IMPACT OF ATM SWITCHING AND FLOW CONTROL ON TCP PERFORMANCE: MEASUREMENTS ON AN EXPERIMENTAL SWITCH

R. Morris and H. T. Kung

The authors are with the Division of Applied Sciences
Harvard University, USA

Abstract -- Measurements on an experimental ATM switch in a local area network have demonstrated significant performance gains for TCP traffic with ATM-level flow control. Without flow control, buffer overruns at ATM switches feeding into bottlenecks can prevent TCP from using more than a few percent of the potential bandwidth. A detailed analysis of the cell loss patterns that foil TCP's loss recovery strategy is presented. With flow control, efficiency is nearly perfect. ATM flow control prevents cell loss due to congestion, and as a result TCP can avoid retransmit time-out delays and maintain a high transmission rate.

I. INTRODUCTION

A good deal of traffic over future ATM networks may well use existing transport protocols such as TCP/IP. TCP has its own window-based flow control mechanism which interprets lost or delayed packets as evidence of congestion. Switch-based ATM networks will be orders of magnitudes faster than most existing networks but many ATM switches are expected to have relatively limited buffer space; how well will TCP work in this situation? Can TCP still be reasonably efficient and fair? If not, how to solve the problem?

There have been many studies and papers [11, 13] addressing these questions. Some have noted potential performance degradation of TCP over ATM, but few have addressed the underlying causes.

This paper studies the impact of ATM switches and ATM-level flow control [2, 9] on TCP performance, and offers detailed explanation and analysis. These results are based on performance measurements obtained from an experimental ATM switch developed by Harvard and Bell-Northern Research (BNR).

II. CREDITNET ATM SWITCH

BNR and Harvard have jointly developed an experimental ATM switch called CreditNet [1], with both 622 megabits per second (OC-12) and 155-Mbps (OC-3) ports. Unique features of the switch include ATM-level credit-based flow control [7, 8, 9, 12], per-VC (per virtual circuit) queueing, round-robin VC scheduling, multicast support in hardware, highly programmable microprocessor-based switch port cards, and built-in instrumentation for performance measurement.

CreditNet's flow control mechanism reserves buffer space in each switch for each VC. As depicted in Fig. 1, the switches send "credits" upstream (towards the data sender) reflecting the amount of unused space in each VC's reserved buffer area. A switch or host will only send data on a VC if the VC has credit; that is, if the downstream switch or host along the VC's path has unused buffer space reserved for the VC. Flow control thus can avoid data loss due to congestion.

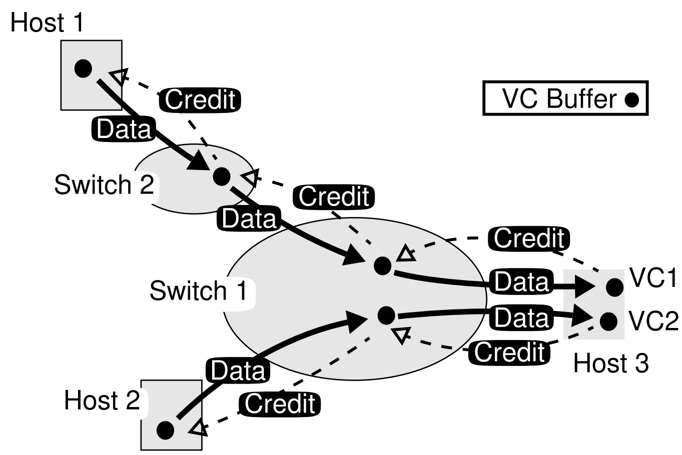

Fig. 1: Credit-based flow control applied to each link of a VC.

The CreditNet switch implements Partial Packet Discard as an option.This means that when the option is turned on, once the switch drops a cell from a packet, it will keep dropping cells until the end of the packet, whether or not it could buffer the cells. To allow the destination host to detect the start of the next packet, the switch does not drop the cell that marks the end of the current packet unless the switch is still out of memory. The Partial Packet Discard feature should be distinguished from Early Packet Discard [13], in which the switch decides whether to drop the entire packet when the first cell of the packet arrives. If the switch does not have roughly one packet's worth of space available, the entire packet is discarded.

