entire packet is dropped. In contrast, ATM networks use a cell as the basic unit, and switching and buffering occur on the cell level. When switch buffers fill, cells are dropped.

In ATM networks, then, it is possible for the network to drop only some cells from a segment and deliver the rest of the cells. The remaining cells in such a segment are essentially "dead", since the receiver cannot reconstruct the segment without the lost cells. Transmitting "dead" cells only wastes bandwidth on the already congested network. When the TCP source retransmits the lost segment, even more bandwidth is wasted. The end result is low throughput for TCP over ATM. Some strategies for dealing with "dead" cells include Partial Packet Discard [18], where all *remaining* cells in a segment are dropped following a lost cell, and Early Packet Discard [18], where switches drop *all* cells from a segment when experiencing congestion. These result in much improved performance, compared to TCP over plain ATM [18].

Another problem is a result of the TCP slow-start algorithm. When several sources share a common link and experience congestion, cells from all sources may be dropped [24]. All sources shrink their congestion windows to one segment (the default), and the link can be mostly idle while all sources are in the slow-start phase of the slow-start algorithm. This idle time results in low throughput for all sources.

Other problems with TCP over ATM are a result of the particular implementation. In 4.3BSD, for example, the socket buffer copy rules and delayed acknowledgements can cause very low throughput over ATM networks [15]. The coarse granularity of round trip and retransmission timers can also decrease performance, particularly in high speed networks.

3 TCP/ATM Simulation Model

3.1 Overview

The TCP model is designed to simulate bulk data transfer between two hosts over an ATM network, similar to the FTP protocol. Data transfer can occur in one or both directions. The primary sender is referred to as the *source* and the other host is the *sink*.[2] TCP connection handshaking is not supported, as the model assumes a single connection lasts for the full length of the simulation. A number of options and parameters allow full configuration of TCP and ATM.

The model is closely based on the TCP/IP networking code from the 4.4BSD-Lite release, developed by the University of California, Berkeley and its contributors (April 1994). This implementation is essentially the popular 4.3BSD Reno release with support for high-bandwidth, long-delay paths [12]. The networking implementation in 4.4BSD-Lite has also been called Net/3 [23].

The simulation model of TCP/ATM is written in SimKit [9], a C^{++} based framework with support for sequential and parallel simulation. A full protocol stack is modeled, from the application level down to the ATM level (see Figure 1). The model is divided into four sections: the application level, socket interface, TCP/IP protocols, and AAL/ATM layers.

3.2 Application Layer

The application level performs simulated data writes through the socket interface. Each host is assigned an amount of data to write during a given simulation (or unlimited data). If either host is configured to send zero bytes, it will only send acknowledgements, and the model functions as a one-way traffic source.

3.3 Socket Layer

The socket layer performs buffering of simulated data and communicates with the TCP/IP layer. The socket model includes facilities for simulating the kernel *mbuf* data structure [13]. Socket copy rules and mbufs have been shown to cause low throughput in TCP over ATM in some situations [6, 7, 15]. The model includes parameters for specifying the send and receive buffer sizes and other socket options.

3.4 TCP/IP Layer

TCP is modeled in great detail. The model implementation of TCP is virtually identical to Net/3 and includes the same features such as slow-start, fast retransmit, fast recovery, and high performance extensions [12]. It also supports full-duplex data communication to investigate the effects of piggybacked acknowledgements [24]. The IP layer is not explicitly modeled; instead, all routing takes place at the ATM level, and the TCP model adjusts segment sizes to account for IP headers and fragmentation.

3.5 AAL/ATM Layer

The AAL and ATM layers are combined. The model accepts TCP/IP segments, breaks them up into ATM cells including AAL5 overhead, and queues them for transmission over the simulated network. Received

[2]These terms are somewhat misleading, as both hosts can create and consume traffic.

cells are reassembled into segments, checked for completeness, and passed up to the TCP/IP level if they are valid. ATM level *cell pacing* is supported and is specified as the maximum number of cells to transmit per second.

