RICE UNIVERSITY

Analysis of TCP Performance over ATM Networks

by

Mohit Aron

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Master of Science

Approved, Thesis Committee:

Dr. Peter Druschel, Chairman
Assistant Professor
Computer Science

Dr. Willy Zwaenepoel
Professor
Electrical and Computer Engineering

Dr. Devika Subramanian
Associate Professor
Computer Science

Houston, Texas

December, 1997

ABSTRACT

Analysis of TCP Performance over ATM Networks

by

Mohit Aron

ATM technology is expected to gain widespread use both in wide-area networks and in high-speed local area networks. The performance of the TCP/IP protocol suite over such networks is of great importance, as it is widely used in the Internet and in private internetworks. However, numerous studies have shown that the effective throughput of conventional TCP implementations suffers over plain ATM networks.

This thesis presents a detailed study of the interactions between TCP's congestion control mechanisms and ATM networks. Based on the results of this study, we propose an enhanced version of TCP Vegas (called new-Vegas) and show that it achieves an increase in throughput of 40-70% over TCP Lite on plain ATM networks and up to 20% on EPD-enhanced ATM networks and packet networks. Moreover, our results indicate that the enhanced TCP Vegas is largely insensitive to EPD/PPD. In fact, even on plain ATM networks, it performs within 7% of its best throughput.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Peter Druschel, my advisor and the committee chairman, for his invaluable guidance and technical advice throughout the course of this research work. I thank the other members of my committee, Dr. Willy Zwaenepoel and Dr. Devika Subramanian, for their comments and suggestions. I also thank Dejan Mircevski for his initial help in implementing the ATM protocol in our network simulator and for the initial idea of testing the performance of TCP Vegas on ATM networks.

Contents

Illustrations

Chapter 1

Introduction

In this thesis, we present a detailed simulation based study of the interactions between TCP's congestion and error control mechanisms and ATM networks. Based on our analysis, we propose an implementation of TCP that yields high performance over ATM networks.

1.1 Motivation for this Thesis

ATM (Asynchronous Transfer Mode) [13] technology is expected to gain widespread use both in wide-area networks (WANs) and in high-speed local area networks (LANs). It holds the promise of delivering an order of magnitude greater bandwidth to both the LAN as well as the WAN environments. It supports several service classes, some of which provide resource guarantees in order to meet the real-time constraints of continuous media. New applications like real-time audio and video can thus be supported by ATM networks.

The UBR (unspecified bit rate) service class in ATM provides "best-effort" service to applications that do not require tightly constrained delay variations and quality of service. The conventional datagram networks (or packet networks) in the Internet and private internetworks today only provide this "best-effort" service. The TCP/IP protocol suite is widely used on these networks for reliable delivery of data by applications like WWW, ftp, telnet, rlogin etc. It is undesirable to change these applications with the advent of new network technologies. The performance of the TCP/IP protocol suite over the UBR service in ATM is thus of great importance because of its wide usage.

However, numerous studies have shown that the effective throughput of conventional implementations of TCP (based on U.C. Berkeley's Net/3 release) suffers when run over plain ATM networks*, particularly under conditions of network congestion

*By "plain" ATM network we refer here to an ATM network with best effort (UBR) service, and no AAL-specific switch-level enhancements.

[33, 14]. ATM networks primarily differ from packet networks in their fragmentation of IP datagrams into fixed sized ATM cells. The network dynamics resulting from this fragmentation have imposed new challenges on the congestion control schemes incorporated in TCP.

Various switch-level enhancements are supported by commercial ATM switches now to improve TCP performance over ATM networks. This thesis aims at studying the dynamics of TCP over ATM networks and enhancing TCP's congestion control mechanisms in order to yield improved performance that is insensitive to the fragmentation of IP datagrams, thus making it perform better over ATM networks irrespective of switch-level enhancements.

1.2 Contributions

In this thesis, we have done a detailed simulation-based study of the dynamics and performance of TCP over ATM networks with UBR service. The main results presented in this thesis as a result of our study can be summarized as follows:

- We analyze the actual causes of performance degradation of conventional TCP implementations over ATM networks.
- Our analysis gives a better understanding of the effectiveness of switch-level ATM enhancements in improving the performance of conventional TCP implementations.
- We propose a TCP implementation called TCP new-Vegas that affords better performance than conventional TCP implementations over ATM networks with or without switch-level enhancements.
- Our proposed TCP new-Vegas implementation is largely insensitive to switch-level ATM enhancements.
- We show that TCP new-Vegas performs nearly as well on ATM networks as on corresponding packet networks.
- Our TCP new-Vegas implementation shows that it is possible to have fragmentation tolerant congestion control techniques in TCP without knowing a priori the type of network that its traffic will traverse.

1.3 Organization

The remainder of this thesis is organized as follows. Chapter 2 gives a brief introduction to the TCP implementations and their congestion control mechanisms that are considered in this thesis. We also briefly introduce the switch-level enhancements that are supported in ATM networks to improve TCP performance. Chapter 3 describes our simulation environment. Chapter 4 discusses the effects of ATM and packet networks on TCP. Chapter 5 presents extensive simulation results showing the performance of TCP new-Vegas on ATM networks. In Chapter 6 we motivate and analyze each of the enhancements in TCP new-Vegas over TCP Lite, and their impact on TCP performance in ATM and packet networks. In Chapter 7 we discuss related work and we conclude in Chapter 8.

Chapter 2

Background

This chapter gives a brief introduction to TCP and introduces the congestion control schemes incorporated in two implementations of TCP – TCP 4.4BSD-Lite (or simply TCP Lite) [35, 36] and TCP Vegas [9, 11]. It also introduces the switch-level enhancements that have been proposed for ATM networks to improve TCP performance.

2.1 Overview of TCP

TCP (Transmission Control Protocol) is a connection-oriented protocol that provides reliable byte stream service to the applications that use it. It operates above the IP protocol in the protocol stack. Data is communicated with a peer using variable sized packets known as TCP segments. All data segments carry sequence numbers that are acknowledged by the receiver. TCP segments are enclosed in IP datagrams when they are transmitted on a packet network. On an ATM network, TCP segments are enclosed in AAL5† frames that are segmented into ATM cells before transmission.

TCP implements reliable delivery by retransmitting lost segments. The loss of segments is detected by maintaining a timer that waits for the reception of an acknowledgment (ACK) after a segment is sent. If the acknowledgment doesn't arrive in time, a retransmission timeout occurs and the segment is retransmitted. The BSD based implementations of TCP also detect and retransmit lost segments by a technique called *fast retransmit* that will be discussed in the next section.

Flow control and congestion control are two other features provided by TCP. Using flow control, a receiving TCP only allows the other end to send as much data as the receiver can buffer; thus preventing a fast sender from overwhelming a slow receiver with too much data. Congestion control prevents a TCP sender from trying to occupy more network resources than are available.

†The AAL5 ATM adaptation layer sits between IP and ATM in the host protocol stack and is responsible for segmenting IP packets into ATM cells at the sender and reassembling them back at the receiver.

2.2 Congestion Control in TCP Lite

TCP Lite is an extension of TCP Reno (4.3BSD Reno) and provides support for long fat pipes (high bandwidth-delay paths) amongst other improvements. The congestion control algorithms used in TCP Lite are essentially the same as those in TCP Reno and are based upon Van Jacobson's seminal work in [21, 22].

For the purpose of congestion control, a dynamically varying sliding-window is maintained at the TCP sender. This window is called the congestion window. It dictates the maximum amount of unacknowledged data that the sender can send. The congestion control algorithms in TCP Lite consist of three parts:

Slow Start : If the congestion window is smaller than a threshold (called *ssthresh*), the sender is said to be in the slow-start mode. In this mode, the sender increases its congestion window by one segment each time an acknowledgment is received. Thus, the congestion window is doubled every round-trip time (RTT). The goal of slow-start is to quickly fill the network "pipe" between the sender and the receiver.

Congestion Avoidance : If the congestion window is larger than *ssthresh*, the sender is said to be in the congestion avoidance mode. In this mode, the congestion window is increased by one segment every RTT. The goal is to increase the load on the network "cautiously" to probe any bandwidth that might become available.

Fast Retransmit and Fast Recovery : While a retransmission timer is maintained to detect lost segments, the retransmission timeout can take a long time to fire (on the order of seconds) because the timeout interval is set very conservatively. Another mechanism called *fast retransmit* allows earlier detection of lost segments. Upon receiving three duplicate ACKs (ACKs sent by the receiver upon receiving out of order segments), the sender decides that the segment immediately after the one acknowledged by the duplicate ACKs is lost. The *ssthresh* is set to half the value of congestion window, the lost segment is retransmitted and the *fast recovery* mechanism is invoked.

The fast recovery mechanism adjusts the congestion window in such a manner so that the number of new segments that are transmitted after a retransmission are half the size of the congestion window before the loss was detected. This avoids stalling the sender for a complete RTT while recovering a lost packet.

The fast recovery terminates when an ACK that acknowledges the lost segment is received. After fast recovery terminates, the congestion window's value is set to the value indicated by *ssthresh* (which now is half the value of the congestion window when the lost segment was detected).

The version of TCP Lite used in the simulations reported in this thesis corresponds to Lite.4 proposed in [10]. This version fixes many bugs in the original 4.4BSD-Lite distribution.