As of the first quarter of 1995, five of these switches have been built. The switches have successfully interoperated, over SONET links, with several commercial or experimental ATM host adapters including those from DEC (for TurboChannel), Sun (S-Bus), Intel (PCI) and Zeitmet (PCI). Both the OC-3 and OC-12 links have been used in applications. In addition, a Q93B signalling system has been implemented on the switch.

One of the switches now operates on site at Harvard University.

III. EXPERIMENTAL CONFIGURATIONS

The experiments described below use two network configurations in a local area network environment. The first, shown in Fig. 2 (a), involves host A sending a continuous stream of data through the switch to host B. Host A's link to the switch runs at 155 megabits per second (Mbps), while host B's link runs at only 53, enforced by a properly programmed scheduler on the link input. This is one of the simplest configurations in which congestion occurs. Note that after SONET and ATM overhead, a 155-Mbps link can deliver roughly 134 Mbps or 17 megabytes per second of useful payload to a host. A 53-Mbps link can deliver about 5.7 megabytes per second.

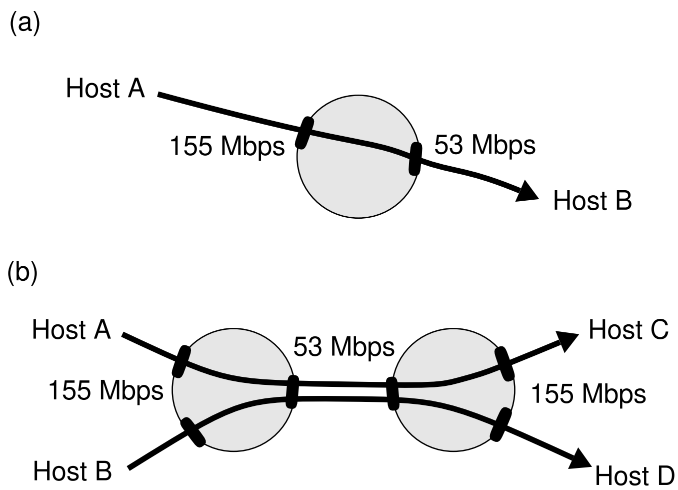

Fig. 2: (a) Network configuration for single TCP experiments on CreditNet; and (b) configuration for two competing TCPs. The circles are switches. Each darkened bar denotes a switch port.

The second configuration, shown in Fig. 2 (b), involves four hosts. Host A sends data to host C, and host B to host D. The four host links run at 155 Mbps, and the bottleneck link between the switches runs at 53 Mbps. The purpose of this configuration is to show how two conversations interact.

The hosts in all these experiments are DEC Alpha 3000/400 workstations running OSF/1 V3.0. The OSF/1 TCP implementation [4], used in all the experiments reported in this paper, is derived from 4.3-Reno [14]. This TCP tends to acknowledge, and thus transmit, pairs of packets. The TCP window size for these experiments is limited to no more than 64K bytes, and the packet size 9180 except when noted. The workstations use 155-Mbps OTTO TurboChannel adapters provided by DEC. The Alphas can send or receive TCP using the OTTOs at about 15 megabytes per second. The OTTO drivers optionally implement CreditNet's credit-based flow control partially in software; with credit turned on they can send and receive TCP at 13 megabytes per second.

The measurements are all directly derived from the instrumentation counters in the CreditNet switch hardware. The hardware keeps track of the total number of cells sent by each VC and the number of cells buffered for each VC. For all the experiments reported in this paper, Partial Packet Discard is not turned on. Some discussion on its impact is given in Section IV.

IV. TCP PERFORMANCE WITHOUT ATM FLOW CONTROL

One of the simplest situations in which TCP has trouble over ATM without ATM-level flow control involves a host with fast link to a switch sending to a host with a slower link. This configuration is depicted in Fig. 2 (a). Fig. 3 shows useful throughput or "goodput" for a range of switch buffer sizes and packet sizes. As noted above, the maximum obtainable bandwidth on the bottleneck link is 5.7 megabytes per second after SONET and ATM overhead. The switch buffer sizes account only for the memory used by the 48-byte payloads of ATM cells.