3.6 Miscellaneous

Almost all socket, TCP/IP, and AAL/ATM features can be controlled through model parameters or compile options. These include timer granularity, maximum segment size (MSS), maximum transmission unit (MTU), and transmission start time. Features can also be disabled, such as the TCP high performance extensions [12] and Nagle's algorithm [16].

A detailed report for each source/sink is generated after each simulation, including bytes sent and received, retransmitted segments, duplicate segments, and dropped cells. Detailed tracing at the TCP or cell level is available for post-simulation processing.

4 Simulation Experiments and Results

A number of experiments have been conducted with the TCP model under various configurations and network loads. The purposes of these experiments were: (1) to validate the TCP model and ensure that it was accurately modeling the behaviour of a TCP source over a real ATM network; and (2) to investigate the interactions of multiple TCP sources when competing for bandwidth, by comparing the individual and aggregate throughput with the network capacity.

4.1 Experimental Setup

Figure 3 shows the network topology used for all the simulation experiments. This topology is intended to represent a typical ATM wide area network. One or more TCP sources connect to their sinks through a switch and a shared link. All sources are configured for unlimited unidirectional data transfer. That is, each source sends data segments and each sink sends only acknowledgements. Network congestion occurs at the output port of the switch.

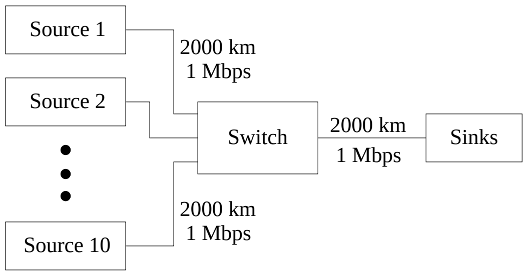

Figure 3: Simulation network topology

All links have a bandwidth of 1 Mbps and a propagation delay of 3.7 μs per km, resulting in a round trip time of approximately 30 ms for each source. The simulated network does not qualify as a high-bandwidth, long-delay link [12], so the TCP high performance extensions were disabled for all experiments. The traffic sources use 64 kilobyte send windows (i.e., send and receive socket buffers) in all experiments.

The switch is a simple single-stage output-buffered model. Outgoing cells are buffered if the output port is busy; if the buffer is full, any new cells for the port are discarded. The output buffer size is set as a simulation parameter. Cells passing through the switch are subject to a simulated lookup delay (1 μs) and internal delay (2 μs), as well as (variable) queueing delays at the output port.

To minimize conflicts between the sources when starting the simulation, the transmission start times of all sources were staggered 100 ms apart. All simulations were run for a 10 second warmup period, during which no statistics were gathered, and then for 300 seconds of simulated time. This allowed for a total data transfer of more than 700,000 cells.

Tests were conducted for maximum segment sizes (MSS's) of 512 bytes and 9140 bytes, which translate into 12 and 192 ATM cells after TCP/IP headers and AAL5 overhead. The first value (512 bytes) is typical for TCP connections over wide area networks, such as the Internet. The second value (9140 bytes) comes from the recommended 9180 maximum transmission unit (MTU) for IP over ATM [3]. This MTU size leaves room for a TCP segment size of 9140 bytes after the 40 byte TCP/IP header.

A commonly used measure of end-to-end protocol performance is *effective throughput*, or *goodput*. This measure is determined from the actual amount of "good" data transmitted over the network (not including retransmissions, lost segments, or duplicate data), divided by the total elapsed time for the transfer. In our simulation results, effective throughput is expressed as a percentage of the maximum possible throughput for a given segment size, which is the maximum link bandwidth adjusted for AAL/ATM and TCP/IP overheads. An effective throughput of 100% is achieved if each packet contains an MSS length segment, the link is never idle, and no segments are ever lost or retransmitted.