2.3 Congestion Control in TCP Vegas

TCP Vegas is an implementation of TCP proposed in [9, 11] and further evaluated in [1]. The original TCP Vegas implementation, as distributed by the University of Arizona, incorporates three distinct enhancements[‡] over the congestion control schemes in TCP Reno:

Modified Slow-Start: It features a modified slow-start procedure (Vegas-SS) that allows a doubling of the congestion window only every other RTT. Vegas-SS makes a decision to move into the congestion avoidance state based on a comparison of the actual sending rate and a predicted sending rate. This usually avoids the heavy losses that are accompanied by doubling the window every RTT and moving to congestion avoidance state only upon detecting losses.

Retransmission Mechanism: Secondly, TCP Vegas incorporates a new retransmission mechanism (Vegas-RM) that detects packet losses earlier than Reno and Lite. On receiving the first and second non-duplicate ACK after a retransmission, it also retransmits any segment that hasn't been acknowledged for a time larger than the timeout period. This is likely to catch any other segments that are lost prior to the first retransmission without having to wait for three duplicate ACKs again. Finally, TCP Vegas does not decrease its congestion window multiple times for losses that occur in the same RTT, and it decreases it congestion window by 3/4 and not by 1/2 as in TCP Lite and Reno.

[‡]The Vegas distribution also contains a bug that prevents the congestion window from increasing beyond 64K on receiving duplicate ACKs while doing fast recovery. Versions used in this thesis had the bug fixed.

Congestion Avoidance: Thirdly, TCP Vegas uses a congestion avoidance procedure (Vegas-CA) that is pro-active, and tries to detect the incipient stages of congestion before losses occur. It does so by additively increasing and decreasing its congestion window according to the difference between the actual sending rate and the predicted sending rate. This is in sharp contrast to TCP Lite and Reno, which periodically create losses to probe the available capacity of the network.

2.4 Switch-level ATM Enhancements

Numerous studies [33, 14] have observed a degradation in TCP performance when run over plain ATM networks. The primary cause of the loss of TCP throughput over ATM is thought to be the transmission of useless ATM cells from partially dropped segments [33].

To improve TCP performance over ATM, commercial switches now implement *Partial Packet Discard* (PPD) and *Early Packet Discard* (EPD) [33]. PPD involves dropping all subsequent ATM cells from an AAL5 frame once one has been dropped. EPD involves dropping entire AAL5 frames prior to switch buffer overflow. The switch drops the frame whenever the proportion of the switch buffer in use exceeds a fixed threshold (known as EPD threshold). This prevents useless cells from corrupted segments from wasting network bandwidth and occupying network resources.

To attempt a fair distribution of buffer space and/or bandwidth among the virtual circuits, some commercial switches also provide an enhancement called per-VC queuing. Per-VC queuing involves queuing cells from different virtual circuits in separate queues in a switch. This prevents congestion conditions in a virtual circuit from affecting other virtual circuits.

Chapter 3

Simulation Environment

This chapter describes the simulation environment used to produce the results presented in the following chapters. We first describe the simulator used. We then describe the network topologies simulated for producing the results reported in this thesis. This is followed by a description of the various network parameters and algorithms used in our simulations. Finally we describe how to read the various graphs shown in this thesis.

3.1 The Simulator

The simulations were done using the x-sim network simulator [12], which is based on the x-kernel [20]. x-sim is an execution-driven network simulator, where the actions of network protocols are simulated by executing its actual protocol implementation code, rather than an abstract behavioral model of the protocol. Host and router computation is assumed to have zero overhead. The simulator clock has a granularity of $1\mu s$.

The original x-sim, as distributed by the University of Arizona, did not include support for ATM. We extended the simulator by implementing the ATM and AAL5 protocols and adding support for them in the simulator. We also implemented switch algorithms for simulating ATM switches in the simulator.

3.2 Network Topologies

Two network topologies were used in our simulations, both for ATM and packet networks. The same link bandwidths and propagation delays were used for comparing packet and ATM networks. In case of the packet network, the switches used simulate IP routers, while for ATM they simulate ATM switches.

The first and simpler network topology used is shown in Figure 3.1. Eight senders (S1, S2, ... S8) use TCP to send data to eight destinations (D1, D2, ... D8) across

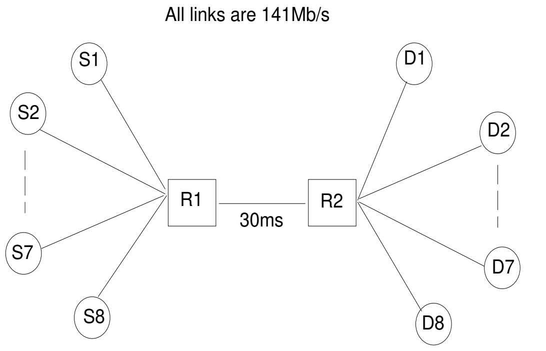

Figure 3.1 Topology 1

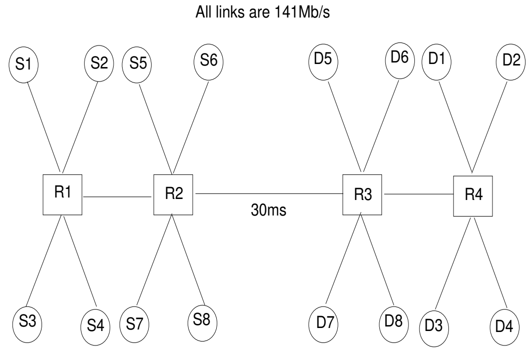

Figure 3.2 Topology 2

a bottleneck link. The bottleneck link connects two switches R1 and R2 where R1 is the bottleneck switch. The link bandwidth was chosen to be 141.33 Mbps so as to make the transmission of an ATM cell (53 bytes) an integral number of microseconds ($3\mu s$). The round-trip propagation delay is 60ms. This topology resembles the ones used in other studies on TCP dynamics [33, 32] but incorporates a high-delay link typical of wide-area networks.

The second topology is shown in Figure 3.2. It has two more switches (R3 and R4) than the first topology. Four senders each are connected to switches R1 and R2. Four destinations each are connected to R3 and R4. This topology differs from the former in providing two bottleneck points (switches R1 and R2). It more closely resembles a real internet where different hosts on a LAN (local area network) transfer data to different destinations across a WAN (wide area network).

Unless stated otherwise, in the simulation results reported in this thesis, all senders transfer 60MB of data to the corresponding destinations and the simulation time was limited to 10s (enough to transfer about 175MB of data across the bottleneck link). The simulation results reported are averages from five runs, each with a different stagger time (explained in Section 3.3.3) and seed for the simulator. The seed affects the initial times after which the fast and the slow TCP timers fire, thus introducing randomness in the simulations.

3.3 Network Characteristics

3.3.1 TCP Parameters

All the simulations use a TCP advertised window size of 512KB to accommodate the relatively large bandwidth-delay products in our simulated networks. Each TCP connection uses a separate ATM virtual circuit (VC). The default MTU for IP over ATM is 9180 bytes [4]. However, the TCP segment size in all the simulations presented in this paper is 9168 bytes. This size was chosen to be an exact multiple of the ATM payload (48 bytes). This minimizes the padding done at the AAL5 layer when a segment is broken into cells, thus enabling a more effective comparison between packet-TCP and TCP over ATM. Padding only occurs when the packets have a size less than 48 bytes (e.g. ACK packets) and possibly for the last segment sent when the total amount of data sent is not an integral multiple of the MTU. The TCP segment size over packet networks used in the simulations is also 9168 bytes so as to make a fair comparison of packet-TCP with TCP over ATM.

3.3.2 Switch Algorithms

The switches used in the simulations involving packet and plain ATM networks are FCFS[§] switches with output buffering and the drop-tail[¶] discipline for dropping packets/cells. In simulations involving enhanced ATM switches, the Early Packet Discard and Partial Packet Discard mechanisms were used according to the suggestions made in [33].

For simulations with EPD and per-VC queuing in the ATM switches, we used the following algorithm. Cells are queued separately in a FIFO fashion for each VC. The first cell from each non-empty VC queue associated with an outgoing link is transmitted in a round-robin manner. Our implementation allows full buffer sharing amongst the various VC queues. Let B be the switch buffer capacity, T the EPD threshold, AVC the number of VCs with non-empty queues, O the buffer occupancy of the switch, and o_i be the occupancy of VC i's queue. When the first cell of a new AAL5 frame arrives on VC i, the frame is dropped iff $O > T*B$ and $o_i > T*B/AVC$. The first condition ($O > T*B$) constrains frames to be dropped only when the overall switch buffer occupancy exceeds the EPD threshold (as in an EPD implementation without per-VC queuing). The second condition ($o_i > T*B/AVC$) implies that frames are only dropped when VC i's queue has become larger than the EPD threshold times its fair share of switch buffers. This prevents frames from getting dropped from those VC's that have short queues even when the overall switch buffer occupancy exceeds the EPD threshold.

[§]This means that ATM cells are transmitted by the switch in the order in which they were received.

[¶]The drop-tail discipline entails discarding the last cell received upon a switch buffer overflow.