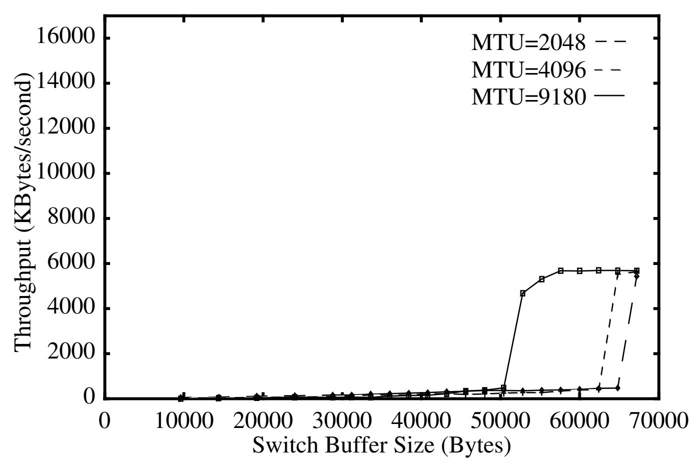

Fig. 3: Measured TCP performance with varying packet size (MTU) and switch buffer space, for the configuration of Fig. 2 (a). The dots indicate the points where experiments were run.

Regardless of packet size, TCP performs badly unless the amount of buffer space is close to an entire 64Kbyte window, the maximum amount of data TCP will send before pausing to wait for an acknowledgment. Performance is good at slightly less than 64K because a few packets are effectively stored in the hosts and adapter cards. The OTTO host adapter buffers one packet in on-board memory during both transmission and reception, and thus will buffer more bytes when packets are large. The fact that large packets have better performance than small in Fig. 3 is caused only by this storage of packets in the hosts and adapters, and not to any deeper advantage.

Figure 4 shows the TCP bandwidth achieved over time by a single connection with 9180-byte packets and 32Kbytes of switch buffer space. With much less than 64K of switch buffering, TCP sends small bursts of data separated by 1.5-second pauses. These pauses are due to retransmission time-outs caused when TCP's window [5] exceeds the switch's buffer space. This happens within a few tens of milliseconds after TCP starts to retransmit each time: TCP can send a window in less than 10 milliseconds, and the window increases by one packet per window sent. Thus after TCP sends a few windows of data, the switch drops some packets because the window is larger than the buffer space. Since TCP's minimum time-out is at least one second, TCP spends far more time waiting to retransmit than it does sending data. The inactivity period of more than 2 seconds in Fig. 4 is likely caused by the TCP exponential back-off triggered by the loss of some retransmitted packet.

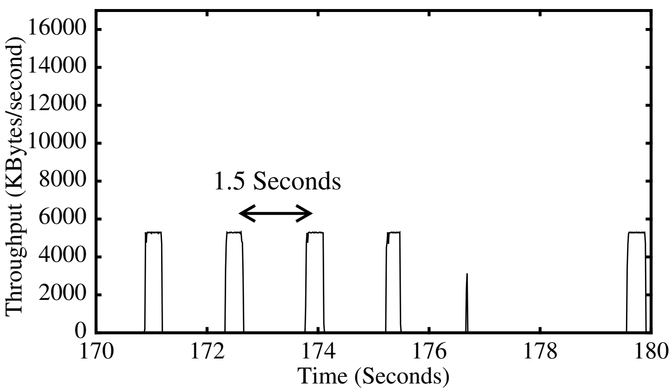

Fig. 4: Measured TCP bandwidth over time showing time-outs, for the configuration of Fig. 2 (a), with 64Kbyte TCP window and 32Kbyte switch buffer.

TCP running through packet switches, which queue and switch packets rather than cells, does not suffer from this problem. Each time TCP increases its window size to be one packet too large for the switch, the packet switch typically drops only one packet. TCP can efficiently detect and recover from a single lost packet with a mechanism called fast retransmit [14], which will decrease the window size and resend the lost packet with very little pause.

Why doesn't fast retransmit work over ATM? Fig. 5 plots switch buffer occupancy in bytes, measured on the ATM switch in the configuration of Fig. 2 (a), just as a TCP connection is opening its window enough that the switch must drop data. Again, the switch input runs at 155 Mbps, the output at 53, and there are 32K bytes of buffer space available. Each peak is caused by the back-to-back arrival of a pair of packets at the full input rate. The peaks are spaced out because packets leave (and are acked and thus new ones transmitted) at the slower output rate. The window increase happens just before time 0.21. The two disks above the buffer use line indicate times at which the switch hardware indicated it was dropping cells. Each disk represents a few dozen lost cells.