4.2 Experiment 1: A Single TCP Source

The first experiment studied the behaviour of the TCP model for a single source in a congested network. A single TCP source was connected to the switch by a 1 Mbps link in the network shown in Figure 3. *For this experiment only*, the bandwidth of the switch output link was lowered to 500 Kbps. The switch buffer was set to 1000 cells and the MSS was 9140 bytes. No simulation warmup period was used in order to view the source start-up dynamics.

In this situation, the TCP source will quickly fill the switch buffer due to the mismatch between the input and output link speeds (1 Mbps vs 500 Kbps). The buffer can hold at most 5 complete TCP segments. Beyond this backlog, the switch will drop cells as necessary, and the source will have to detect and retransmit lost segments. It is expected that the source will exhibit a cyclic behaviour of increasing its send window, filling the switch buffer, detecting a dropped segment, reducing its send window, and retransmitting.

The first ten seconds of the simulation are shown in Figure 4. The upper plot shows transmitted and dropped cells, the middle plot shows the sender's congestion window (in bytes), and the lower plot indicates switch buffer occupancy (in cells). In the topmost plot, transmitted cells (grouped into boxes to show entire TCP segments) are shown *above* the horizontal line, while cell drop events are shown *below* the horizontal line. That is, the lower half of this graph indicates the transmission time of cells that are eventually dropped by the switch buffer.[3] Received acknowledgements are indicated by '×' characters. The horizontal axis in all three plots is the simulated time, in seconds.

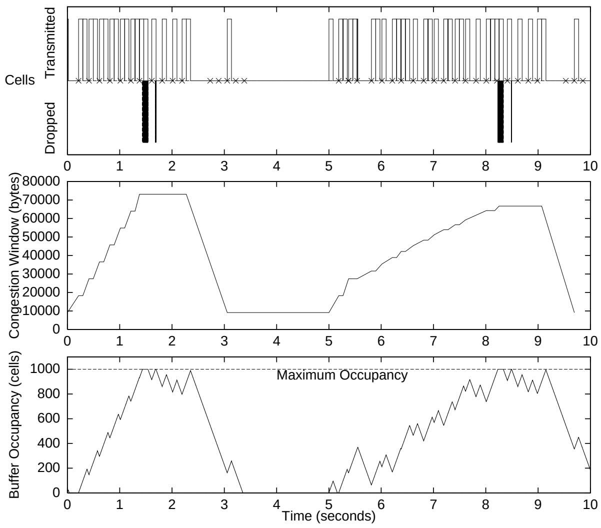

Figure 4: Dynamics of a single TCP source

Figure 4 illustrates a number of the features of the TCP model. First, looking only at the topmost plot (transmitted and dropped cells), one can distinctly see the delayed acknowledgement option, which postpones sending acknowledgements for 200 ms, indicated by the evenly-spaced ACKs. The receipt of each ACK triggers the transmission of a new segment, and often two back-to-back segments when the congestion window is opening in the slow-start phase. The plot also shows the fast retransmit feature, where the sender retransmits a segment after receiving a number of duplicate ACKs. This retransmission occurs at time 3, when the source has received four duplicate acknowledgements and retransmits the first lost segment. The other lost segments are retransmitted when the retransmit timer (2 seconds) expires at time 5.

The middle plot shows the operation of TCP slow-start. The TCP congestion window starts at one MSS (9140 bytes) and opens during the slow-start phase, then levels off due to the imposed limit on the send window (64 kilobytes). When the sender detects a lost segment, it drops the congestion window to one segment, retransmits, and again increases the congestion window, but at a slower rate. When another segment is dropped, the process repeats.

[3] These events appear as solid black boxes because of the time scale used in this figure. The individual cell drops show up more clearly at finer time scales.

Finally, the bottom plot indicates that cells are only dropped when the switch buffer fills (as expected). The source detects the lost segments, retransmits, and resumes transmission which fills the switch buffer again. The source therefore exhibits cyclic behaviour and achieves an average goodput of 61%.