3.3.3 Fairness

When all senders start simultaneously sending data to the respective destinations, we noticed segregation effects [17, 18, 33] for low switch buffer sizes. This caused some of the connections to get most of the bandwidth causing the other connections to starve leading to extreme unfairness.

We staggered the startup times of the senders by a few milliseconds to allow their congestion windows to grow for some time before experiencing congestion. In particular, stagger times of 6ms, 8ms, 10ms, 12ms and 14ms were used for the results reported in this paper. This staggering eliminates most of the segregation effects and all results reported in this paper have a fairness of at least 0.79 for topology 1 and 0.70 for topology 2, as measured by Jain's Fairness index [23]. The vast majority of results involving modest to high switch/router buffer sizes have a fairness of above 0.95.

3.4 Graph Description

There are two kinds of graphs that are presented in this thesis.

3.4.1 Effective Throughput

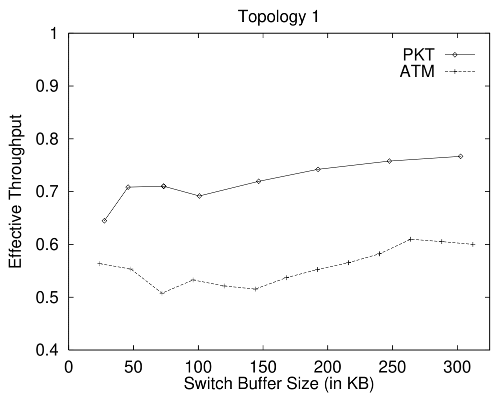

Figure 3.3 Throughput vs switch-buffer size

Figure 3.3 shows an example of a plot between effective throughput vs switch buffer size. The network topology used is indicated on the top of the graph. For ATM networks, the x-axis refers to the *effective* switch buffer size. The effective switch buffer size for ATM does not include the space needed for the cell headers. The effective throughput is shown as a ratio of the total amount of useful data that was transferred, to the maximum amount of useful data that could have been transferred over the bottleneck link for the time of the simulation. Defining effective throughput in this manner makes it possible to compare directly the corresponding graphs for ATM and packet networks without making adjustments for packet header overheads.

3.4.2 TCP Traces

Figures 3.4 and 3.5 explain the TCP trace graphs used in this paper. Figure 3.4 shows some general information shown in the graphs:

1. Hash marks on the x-axis indicate when an ACK was received.
2. Hash marks at the top of the graph indicate when a segment was sent.
3. The numbers on the top of the graph indicate when the n^{th} kilobyte (KB) was sent.
4. Diamonds on top of the graph indicate when TCP checked whether a coarse-grained timeout should happen.
5. Black circles on top of the graph indicate that a coarse-grained timeout actually occurred.
6. Solid vertical lines running the whole height of the graph indicate when a segment that is retransmitted was actually sent.

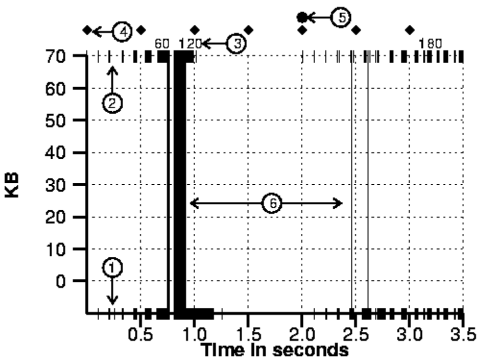

Figure 3.4 General Elements

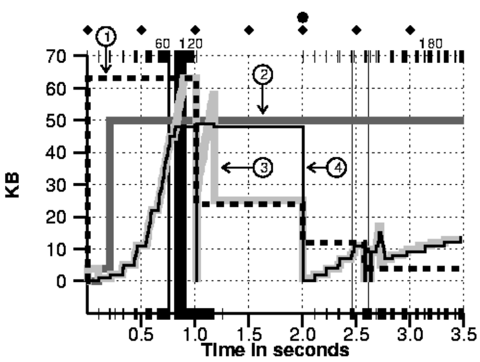

Figure 3.5 TCP Windows

Figure 3.5 shows traces of the TCP windows:

1. The dashed line gives the slow-start threshold (ssthresh).
2. The dark gray line gives the send window (minimum of the sender's buffer size and receiver's advertised window) and gives the upper limit on the amount of unacked data.
3. The light gray line gives the congestion window.
4. The thin-line gives the actual amount of unacked data.

Chapter 4

Dynamics of TCP over ATM

This chapter discusses key differences between ATM and packet networks, how these differences interact with TCP dynamics, and the manner in which they affect TCP performance. Many of these observations have been made elsewhere in the literature; we recount them here for completeness. We also present simulation results comparing the performance of TCP Lite over ATM and packet networks.

4.1 Impact of ATM on TCP Performance

ATM networks affect TCP in two different ways – firstly because of the intrinsic differences between ATM and packet networks, and secondly because ATM networks are typically high bandwidth networks.

4.1.1 Differences between ATM and Packet Networks

Discard Unit and Retransmission Unit: A key difference between ATM and packet networks is that in ATM the switches' discard unit (a cell) typically differs from the end host's retransmission unit (e.g., a TCP segment). When TCP/IP is run over ATM, a TCP segment is fragmented at the AAL5 layer into 48 byte ATM cells. A congested plain ATM switch drops data at the granularity of ATM cells. When even a single cell from a TCP segment is dropped, the segment has to be retransmitted in its entirety. In a packet network, on the other hand, the retransmission unit and the discard unit are identical. This mismatch between ATM and TCP/IP can lead to several undesirable effects on TCP performance:

Bandwidth and resource wastage: The remaining cells from partially dropped segments occupy network resources and waste link bandwidth, only to be discarded at the destination [33]. This is likely to cause more congestion, since dropping n cells leads to the eventual retransmission of m cells, for $n \leq m \leq 2 * n * MTU/48$ (in the worst case, each cell drop could corrupt two segments and each segment has MTU/48 cells).

Multiple segment loss: A conventional ATM switch does not know about segment boundaries. Hence in general, dropping a segment's worth of data at an ATM switch might lead to the loss of cells from several TCP segments. In particular, two consecutive segments from a TCP connection are often lost when a switch drops only one segment's worth of data [30].

Given these phenomena, one can expect that under otherwise identical conditions, a TCP connection over an ATM network is likely to loose more segments, and it is more likely to suffer consecutive segment losses than a TCP connection run over packet switched network. To counter these effects, many commercial switches now implement Early Packet Discard (EPD), which involves dropping entire AAL5 frames prior to switch buffer overflow. Some commercial switches also provide per-VC queuing, which attempts a fair distribution of buffer space and/or bandwidth among the virtual circuits using a given link. In particular, per-VC queuing can prevent multiple connections from being affected when switch buffers overflow.

4.1.2 Additional ATM Characteristics

There are some additional characteristics of ATM that have an impact on TCP performance. These characteristics are not specific to ATM; instead, they are related to the fact that ATM networks are typically high-bandwidth networks.

High Bandwidth: ATM is considered a gigabit technology and ATM networks are typically high bandwidth networks. TCP implementations use coarse-grained clocks to estimate round-trip delay (RTT) and schedule retransmission timeouts (RTOs). Consequently, RTOs tend to exceed the actual RTT by a large amount. The throughput loss resulting from the associated period of inactivity preceding a timeout is proportional to the bandwidth of the network.

Large Bandwidth-Delay Product: Wide-area, high-bandwidth ATM networks can have high bandwidth-delay products. During its initial slow-start, TCP doubles its congestion window every RTT until losses are incurred or the congestion window reaches the receiver's advertised window. When losses occur, it can be expected that a substantial fraction of the current congestion window is lost. Since the size of that congestion window is related to the network's bandwidth-delay product, it follows that the amount of data lost—and the degree of congestion caused in the network—during the initial slow-start increases with the network's bandwidth-delay product.

Large MTU: In typical use, IP over ATM uses a packet size of 9180 bytes compared to about 1500 bytes for Ethernet networks. The MTU determines TCP's maximal segment size, which in turn determines the unit and rate of TCP's congestion window growth. In particular, it affects the length of traffic bursts generated by a TCP sender (TCP can send up to 2 maximal sized segments per receiver acknowledgment during slow-start; under certain conditions it can even send more [10]). A larger MTU therefore requires more buffering at the switches to absorb traffic bursts. In addition, a larger MTU may aggravate the problem of ACK compression [38].

4.2 Performance of TCP Lite

In this section, we present comparative performance results for TCP Lite over ATM and packet networks. Our results show that the effective throughput of TCP Lite suffers by as much as 30% when run over plain ATM networks.