The behavior depicted by Fig. 5 is typical: the switch typically drops cells from two packets when TCP opens its window too far. Since fast retransmit reliably recovers from only one lost packet, it leaves a lot of bandwidth unused while it times out.

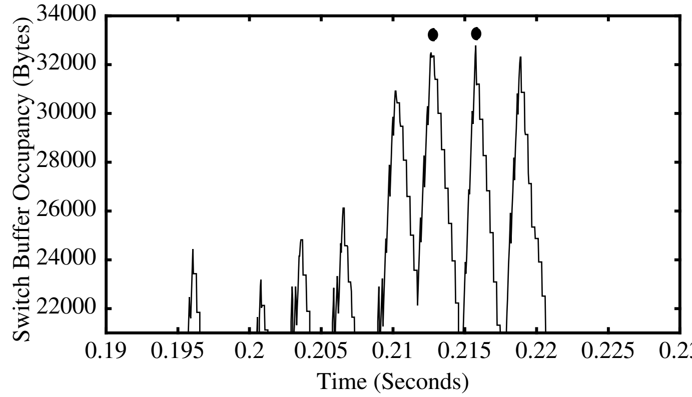

Fig. 5: Measured switch buffer occupancy as TCP increases the window size over time. The disks mark dropped cells.

Intuitively, TCP has increased its window or the amount of data it wants the switch to buffer by one packet. Since the switch hasn't enough space, it must discard up to one packet's worth of data. A packet switch would drop one entire packet. But the ATM switch does not know about packets, so it typically drops a packet's worth of cells spread over multiple packets.

A more formal argument that drops from multiple packets are common can be made. Assume that under the old window size, some number of cells N will be free just after the end of each packet's arrival, and thus the switch buffer has enough space to accommodate the future arrival of N cells plus one packet just before each packet arrives. When TCP increases its window by a packet, the switch can buffer the first N cells of this packet but must drop some of the rest. Since at least N extra cells are buffered, at most one packet's worth of buffer is available when the next packet arrives. If the next packet is even one cell early, some of it must be dropped.

Figure 6 illustrates the situation predicted by this argument. It plots predicted (not measured) switch buffer use as a function of time, much like Fig. 5. In this graph, however, time is measured in packet arrival times at the switch buffer, and the vertical axis in packets worth of switch buffering. The input to the switch buffer runs at three times the speed of the output. The disks mark the times at which the sender starts to send a packet; the packet transmitted at time 2.333 (packet i in Fig. 6) is the extra packet in a growing window. The dashed line shows what would happen if there were no limit on buffer space. The solid line shows what happens in a switch that can buffer only two packets. The first half of packet i is buffered. Some of the second half is dropped, but some is buffered since the switch is transmitting at the same time.These fragments cause the switch buffer to overflow again when the packet sent at time 3 arrives. In this way two packets are damaged

A precise explanation is given as follows. The white circle in Fig. 6 indicates the maximal switch buffer occupancy level

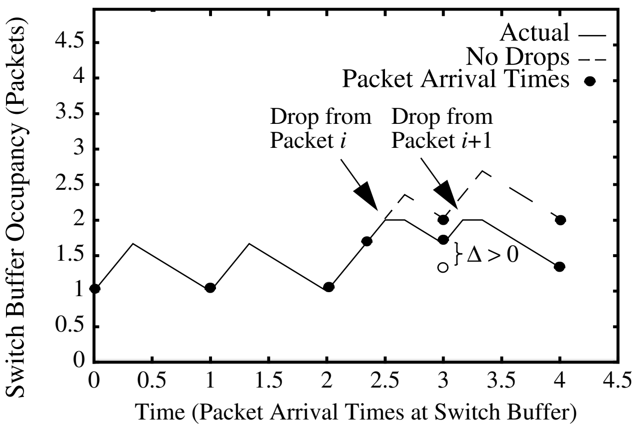

Fig. 6: Predicted graph of switch buffer space used as TCP increases the window size over time. The disks indicate when each packet starts to arrive. The arrows point to dropping. The white circle indicates the maximal switch buffer occupancy level beyond which a drop from packet i+1 will occur.

beyond which there will be drop from the next packet, packet i+1, will occur. The thick line in Fig. 7 shows how the vertical position of the white circle is determined. Note that the Δ value in Fig. 6 is greater than zero, as explained in the caption of Fig. 7. Therefore cells will be dropped from packet i+1.