4.3 Experiment 2: Two TCP Sources

A second experiment was run to compare the dynamics of multiple sources with that of a single source. Two sources were connected to the switch and the output link speed was reset to 1 Mbps. As in the first experiment, the switch buffer size was 1000 cells and the MSS was 9140 bytes. The second source was started 10 ms after the first. Each source attempts to capture the full 1 Mbps bandwidth, leading to congestion at the switch output port and dropped cells.

The results of the two-source simulation are shown in Figure 5. Time plots of transmitted and dropped cells are given for both sources, as well as a plot of switch buffer occupancy. Only the first two seconds of the simulation run are shown. In the top two plots of Figure 5, solid lines above the horizontal indicate transmitted cells (and segments), and dashed lines below the horizontal line indicate dropped cells (again grouped into segments, for clarity). The congestion windows used by the two sources are not shown, since their behaviour is very similar to that in Figure 4.

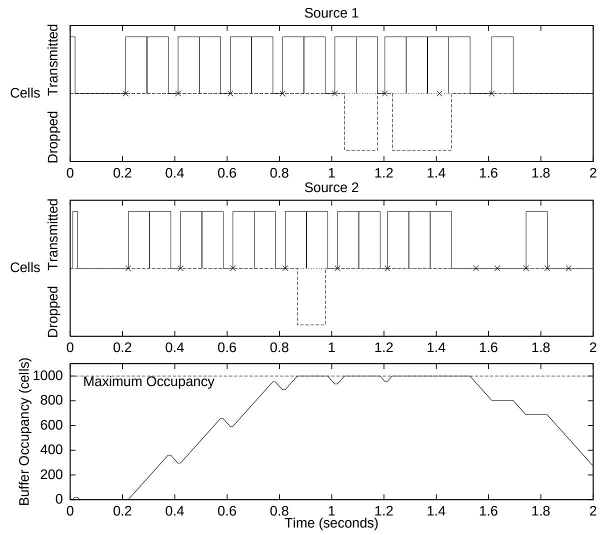

Figure 5: Dynamics of two TCP sources

The results shown in Figure 5 indicate that cells are only dropped when the switch buffer is full and both sources are transmitting cells (as expected). However, it is interesting to note that when congestion occurs, cells are only dropped from a *single* source during the congestion period. This unfairness happens to each source, but in different congestion periods.

The explanation for this unfairness phenomenon lies in the switch model used in ATM-TN. When two cells arrive at the same time at an output port that has only one empty buffer remaining, then the first cell captures the buffer vacancy and the second cell is dropped. Since the output link is the same speed as both input links, and both sources transmit at the maximum input cell rate, one source continues to dominate for an entire segment period[4] due to the deterministic cell drop strategy in ATM-TN switches. When round trip times, packet loss and retransmission, or slow-start change the bias, the other source may dominate. This phenomenon is similar to the "traffic phase effects" described in [8], except at the cell level instead of the packet level.

Normally, this phasing effect can lead to large differences in the share of bandwidth each source receives [8]. However, in our experiments the bias alternated between sources throughout the simulation duration and each received roughly the same share of bandwidth, since TCP timeouts and round trip times sufficiently "randomized" packet transmission times. In addition, the result of dropping all cells from the same source functions as a primitive type of Partial Packet Discard [18] which actually *improves* the performance of TCP over ATM.

4.4 Experiment 3: Switch Buffer Size

A final experiment was conducted to measure the effect of switch buffer size on the aggregate throughput of the network. Ten sources were connected to the switch and the switch buffer size was varied from 100 to 15000 cells. Simulations were performed for MSS sizes of 512 bytes and 9140 bytes. A similar experiment appears in Romanow & Floyd [18], though their network is a much faster local area network, while ours is a slower, longer wide area network. Our network was chosen in order to avoid modifying the granularity of the TCP timers, as required in their experiment. Our results are shown in Figure 6.