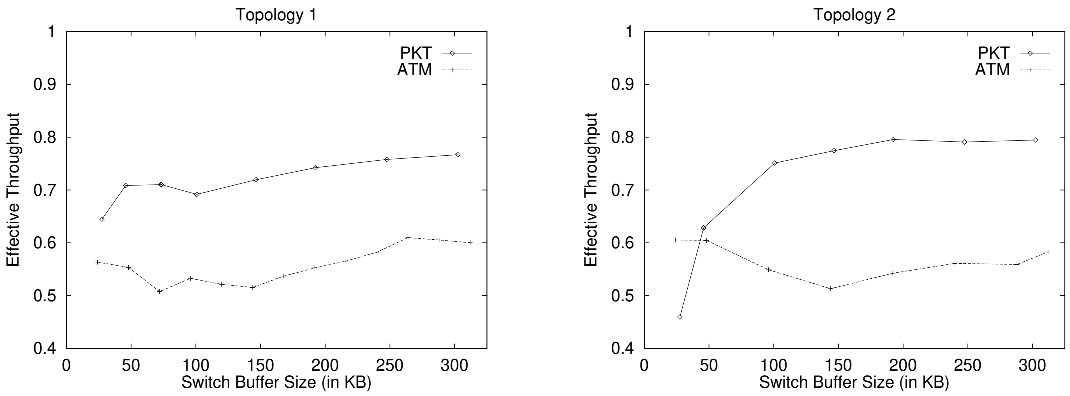

Figure 4.1 TCP Lite over ATM and packet network

Figure 4.1 compares the effective throughput of the TCP senders as a function of switch buffer size on both network topologies. As mentioned in Section 3.2, all senders transfer 60MB of data to the corresponding destinations while the simulation time was limited to 10s. It is clear from the plots that the effective throughput of TCP Lite over ATM can be 10-30% lower than that over packet networks.

Earlier studies have suggested that the primary cause of TCP throughput loss over ATM are the transmission of useless cells from partially dropped segments [33].

Our results show that this effect can only account for a fraction of the observed throughput loss. Consider the following. In the simulations shown in Figure 4.1, the average amount of retransmitted data never exceeds 560KB. This gives an upper bound on the number of segments that were retransmitted due to cell loss. In the worst case, only one cell from each of these segments is dropped, in which case almost all of the 560KB will occupy link bandwidth and waste resources. As the simulation time was limited to 10s, the bandwidth occupied by these dead cells does not exceed 56KB/s per sender. However, a bandwidth loss of 56KB/s only accounts for about 3% loss in throughput of each sender, whereas we observe a 10-30% loss in throughput. In the next chapters, we will analyze the causes of TCP throughput loss over ATM.

Chapter 5

Performance Evaluation

In this chapter, we present comparative performance results for TCP Lite and new-Vegas over ATM and packet networks. We have already shown in Section 4.2 that the effective throughput of TCP Lite suffers by as much as 30% when run over plain ATM networks. We first show that our proposed new-Vegas performs on plain ATM networks equally well as on packet networks for modest to large, and within 10% for small switch buffer sizes. Second, when compared to TCP Lite, the proposed new-Vegas achieves between 40-70% higher effective throughput on plain ATM networks, and approximately 20% higher effective throughput on ATM networks with EPD and optimal setting of the EPD threshold. On packet networks, new-Vegas achieves up to 20% higher effective throughput than TCP Lite, and up to 10% higher throughput than TCP Vegas. new-Vegas is relatively insensitive to EPD and PPD in the ATM switches and performs within 7% of its best performance even over plain ATM. Finally, we show that new-Vegas also performs well for short transfer sizes.

As noted in Section 3.3, all simulation results presented in this thesis use a TCP segment size of 9168 bytes. To test the sensitivity of our results to the segment size, we performed simulations with segment sizes of 4352 bytes and 1500 bytes (segments of these sizes may commonly appear on ATM networks that are part of a heterogeneous internetwork). The effective throughput of TCP connections is affected by the segment size in two ways. First, at smaller segment sizes, the mismatch between TCP's retransmission unit and plain ATM's discard unit is less severe, causing a tendency towards better throughput with smaller segment sizes. This effect appears to dominate the results reported in [33], which were obtained in a simulated ATM LAN environment. Second, the segment size determines the unit and rate of congestion window growth in all versions of TCP. On networks with substantial bandwidth-delay products (such as our simulated networks), small segment sizes cause prolonged slow-start phases, resulting in reduced effective throughput. Small segment sizes also hurt performance in the congestion avoidance mode because TCP takes a long time to fill up the pipe (as congestion window is increased by one segment every RTT). In our simulations, the second effect dominates the first, resulting in reduced throughput with smaller segment sizes for all versions of TCP and on both ATM and packet networks.

5.1 Performance over plain ATM networks

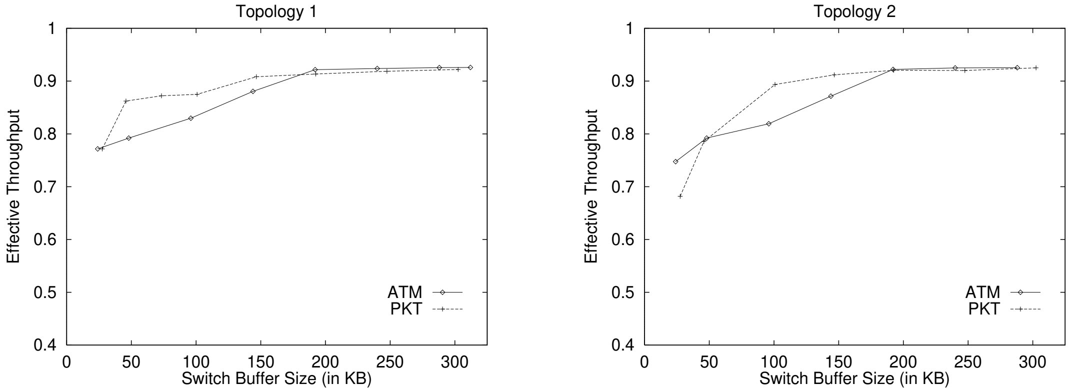

Figure 5.1 new-Vegas over ATM and Packet Network

Figure 5.1 shows the effective throughput of our enhanced version of Vegas (new-Vegas) as a function of switch buffer size for plain ATM and packet networks, and on the two topologies shown in Figures 3.1 and 3.2. The results clearly show that unlike TCP Lite, new-Vegas performs almost equally well on plain ATM and on packet network across a wide range of switch buffer sizes, and on both topologies.

5.2 Performance over enhanced ATM networks

Figures 5.2 and 5.3 show the performance of TCP Lite and new-Vegas, respectively, over packet networks and ATM networks with and without switch-level enhancements. Simulations were done using no enhancements, using PPD, and using EPD in the ATM switches. Three different EPD thresholds were considered—0.50, 0.75 and 0.90. The curve denoted by 'ATM-plain' shows the results when no switch-level enhancements were used.

The results in Figure 5.2 show that with EPD and an appropriate setting of the threshold, the performance of TCP Lite over ATM becomes comparable to the performance over packet networks. PPD only leads to a modest improvement of TCP

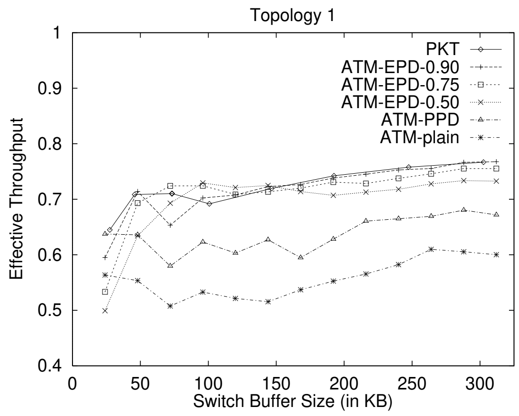

Figure 5.2 Lite

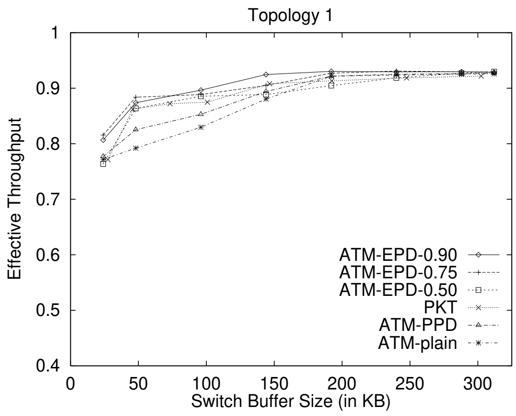

Figure 5.3 new-Vegas

Lite's effective throughput. The performance of TCP Lite is thus very sensitive to the presence of PPD and EPD, and moderately sensitive to the threshold setting used in EPD. This confirms earlier results reported in [33].

Results in Figure 5.3 indicate that the best effective throughput obtained with new-Vegas (new-Vegas on ATM with EPD-0.90) exceeds the best performance of TCP Lite (TCP Lite on packet network) by approximately 20% over the range of switch buffer sizes. When compared to TCP Lite, the proposed new-Vegas achieves between 40-70% higher effective throughput on plain ATM networks, and approximately 20% higher effective throughput on ATM networks with EPD and optimal setting of the EPD threshold. On packet networks, new-Vegas achieves up to 20% higher effective throughput than TCP Lite, and up to 10% higher throughput than TCP Vegas. Moreover, new-Vegas over plain ATM still affords significantly better performance than TCP Lite on either the packet network, or ATM with EPD at its best threshold setting.

Secondly, the closely clustered curves in Figure 5.3 indicate that new-Vegas is relatively insensitive to switch-level enhancements. In particular, new-Vegas with EPD never performs more than 7% better than new-Vegas over plain ATM. This is significant, since an optimal setting of the EPD threshold may be difficult to obtain in practice [33, 24], and some deployed ATM networks do not even support EPD/PPD.