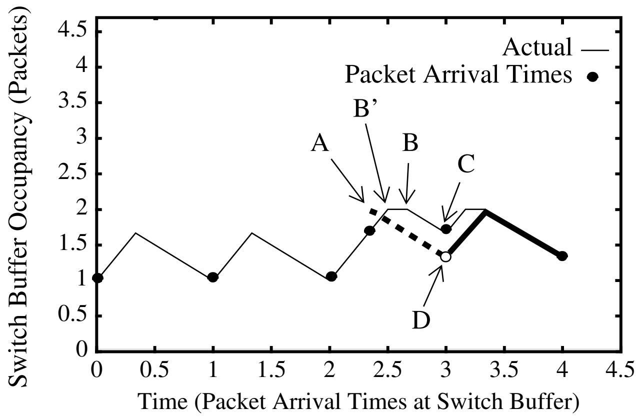

Fig. 7: A proof that the Δ value in Fig. 6 is greater than zero by noting that B is to the right A and thus C is above D. The dotted line has its slope equal to the output rate of the switch buffer.

With Partial Packet Discard, B will move to B' in Fig. 7. Thus C will move down somewhat but will still be above D. Therefore the switch will have increased buffer space available for packet i+1 but will still drop part of it. Experiments confirm that Partial Packet Discard has only a marginal effect on efficiency. Perhaps the reason for the good throughput performance of Early Packet Discard in the simulation results [13] is that it concentrates all the dropped cells into a single packet, from which TCP's fast-retransmit mechanism can recover.

The same phenomenon occurs when two TCPs enter a switch on different fast links and leave the switch sharing a slower link. The TCPs rarely compete against each other. Both spend most of their time in retransmit time-outs; whenever either starts to send, it almost immediately opens its window far enough that the switch drops multiple packets.

V. TCP PERFORMANCE WITH ATM FLOW CONTROL

ATM-level credit flow control resolves these TCP performance problems over ATM as described in the preceding section. The bottleneck switch no longer discards data when it runs out of buffer memory. Instead, it withholds credit from the switches and/or hosts upstream from it, causing them to buffer data instead of sending it. This backpressure can extend all the way back through a network of switches to the sending host. The effect is that a congested switch can force excess data to be buffered in all the upstream switches and in the source host. Data need never be lost due to switch buffer overrun. Thus if TCP chooses a window that is too large, the data will simply be buffered in the switches and in the host; no data loss and retransmission time-outs will result.

Fig. 8 compares the useful bandwidths achieved with and without credit-based ATM-level flow control in the configurations shown in Fig. 2. For the flow-controlled cases, the switch has 100 cell buffers (4800 payload bytes) reserved per VC. For the non-flow-controlled cases, the switch has 682 (32 payload kilobytes) of buffering per VC. Recall that for the configuration in Fig. 2 the slow link can deliver at most 5.7 payload megabytes per second, and the fast link 17. Thus in both one TCP and two TCPs cases, TCP with credit-based flow control achieves its maximum-possible bandwidth.

	Without Flow Control	With Flow Control
(a) One TCP	0.1 MByte/sec	5.7 MByte/sec
(b) Two TCPs	0.2 MByte/sec	5.7 MByte/sec

Fig. 8: Measured total bandwidth achieved with and without ATM-level credit-based flow control, for the (a) and (b) configurations of Fig. 2.

Using a configuration similar to Fig. 2 (b), experiments involving one TCP and one UDP instead of two TCPs have also been carried out. A typical measured result is as follows. When ATM-level credit-based flow control is used, UDP gets its maximum bandwidth only limited by the source, while TCP gets essentially the remaining bandwidth of the bottleneck link between the two switches. However, when credit-based flow control is turned off, TCP's throughput is significantly dropped and the total utilization on the bottleneck link by both TCP and UDP reduces to be less than 45%. Thus, when competing with UDP, TCP with ATM-level flow control can keep up its throughput even though UDP does not reduce its bandwidth during network congestion.

Fig. 9 shows how much switch buffer space is used when one TCP sends into a slow link with credit flow control turned on, for the configuration depicted in Fig. 2 (a). The flow control system makes sure that enough cells are always buffered that it can keep the output link busy, but never much more than that. The large oscillations correspond to packet boundaries.