As expected, throughput improves with increasing buffer size. More traffic bursts can be accommodated by larger buffers and fewer cells are dropped. This increase in performance comes at the expense of longer cell transmission delays as cells wait in the buffer for transmission over the congested output link. However, in the simulated network these delays are negligible compared with the delay associated with retransmitting a lost segment.

At all buffer sizes, effective throughput is higher with a smaller MSS. This result is intuitive; cells are less likely to be lost from small segments, and the network overhead involved in retransmitting a lost segment is also lower. Large segments with lost cells result in wasted bandwidth as the remainder of the segment is transmitted over the congested link.

Qualitatively, our results are similar to those in Romanow & Floyd [18]. Our effective throughput is higher for all switch buffer sizes. We attribute the difference to the switch model's deterministic cell drop strategy; Romanow & Floyd use a switch model which discards arriving cells with equal probability when the buffer is full.

[4] That is, if cell i of source 1 beats cell j of source 2 to the buffer, then cell $i+1$ of source 1 will beat cell $j+1$ of source 2, and so on. Source 2 will obtain a free buffer only when source 1 completes the transmission of its current segment.

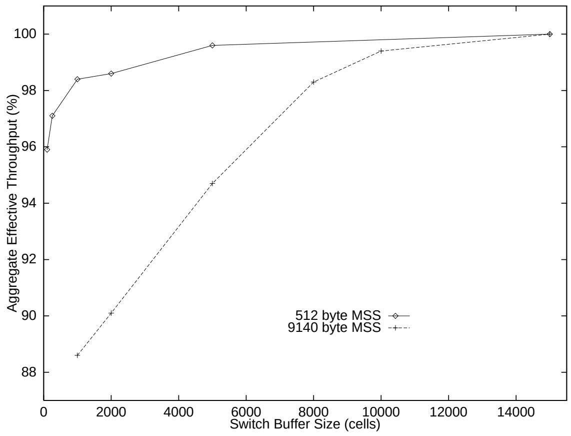

Figure 6: Effective Throughput vs Switch Buffer Size

5 Conclusions

We have designed and implemented a detailed simulation model of the TCP protocol over ATM networks. The model has been validated against known behavioural properties of TCP/ATM, using results from the literature and measurements from an ATM network. The model has been used in some preliminary experiments to assess the performance of TCP/ATM under various congested conditions.

The simulation experiments have served to illustrate several interesting issues in the performance of TCP over ATM networks. First, the aggregate throughput of several TCP sources multiplexed on a common output link was found to decrease when larger TCP segment sizes were used. Second, TCP/ATM traffic is subject to traffic phasing effects, both at the segment level and the cell level. Third, a deterministic cell drop strategy at the switches was found to improve throughput over a random cell drop method [18]. Finally, there is a tradeoff between increased throughput and the possibility of some sources receiving the majority of the available bandwidth while others receive very little. While this segregation effect is undesirable, the aggregate throughput remains higher than with the random cell drop strategy.

While several solutions to TCP/ATM performance problems are suggested in the literature, clearly more work is required in order to understand and improve the performance of TCP over ATM. Solutions to the performance problems may require changes to ATM switches within the network, revised implementations of TCP, or specialized TCP protocol support at the edges of an ATM network. Because our TCP model is detailed enough to simulate many of the performance problems identified by other researchers, it can also be used to evaluate potential solutions to TCP/ATM performance problems. Thus, the TCP model provides a valuable tool for networking researchers involved in the design and analysis of high speed ATM networks.

Acknowledgements

Financial support for this research was provided by the TeleSim project, and by the Natural Sciences and Engineering Research Council (NSERC) of Canada through research grant OGP0121969 and the NSERC Summer Undergraduate Research Assistant program.

The authors are grateful to the TeleSim project team members for their encouragement of and contribution to our ATM traffic modeling efforts, and for the availability of the ATM-TN network simulator and the Wnet experimental ATM network. Tim Harrison conducted several of the Wnet TCP/ATM experiments used in the early validation of our simulation model.