We have already shown in Section 4.2 that the bandwidth wastage possible from the useless cells in corrupted segments is insignificant as compared to the performance degradation actually observed for TCP Lite over ATM. Our simulation results indicate that the main effectiveness of EPD arises from the prevention of switch buffer occupancy by cells from corrupted segments. In particular, this results in the prevention of consecutive segment losses. By dropping more cells than are necessary, enough buffer space is available for subsequent segments. So in effect, EPD causes TCP to experience single segment losses, and TCP is well specialized in recovering from those.

To investigate the impact of per-VC queuing in ATM switches, we performed simulations with support for both EPD and per-VC queuing in the switches, using the algorithm described in Section 3. The results show that the addition of per-VC queuing to an EPD switch does not have a significant impact on the effective throughput of any of the TCP versions we studied. However, per-VC queuing did show a noticeable increase in fairness among the TCP connections, particularly at small switch buffer sizes. In all simulations with per-VC queuing, the fairness index was above 0.96 with new-Vegas and above 0.92 with TCP Lite.

5.3 Short Transfers

This section presents simulation results showing the performance of new-Vegas versus Lite on TCP connections with short transfer sizes. The widespread use of the World Wide Web, combined with the fact that HTTP transfers sizes currently average about 13 KBytes [3] makes short transfers sizes an important case to consider.

Figure 5.4 shows the variation of effective throughput with transfer size. The simulations were done using the topology shown in Figure 3.1, with a switch buffer

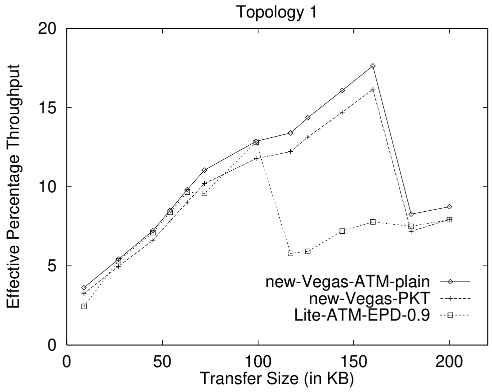

Figure 5.4 Short Transfers

size of 192.5 KB. The effective throughput achieved with new-Vegas is comparable to that achieved with TCP Lite on an ATM network with EPD 0.90 for transfer sizes up to 100 KB and beyond 180 KB. Between 100 KB and 180 KB, new-Vegas performs significantly better that TCP Lite. Also, new-Vegas over plain ATM performs slightly better than on the equivalent packet network over the entire range of transfer sizes shown.

At small transfer sizes, the effective throughput of TCP is dominated by the slow-start phase [29]. TCP Lite starts its slow-start phase with a congestion window size of one segment, and doubles its congestion window every round-trip time until it either reaches the receiver's advertised window size or packet losses occur. Like the original TCP Vegas, new-Vegas starts with a congestion window size of two segments and doubles its congestion window every other round-trip time until it detects signs of impending congestion in the network. Our results indicate that Vegas's larger initial window size and smarter slow-start termination compensate for the slower window growth during slow-start, resulting in competitive throughput for short transfers.

Chapter 6

Analysis of TCP Enhancements

In this chapter, we present a detailed study of all enhancements over TCP Lite that are embodied in TCP new-Vegas. In particular, we discuss how these techniques address the issues raised by ATM networks and separately quantify their impact on TCP throughput over ATM and packet networks. To fully expose the interactions between ATM networks and TCP, the detailed ATM simulations presented in this section were performed on plain ATM networks.

The original TCP Vegas implementation incorporates three distinct enhancements – Vegas-SS, Vegas-RM and Vegas-CA as mentioned earlier in Section 2.3. Our proposed version of TCP Vegas (new-Vegas) enhances the Vegas-RM fast retransmission mechanism. The enhanced mechanism (new-Vegas-RM) allows recovery from multiple segment losses in the same congestion window, and it eliminates stalls in the transmission pipeline caused by consecutive packet losses. All of the TCP enhancements considered in this paper are discussed in detail in the following subsections.

Observe that Vegas-RM and new-Vegas-RM deal with the situation when congestion occurs and segments are lost. In particular, they aim at avoiding timeouts and stalls in the transmission pipeline as a result of segment losses. Vegas-SS and Vegas-CA, on the other hand, aim at avoiding congestion in the first place. Thus, they reduce the number of situations where losses occur, and they reduce the amount of data lost in situations when losses cannot be avoided.

The following sections present a detailed discussion of the techniques that were used in new-Vegas and their interaction with ATM networks. The purpose of this study is to analyze the contribution of the various enhancements embodied in new-Vegas, their relative importance, and their individual and combined impact on the performance of TCP over ATM and packet networks.

6.1 Handling Multiple Segment Losses

We begin with the issue of consecutive TCP segment losses. Consecutive losses commonly arise from congestion conditions in plain ATM networks, and they have a profound impact on the performance of TCP Lite. This section motivates and evaluates the fast retransmission mechanisms in TCP Vegas and new-Vegas.

As mentioned earlier, cell losses in plain ATM switches may span segment boundaries, resulting in the loss of two TCP segments as opposed to one in a packet network [30]. While TCP Lite can effectively recover from the loss of a single segment in one RTT using its fast retransmit mechanism [35], it is not well equipped to handle the loss of two or more segments in a single RTT [19, 16]. First, the fast retransmit mechanism can only detect lost segments if the congestion window is large enough to supply duplicate ACKs for each lost segment. If the congestion window is too small, a costly timeout is necessary to recover from the losses. A second reason is that Lite (and Reno) halve the congestion window for each lost segment detected by duplicate ACKs. As a result, after the fast retransmission of the 2nd lost segment, no further segments can be sent until an ACK for the lost segments arrives (also noted in [15]). This causes a stall in TCP's transmission pipeline, temporarily under-utilizing the available network capacity.

Finally, consider the case where a timeout occurs after some of the losses were detected by fast retransmit, causing the congestion window to be halved multiple times. As a result of the multiple congestion window reductions, TCP will terminate the slow-start following the timeout potentially too early, preventing it from quickly reaching the congestion window size necessary to fully utilize the network.

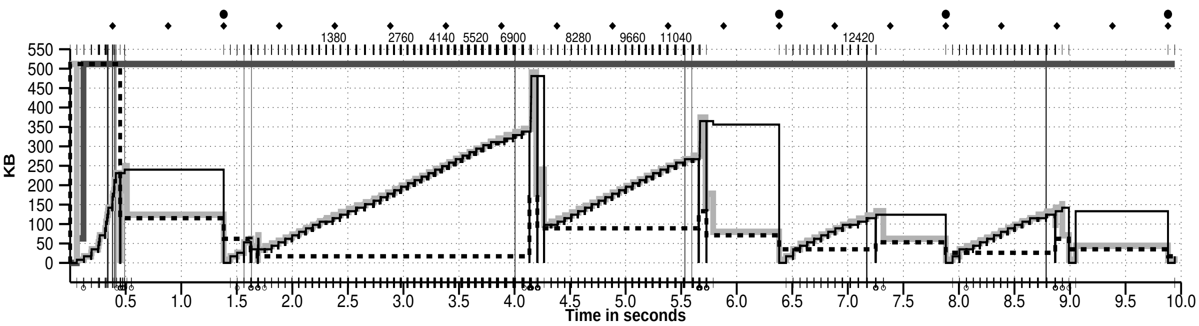

Figure 6.1 TCP Lite over ATM

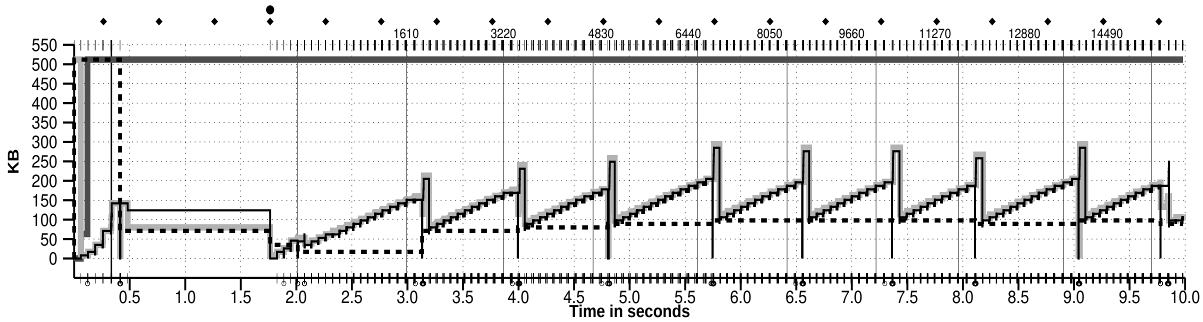

Figure 6.2 TCP Lite over packet network

Figure 6.1 shows the window traces of a typical TCP Lite transfer over plain ATM[‖]. Figure 6.2 shows a corresponding trace over the packet network. The traces depict a transfer with the standard TCP coarse-grained timers. On ATM, a large number of retransmission timeouts (near 1.5, 6.5, 8 and 10 seconds) can be seen as compared to the non-ATM network. In general, we have observed 4-5 times more retransmission timeouts in simulations over plain ATM networks than over packet networks.