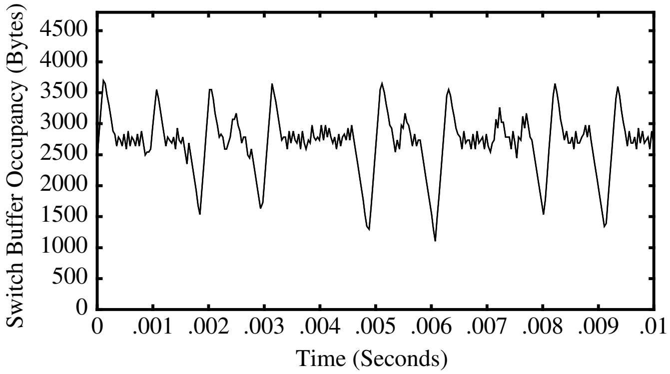

Fig. 9: Measured switch buffer occupancy when one TCP sends into a slow link, as depicted in Fig. 2 (a), with credit flow control turned on.

ATM flow control requires switch memory proportional to the bandwidth-delay product of the links attached to the switch [6]. In a local-area net with low propagation delays a few dozen cells per VC might suffice, since this space would be supplemented by large host memories. For this reason credit flow control should scale well.

VI. CONCLUSIONS

Many investigators have noted that TCP performs worse over cell-switched ATM than it does over packet-switched networks. The experiments and analysis described here suggest that this is caused by the particular pattern in which ATM switches tend to drop cells. This pattern implies that non-flow controlled, cell switches will likely drop more than one packet in a row and as a result the efficient TCP fast transmission mechanism for single lost packet will not apply.

While TCP performance can be made quite good using packet switches, the amount of switch memory required is often bounded below by packet size rather than the more fundamental limit of bandwidth-delay product. A switch with ATM-level flow control can achieve near-perfect link utilization while approaching the minimum possible buffer use.

ACKNOWLEDGMENTS

This research was supported in part by BNR and Intel, and in part by the Advanced Research Projects Agency (DOD) monitored by ARPA/CMO under Contract MDA972-90-C-0035 and by AFMC under Contract F19628-92-C-0116.

REFERENCES

[1] Blackwell, et al. “An Experimental Flow Controlled Multicast ATM Switch,” *Proceedings of First Annual Conference on Telecommunications R&D in Massachusetts*, Vol. 6, October 25, 1994, pp. 33-38.

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

[3] CCITT, “Draft Recommendation I.363”, CCITT Study Group XVIII, Geneva, 19-29 January 1993.

[4] Chran-Ham Chang et. al., “High-performance TCP/IP and UDP/IP Networking in DEC OSF/1 for Alpha AXP,” Digital Technical Journal, Winter 1993.

[5] V. Jacobson, “Congestion Avoidance and Control,” ACM SIGCOMM 1988 Conference Proceedings, August 1988.

[6] H. T. Kung, T. Blackwell and A. Chapman, “Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing,” SIGCOMM 1994, pp. 101-114.

[7] H. T. Kung and K. Chang, “Receiver-Oriented Adaptive Buffer Allocation in Credit-Based Flow Control for ATM Networks,” INFOCOM ’95, April 1995, pp. 239-252.

[8] H. T. Kung and A. Chapman, “The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks,” Version 2.0, 1993. A summary appears in Proc. 1993 International Conf. on Network Protocols, pp. 116-127. (Postscript files of this and other related papers by the authors and their colleagues are available via anonymous FTP from virtual.harvard.edu:/pub/htk/atm.)

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

[10] S. Leffler, et al. *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley, 1989.

[11] G. Minden, V. Frost, J. Evans, and B. Ewy, “TCP/ATM Experiences in the MAGIC Testbed,” ftp://ftp.tisl.ukans.edu/pub/papers/TCP-Perform.ps.

[12] C. Ozveren, R. Simcoe, and G. Varghese, “Reliable and Efficient Hop-by-Hop Flow Control,” SIGCOMM ’94, pp. 89-100.

[13] A. Romanow and S. Floyd, “Dynamics of TCP Traffic over ATM Networks,” ACM SIGCOMM 1994 Conference Proceedings, October 1994.

[14] G. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume 2*, Addison-Wesley, 1995.