The presentation of this paper has benefited from the comments provided by Dr. Herbert Bauer at Newbridge Networks, and by the anonymous referees.

References

[1] ANSI, "AAL5 - A New High Speed Data Transfer AAL", ANSI T1S1.5 91-449, November 1991.

[2] M. Arlitt, Y. Chen, R. Gurski and C. Williamson, "Traffic Modeling in the ATM-TN Telesim Project: Design, Implementation, and Performance Evaluation", *Proceedings of the 1995 Summer Computer Simulation Conference*, Ottawa, Ontario, pp. 847-851, July 1995.

[3] R. Atkinson, "Default IP MTU for use over ATM AAL5", RFC 1626, 5 pages, May 1994.

[4] A. Bianco, "Performance of the TCP Protocol over ATM Networks", *Proceedings of the 3rd International Conference on Computer Communications and Networks*, San Francisco, CA, pp. 170-177, September 1994.

[5] D. Comer, *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture*, Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[6] D. Comer and J. Lin, "TCP Buffering and Performance over an ATM Network", Technical Report CSD-TR 94-026, Purdue University, West Lafayette, Indiana, 15 pages, March 1994.

[7] J. Crowcroft, I. Wakeman, Z. Wang, and D. Sirovica, "Is Layering Harmful?", *IEEE Network*, Vol. 6, No. 1, pp. 20-24, January 1992.

[8] S. Floyd and V. Jacobson, "On Traffic Phase Effects in Packet-Switched Gateways", *Internetworking: Research and Experience*, Vol. 3, No. 3, pp. 115-156, September 1992.

[9] F. Gomes, S. Franks, B. Unger, Z. Xiao, J. Cleary, and A. Covington, "SimKit: A High Performance Logical Process Simulation Class Library in C++", *Proceedings of the 1995 Winter Simulation Conference*, Arlington, VA, December 1995.

[10] M. Hassan, "Impact of Cell Loss on the Efficiency of TCP/IP over ATM", *Proceedings of the 3rd International Conference on Computer Communications and Networks*, San Francisco, CA, pp. 165-169, September 1994.

[11] V. Jacobson, "Congestion Avoidance and Control", *Proceedings of the 1988 ACM SIGCOMM Conference*, Stanford, CA, pp. 314-329, August 1988.

[12] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance", RFC 1323, 37 pages, May 1992.

[13] S. Leffler, M. McKusick, M. Karels, and J. Quarterman, *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley, Reading, Massachusetts, 1989.

[14] S. Minzer, "Broadband ISDN and Asynchronous Transfer Mode (ATM)", *IEEE Communications*, Vol. 27, No. 9, pp. 17-24, September 1989.

[15] K. Moldeklev and P. Gunningberg, "How a Large ATM MTU Causes Deadlocks in TCP Data Transfers", *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, pp. 409-422, August 1995.

[16] J. Nagle, "Congestion Control in IP/TCP Internetworks", *ACM Computer Communication Review*, Vol. 14, No. 4, pp. 11-17, October 1984. (Nagle's algorithm is also described in RFC 896, January 1984).

[17] J. Postel, editor, "Transmission Control Protocol", RFC 793, 85 pages, September 1981.

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

[19] W. Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley, Reading, Massachusetts, 1993.

[20] A. Tanenbaum, *Computer Networks*, Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[21] 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*, Vol. 13, No. 8, pp. 1457-1464, October 1995.

[22] 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", *Proceedings of the 1995 Winter Simulation Conference*, Arlington, VA, December 1995.

[23] G. Wright and W. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*. Addison-Wesley, Reading, Massachusetts, 1995.

[24] L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic", *Proceedings of the 1991 ACM SIGCOMM Conference*, Zürich, Switzerland, pp. 133-147, September 1991.