Figure 6.1 illustrates the effect of losses in an ATM network on TCP Lite's windows. At around 4.2s, two segments were lost (only one vertical line can be seen as the segments were transmitted within a few microseconds of each other). The congestion window however was large enough to recover from both the losses. However, as a result of the recovery, the congestion window is halved twice, once for each segment that was lost. Near 5.6s, 3 segments were lost, which resulted in a retransmission timeout. In this case, the congestion window was halved twice when the first two losses were detected. Then Lite's slow-start threshold (ssthresh) was set to half of this value. As a result, slow-start ends at 1/8th the value of the congestion window when losses were first detected. The timeout near 7.9s was caused by two consecutive segment losses near 7.3s. The congestion window was not large enough to recover from both losses in this case. The timeout near 9.8s was caused by 3 segment losses near 8.8s.

Similar problems can arise during the slow-start mode of TCP Lite. Here, the problem is even more acute as TCP Lite doubles its congestion window every RTT. This increases the possibility of more than two consecutive segment losses when congestion occurs, rendering the fast retransmit mechanism ineffective. Hence in this

[‖]A description of these traces was given in Section 3.4.2.

case there is an even greater chance that TCP Lite suffers a retransmission timeout. The timeout in Figure 6.1 near 1.5s is an example of this phenomenon.

The above discussion suggests that a TCP implementation that can recover more effectively from multiple segment losses during a single RTT interval can have improved performance over plain ATM networks. TCP Vegas incorporates a retransmission mechanism (Vegas-RM) that is more effective in handling multiple segment losses within one RTT. Upon arrival of the first or second non-duplicate ACKs after a fast retransmission, it uses timestamps to determine if more segments have been unacknowledged for a period exceeding the calculated timeout interval. If so, it retransmits the segments in question immediately. Furthermore, TCP Vegas decreases its congestion window at most once while recovering from losses in a single RTT. The Vegas-RM mechanism proves to be effective in handling increased rates of multiple segment losses characteristic of an ATM network.

We next introduce two enhancements to the TCP Vegas retransmission mechanism that form the basis of our proposed new-Vegas. The first modification to Vegas is primarily beneficial to ATM networks, while the second modification is beneficial to both packet and ATM networks.

6.1.1 Enhanced new-Vegas Retransmission Mechanism

Figure 6.1 and the previous analysis show that consecutive segment losses are very common in plain ATM networks. We have shown that the Vegas-RM retransmission mechanism can more effectively handle multiple segment losses in one RTT than Lite. In particular, on receiving the first and second non-duplicate ACK after a retransmission, Vegas-RM retransmits any segment that hasn't been acknowledged for a time larger than the timeout period (estimated using the relatively fine-grained system clock).

However, consecutive losses in Vegas stall the transmission pipeline for one RTT, causing a temporary underutilization of the network. This is because Vegas deflates its congestion window after receiving the first non-duplicate ACK, irrespective of whether any further segments need to be retransmitted. As a result, the amount of unacknowledged data in the network decreases only by one segment, while the congestion window decreases to 3/4 of its value before the first loss was detected. Thus, the congestion window gets smaller than the unacknowledged data, preventing the transmission of any new segments. Retransmissions of any further segments are not followed by any new transmissions until the retransmitted segments are acknowledged.

The first enhancement in new-Vegas involves deflating the congestion window only when no other segments need to be retransmitted after receiving a non-duplicate ACK following a retransmission. This keeps the transmission pipeline flowing for the entire round trip time until the acknowledgment for the retransmitted segment is received. Further, the new segments transmitted during every such round trip time never amount to more than 3/4 of the congestion window just before the first loss was detected. This ensures that new-Vegas does not become more aggressive after the first round trip time following the detection of the first loss.

The second enhancement in new-Vegas focuses on the number of packets that can be recovered during a single fast recovery phase. Vegas can only recover a maximum of two more packets beyond the first one lost during a single round-trip time. We have modified Vegas's retransmission mechanism so that it can recover all packets in the congestion window that were sent prior to the invocation of the fast retransmit mechanism. This is similar to a suggestion made by Hoe in [19].

6.2 Congestion Avoidance

We have shown that new-Vegas-RM can effectively deal with consecutive segment losses in ATM networks, and that the addition of this mechanism to TCP can significantly improve the performance of TCP over ATM networks. In fact, we'll show in the next section that new-Vegas-RM achieves 1-5% higher throughput on plain ATM than TCP Lite achieves over a packet network. In this section, we analyze the effect on performance of using Vegas congestion avoidance (Vegas-CA and Vegas-SS).

TCP Lite's congestion control scheme reduces the size of the congestion window only in response to segment losses. Furthermore, to probe the available capacity of the network, TCP Lite periodically increases its congestion window until congestion occurs, as indicated by losses. As a result, the variation of TCP Lite's congestion window exhibits a typical saw-tooth pattern as can be seen in Figures 6.1 and 6.2. TCP Vegas aims at eliminating these self-induced losses using its congestion avoidance mechanisms (Vegas-CA and Vegas-SS). We'll quantify the impact of these mechanisms in Section 6.3.

6.3 Performance Impact of new-Vegas Enhancements

In this section we quantify the impact of all the enhancements in TCP new-Vegas. As mentioned earlier, all simulations presented in this chapter are performed on plain ATM networks to study the interactions between ATM networks and TCP.

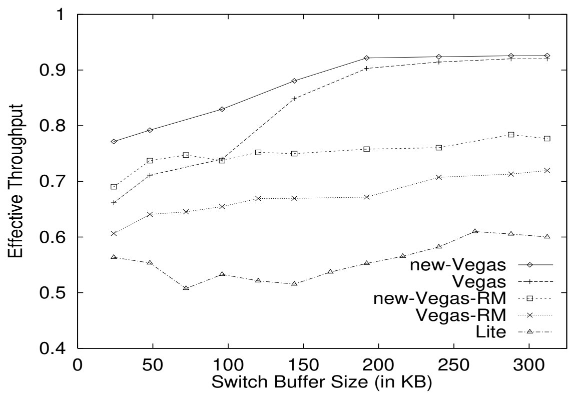

Figure 6.3
TCP over ATM

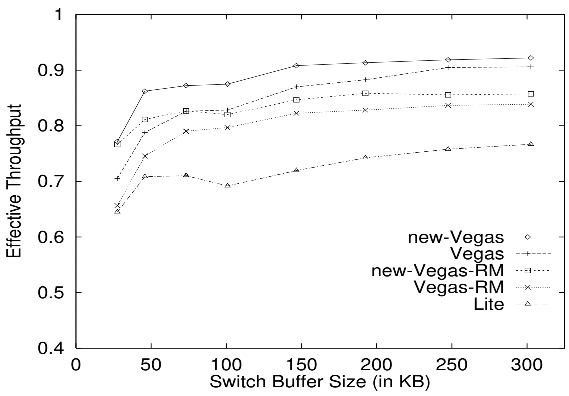

Figure 6.4
TCP over PKT

Figures 6.3 and 6.4 show the throughput results obtained in simulations involving various versions of TCP, both on ATM and packet networks. The network topology used is the one depicted in Figure 3.1. Observe that the effective throughput achieved with new-Vegas surpasses or matches that of all other versions of TCP on ATM networks, across all switch buffer sizes. Moreover, new-Vegas also surpasses the throughput of Vegas on packet networks by 2–10%, which in turn achieves between 10–20% higher throughput than TCP Lite on packet networks.

To evaluate the impact of Vegas-RM on the throughput over ATM networks, we performed simulations with a version of TCP Vegas that has Vegas-SS and Vegas-CA disabled, leaving only the Vegas-RM enhancement active. The plots for Vegas-RM in Figures 6.3 and 6.4 show the effective throughput with this mechanism. Note that the Vegas-RM mechanism alone does indeed afford a significant performance advantage over Lite on *both* ATM and packet networks. On ATM networks, Vegas-RM alone achieves approximately two thirds of the throughput gains that new-Vegas yields. Note also that Vegas-RM alone accounts for a large fraction of the throughput gains that Vegas obtains over Lite on packet networks.

Figures 6.3 and 6.4 show the effective throughput of new-Vegas-RM over ATM and packet networks. Like Vegas-RM, new-Vegas-RM incorporates only the improved retransmission mechanism, but not the Vegas slow-start and congestion control mechanisms. In the case of ATM, there is a noticeable throughput increase with new-Vegas-RM (about 10%) over Vegas-RM. This improvement reflects the avoidance of pipeline stalls associated with consecutive segment losses. Figure 6.4 shows a minor improvement (less than 3%) afforded by new-Vegas-RM over Vegas-RM over the packet network. This can be attributed to the elimination of an initial retransmission timeout by new-Vegas-RM as a result of the second modification. This timeout (occurs in Figure 6.2 at 1.7s) is usually the result of an unchecked increase of the congestion window during connection startup in the absence of slow-start congestion avoidance (Vegas-SS). It is to be noted that new-Vegas-RM achieves 1-5% higher throughput on plain ATM than TCP Lite achieves over the packet network.

The performance of the congestion avoidance mechanisms (Vegas-CA and Vegas-SS) over ATM and packet networks is reflected in the difference between the curves for new-Vegas versus new-Vegas-RM, and Vegas versus Vegas-RM, respectively, in Figures 6.3 and 6.4. For both Vegas and new-Vegas on plain ATM, the congestion control mechanisms afford approximately 10% throughput improvement for low switch buffer sizes (below 100 KB), increasing to 30% and 20%, respectively, for switch buffer sizes of 200KB and above. The same trend can be seen on packet networks, although the improvements are less pronounced.

The Vegas congestion avoidance mechanisms avoid losses by monitoring the difference between the expected sending rate and actual sending rate [9]. This difference is maintained so as to occupy only a small, fixed amount of buffer space per TCP connection in the network. TCP Lite, on the other hand, tends to keep the buffers in the bottleneck switch filled, independent of the size of these buffer. With Vegas, increasing the switch buffer size under otherwise identical conditions therefore increases the available buffer space in switches, improving the network's ability to absorb traffic bursts and reducing losses due to congestion. This explains why Vegas and new-Vegas are able to translate larger switch buffer sizes into improved throughput.

When the available buffer space in the bottleneck switch/router is below the fixed amount Vegas is trying to use for a connection, losses occur, thus increasing the importance of efficient handling of retransmissions. This explains why the enhanced retransmission mechanism in new-Vegas has the most pronounced effect on throughput for small switch buffer sizes. This effect is amplified on plain ATM networks, where congestion leads to an increased number of segment losses.

6.4 Fine-grained Clock for Timeouts

This section considers the issue of timer granularity in TCP. TCP Lite performs round-trip time (RTT) estimation and schedules retransmission timeouts (RTOs) at a granularity of 500 msecs. It is widely recognized that these coarse-grained timers pose problems for TCP on high-bandwidth networks due to the large discrepancy between the actual RTT and the scheduled RTO [10, 19]. In this section, we explore the extent to which performance can be improved using a finer granularity for the timers.

We have performed simulations with TCP implementations that schedule timeouts at a granularity of 10ms. The performance of Vegas and new-Vegas on both ATM and packet networks are not significantly affected by the addition of fine-grained timeouts. This can be explained by the decreased reliance on timeouts to handle segment losses. With TCP Vegas and new-Vegas, fine-grained timers do not appear to be necessary to achieve good throughput on wide-area ATM networks.

The addition of fine-grained timers does however improve the throughput of TCP Lite over ATM. Figure 6.5 shows the effect of increasing the granularity of TCP Lite's timeouts to 10ms. However, even with fine-grained timers, Lite's throughput over ATM remains about 15% below that achieved on packet networks. This suggests that improving timer granularity alone is not sufficient to make TCP Lite perform

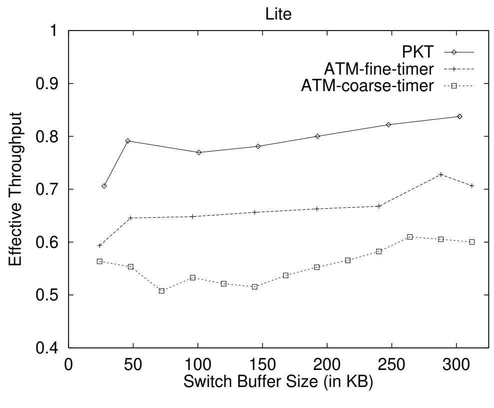

Figure 6.5 TCP Lite with 10ms timer

as well over ATM as on packet networks. The work in [37] discusses the intrinsic limitations of timers in improving performance.

Chapter 7

Related Work

The related work falls into three main categories:

- Techniques both at the protocol as well as at the switch level to improve TCP performance over ATM networks in particular.
- Techniques to improve TCP performance over other unconventional network technologies like wireless, satellite etc.
- Techniques to improve TCP so as to improve its performance in general irrespective of the kind of network on which it is used.

This thesis attempts to improve TCP congestion control algorithms so as to improve its performance over ATM networks.

7.1 ATM

Floyd and Romanow propose and evaluate the PPD and EPD switch enhancements to improve TCP performance over ATM networks with UBR service [33]. The ATM ABR (Available Bit Rate) service was designed as an alternative to UBR for bursty data traffic. The ABR service provides the sender with explicit information about the bit rate currently available to a connection and guarantees a zero cell loss rate provided that the sender adjusts its bit rate correspondingly. To effectively operate TCP over an ABR connection, the window-based TCP congestion control mechanism must be modified to honor the rate-based notifications provided by the ABR service. Simulation results in [25] suggest that running TCP over ABR achieves good throughput when buffer space in the network is sufficient. A more recent study suggests that for TCP workloads, the performance improvements may not justify the greater complexity of ABR [31].

Bestavros and Kim [8] propose a version of TCP that uses partial segments it receives as a result of cell loss on ATM networks. The implementation must be parameterized for the underlying network type, requiring the application to know a priori what type of network its traffic will traverse. Furthermore, a change to the AAL5 layer is required to deliver partial frames to TCP. Our proposed new-Vegas does not make any assumptions about the underlying network, and requires no changes to other protocols layers.

7.2 Other Emerging Network Technologies

The emergence of other new network technologies has also affected TCP performance. Several research efforts have been made in improving TCP performance in such environments.

Balakrishnan et al [7, 6] have observed the degradation in TCP performance over wireless networks. Packet losses due to bit corruptions on wireless networks is misinterpreted by TCP as a loss due to congestion, leading to an unnecessary back-off that degrades performance. Several mechanisms for improving TCP performance over wireless networks have been proposed and a comparative study of the performance afforded by these techniques has been made.

In another paper, Balakrishnan et al [5] have studied the effects of network asymmetry on TCP performance. Network asymmetry can arise due to mismatch in bandwidth, latency, packet error rate or media-access in the two directions of data transfer. Ack congestion control, ack filtering and TCP sender adaptation are some techniques that have been suggested to improve performance.

Allman et al [2] have studied the performance of TCP over satellite links. They observe that slow-start and congestion avoidance techniques have a limiting effect on TCP performance due to the large times taken to fill the high bandwidth-delay paths typically present in satellite links.

7.3 TCP Congestion Control

Van Jacobson's seminal work in [21] defines the congestion avoidance and control schemes that were used in 4.3BSD-Tahoe TCP (TCP Tahoe). His work in [22] describes the fast recovery scheme incorporated in 4.3BSD-Reno TCP (TCP Reno). We have described these schemes in Section 2.2.

Shenker and Zhang [34] have made a simulation based study of the behavior of the TCP Tahoe implementation. They have observed that packets from a single host from individual one-way connections tend to be clustered together. They have also observed that every connection loses a single packet during each congestion epoch. This study applies to the simulations involving TCP Lite over packet networks that are reported in this thesis. However, the dynamics over plain ATM networks are different as congestion conditions tend to cause multiple segment losses. The work in [38] extends the study of TCP dynamics over packet networks to two-way traffic. Two new observations – ACK compression and out-of-phase queue synchronization are made. Mogul [28] shows how to observe and measure some of these phenomena in real networks.

Brakmo and Peterson [9] have proposed TCP Vegas which incorporates new techniques (described in Section 2.3) for detecting and controlling congestion. Our proposed TCP new-Vegas is an extension of TCP Vegas that provides improved techniques for handling multiple segment losses. In another paper, Brakmo and Peterson [10] have also pointed out various performance problems with the TCP 4.4BSD-Lite implementation and propose fixes to improve throughput. The TCP implementations used for simulation results reported in this thesis incorporate the fixes in the Lite.4 version proposed in that paper. These fixes remove problems in the implementation of header prediction and estimating the retransmit timeouts.

Hoe [19] proposes two methods to improve TCP's congestion control algorithms. First, it attempts to set the slow-start threshold (ssthresh) to an appropriate value by measuring the bandwidth-delay product using a variant of the packet-pair technique [26]. A similar attempt was also made in the context of TCP-Vegas [11]. Queuing delays, ACK compression [38] and ACK clustering [34] can affect the estimation of the available bandwidth. Vegas and new-Vegas achieve slow-start termination differently, by detecting the flattening of the sending rate. As a result, new-Vegas is not dependent on the initial ssthresh value. The second technique proposed by Hoe improves TCP's retransmission mechanism to deal effectively with multiple segment losses in the same congestion window. This method has partly inspired our improved retransmission mechanism in new-Vegas.

Floyd [16] discusses the problem of invoking fast retransmit mechanism multiple times for the same window of data. The retransmission mechanism in new-Vegas addresses this problem. Fall and Floyd [15] investigate the effect of multiple packet losses on the congestion control algorithms of TCP-Reno. They point out that the absence of selective acknowledgments imposes limits on TCP's performance. Their work also shows that TCP with selective acknowledgments (SACK-TCP) [27] can effectively recover from multiple packet losses. While new-Vegas requires changes only at the sender side, SACK-TCP requires making changes to existing implementations both at the sender as well as the receiver side. Furthermore, new-Vegas obtains additional performance benefits from its congestion control mechanism.

In the present Internet, the majority of bulk data transfers occur from an information server (e.g. WWW server) to a client host. The number of client hosts far exceeds the number of information servers. Since TCP new-Vegas requires only sender-side support, the deployment of new-Vegas by these servers can yield improved performance in a cost effective manner. TCP-SACK, on the other hand, requires deployment at both client hosts and information servers to be effective. Finally, as new-Vegas is relatively insensitive to enhancements in ATM switches, good performance can be achieved whether or not the networks used incorporate EPD/PPD.

Chapter 8

Conclusions and Future Work

In this chapter, we summarize the main results from this thesis and discuss possibilities for future work.

8.1 Main Results

This thesis presents a detailed simulation-based study of the dynamics and performance of TCP over ATM networks with UBR service. Three versions of TCP are evaluated, TCP Lite, TCP Vegas, and new-Vegas, our proposed enhanced version of Vegas. We summarize the main results as follows.

- TCP Lite suffers a loss of effective throughput of up to 30% when run over a plain ATM network, as compared to a packet network. This loss can be prevented almost entirely through the use of Early Packet Discard (EPD) in the ATM switches, assuming an appropriate setting of the EPD threshold.

- The primary reason for this throughput loss is TCP Lite's inefficiency in handling multiple segments losses, which occur frequently in a congested plain ATM network. Our simulations show that the transmission of remaining cells from partly dropped AAL5 frames—previously cited as the main cause of throughput loss—accounts for only a small fraction of lost throughput.

- When compared to its performance on packet networks, TCP Vegas suffers a throughput loss of no more then 10% over plain ATM networks, and only at small switch buffer sizes. Analysis shows that Vegas's advantage over Lite results from its improved handling of multiple segments losses and its improved congestion control mechanism, which avoids self-induced congestion and makes more effective use of available buffer space in the switches.

- Based on this observation, we propose new-Vegas, a version of TCP Vegas with a further enhanced retransmission mechanism. When compared to TCP Lite, the proposed new-Vegas achieves between 40-70% higher effective throughput on plain ATM networks, and approximately 20% higher effective throughput on ATM networks with EPD. On packet networks, new-Vegas achieves up to 20% higher effective throughput than TCP Lite, and up to 10% higher throughput than TCP Vegas.

- Both TCP Vegas and new-Vegas are relatively insensitive to the presence of EPD or PPD in the ATM switches, and the EPD threshold setting. Even on plain ATM networks, new-Vegas performs within 7% of its best effective throughput. The addition of per-VC queuing to an EPD switch does not affect the throughput of TCP, but leads to increased fairness.

- The main effectiveness of EPD arises from the prevention of switch buffer occupancy by cells from corrupted segments. This is instrumental in preventing multiple segment losses.

8.2 Future Work

We have proposed enhancements to TCP based on our study of a limited number of network topologies. While we believe that the results are much broader in their scope, it would still be interesting to experiment with more complicated network topologies as well as performing the experiments on real networks.

The proposed enhancements to TCP resulted from some of the challenges posed by emerging high bandwidth network technologies such as ATM. Future Internet applications and technologies would put additional demands on transport protocols. Our long-term research aims towards studying the design changes required in existing protocols as well as new protocols that will be required for the stability and scalability of the Internet.

Bibliography

[1] J. Ahn, Peter Danzig, Zhen Liu, and Elliot Yan. Evaluation of TCP Vegas: Emulation and Experiment. In *Proceedings of the ACM SIGCOMM '95 Symposium*, 1995.

[2] Mark Allman, Chris Hayes, Hans Kruse, and Shawn Ostermann. TCP Performance over Satellite Links. In *Proceedings of 5th International Conference on Telecommunication Systems*, March 1997.

[3] Martin F. Arlitt and Carey L. Williamson. Web Server Workload Characterization: The Search for Invariants. In *Proceedings of the ACM SIGMETRICS '96 Conference*, Philadelphia, PA, April 1996.

[4] R. Atkinson. Default IP MTU for use over ATM AAL5. *IETF Internet Draft*, February 1994. Available by anonymous ftp from ds.internic.net:internet-drafts/internet-drafts/draft-ietf-atm-mtu-07.txt.

[5] Hari Balakrishnan, Venkata N. Padmanabhan, and Randy H. Katz. The Effects of Asymmetry on TCP Performance. In *Proceedings of 3rd ACM Conference on Mobile Computing and Networking*, September 1997.

[6] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A Comparison of Mechanisms for improving TCP Performance over Wireless Links. In *Proceedings of the ACM SIGCOMM '96 Symposium*, August 1996.

[7] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz. Improving TCP/IP Performance over Wireless Networks. In *Proceedings of 1st ACM Conference on Mobile Computing and Networking*, November 1995.

[8] Azer Bestavros and Gitae Kim. TCP Boston: A Fragmentation-tolerant TCP Protocol for ATM Networks. In *Proceedings of IEEE INFOCOM'97*, April 1997.

[9] Lawrence Brakmo, Sean W. O' Malley, and Larry Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of the ACM SIGCOMM '94 Symposium*, pages 24–35, 1994.

[10] Lawrence Brakmo and Larry Peterson. Performance Problems in 4.4BSD TCP. *ACM Computer Communication Review*, 25(5):69–86, October 1995.

[11] Lawrence Brakmo and Larry Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, October 1995.

[12] Lawrence S. Brakmo and Larry L. Peterson. Experiences with Network Simulation. In *Proceedings of the ACM SIGMETRICS '96 Conference*, 1996.

[13] M. DePrycker. *Asynchronous Transfer Mode*. Ellis Horwood, London, 1993.

[14] B. J. Ewy, J. B. Evans, V. S. Frost, and G. J. Minden. TCP/ATM Experiences in the MAGIC Testbed. In *Proceedings of the Fourth IEEE International Symposium on High Performance Distributed Computing (HPDC-4)*, 1995.

[15] Kevin Fall and Sally Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Computer Communication Review*, 26(3), July 1996.

[16] Sally Floyd. TCP and Successive Fast Retransmits. Technical report, Lawrence Berkeley Laboratory, May 1995. ftp://ftp.ee.lbl.gov/papers/fastretrans.ps.

[17] Sally Floyd and Van Jacobson. Traffic Phase Effects in Packet-Switched Gateways. *Computer Communication Review*, 21(2), April 1991.

[18] Sally Floyd and Van Jacobson. On Traffic Phase Effects in Packet-Switched Gateways. *Internetworking: Research and Experience*, 3(3):115–156, September 1992.

[19] Janey C. Hoe. Improving the Start-up Behaviour of a Congestion Control Scheme for TCP. In *Proceedings of the ACM SIGCOMM '96 Symposium*, 1996.

[20] Norman C. Hutchinson and Larry L. Peterson. The x-kernel: An Architecture for Implementing Network Protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, January 1991.

[21] Van Jacobson. Congestion Avoidance and Control. In *Proceedings of the ACM SIGCOMM '88 Symposium*, pages 314–32, August 1988.

[22] Van Jacobson. Berkeley TCP evolution from 4.3-tahoe to 4.3-reno. In *Proceedings of the Eighteenth Internet Engineering Task Force*, August 1990.

[23] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modelling* . John Wiley & Sons, New York, 1991.

[24] L. Kalampoukas and A. Varma. Performance of TCP over Multi-Hop ATM Networks: A Comparative Study of ATM-Layer Congestion Control Schemes. In *Proceedings of the ICC '95 Conference*, June 1995.

[25] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, Fang Lu, and Saragur Srinidhi. Performance of TCP/IP over ABR Service on ATM Networks. *Proceedings of the IEEE GLOBECOM '96 Conference*, 1996.

[26] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Proceedings of the ACM SIGCOMM '91 Symposium*, pages 3–15, September 1991.

[27] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018: TCP selective acknowledgment options, October 1996.

[28] Jeffrey C. Mogul. Observing TCP Dynamics in Real Networks. In *Proceedings of the ACM SIGCOMM '92 Symposium*, pages 281–292, August 1993.

[29] Jeffrey C. Mogul. The Case for Persistent-Connection HTTP. In *Proceedings of the ACM SIGCOMM '95 Symposium*, 1995.

[30] R. Morris and H. T. Kung. Impact of ATM Switching and Flow Control on TCP Performance: Measurements on an Experimental Switch. *Proceedings of the IEEE GLOBECOM '95 Conference*, 1995.

[31] Teunis J. Ott and Neil Aggarwal. TCP over ATM: ABR or UBR? In *Proceedings of the ACM SIGMETRICS '97 Conference*, Seattle, WA, June 1997.

[32] Michael Perloff and Kurt Reiss. Improvements to TCP Performance in High-Speed ATM Networks. *Communications of the ACM*, 38(2):90–100, February 1995.

[33] A. Romanow and S. Floyd. The Dynamics of TCP Traffic over ATM Networks. *IEEE Journal on Selected Areas in Communication*, 13(4), May 1995.

[34] S. Shenker, L. Zhang, and D. Clark. Some Observations on the Dynamics of a Congestion Control Algorithm. *ACM Computer Communication Review*, 20(4):30–39, October 1990.

[35] W.R. Stevens. *TCP/IP Illustrated Volume 1 : The Protocols*. Addison-Wesley, Reading, MA, 1994.

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

[37] Lixia Zhang. Why TCP Timers Don't Work Well. In *Proceedings of the ACM SIGCOMM '86 Symposium*, August 1986.

[38] Lixia Zhang, Scott Shenker, and David D. Clark. Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic. In *Proceedings of the ACM SIGCOMM '91 Symposium*, pages 133–148, 1991.