[5] V. Jacobson, "Congestion avoidance and control," *Proc. ACM SIGCOMM '88*, pp. 314-329.

[6] V. Jacobson, "Berkeley TCP evolution from 4.3-tahoe to 4.3-reno," *Proc. of the 18th Internet Engineering Task Force*, Vancouver, August, 1990.

[7] V. Jacobson, R. Braden and D. Borman, "TCP extensions for high performance," RFC (request for comment) 1323, May 1992.

[8] P. Karn and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," *ACM Trans. on Computer Systems*, vol. 9, no. 4, pp. 364-373, November 1991.

[9] T. V. Lakshman and U. Madhow Performance analysis of window-based flow control using TCP/IP: the effect of high bandwidth-delay products and random loss. *IFIP Transactions C-26, High Performance Networking V*, pp. 135-150, North-Holland, 1994.

[10] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer," *Proc. ACM SIGCOMM '88*, pp. 303-313.

[11] A. Romanow and S. Floyd, "Dynamics of TCP Traffic over ATM Networks" Proc. ACM Sigcomm Conference 1994, pp.79-88.

[12] S. Shenker, L. Zhang, and D. D. Clark, "Some observations on the dynamics of a congestion control algorithm," *Computer Communication Review*, pp. 30-39, October 1990.

[13] N. Yin, and M. G. Hluchyj, Implication of Dropping Packets from the Front of a Queue. 7-th ITC, Oct 1990. Copenhagen, Denmark.

[14] L. Zhang, "A new architecture for packet switching network protocols," Ph. D. dissertation, M.I.T. Lab. Comput. Sci., Cambridge, MA, 1989.

[15] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic," *Proc. ACM SIGCOMM '91*, pp. 133-147.

protocol, in particular the fast retransmit feature, this earlier notification (or reduced latency) translates into larger throughput. In addition, we experimentally found that "Drop from Front" policies reduce (compared with "Drop from Tail" mechanisms) the throughput advantage that TCP sessions with shorter Round Trip Times have, and so drop from front alleviates unfairness. This is particularly so when there are very few active sources. We also gave an explanation for this experimentally observed phenomena.

We found that drop from front typically requires less buffering. For good throughput, the simple "Pure Cell Drop from the Tail" policy needs a normalized buffer size of $\beta = 3$ to get high throughput. The more sophisticated "Partial Frame Drop from the Tail" requires β values of 2 and even 3 in some cases for high throughput. In comparison, "Pure Cell Drop at the Front" and "Partial Frame Drop at the Front" allow β values close to 1 and possibly even less than one for "Partial Frame Drop at the Front".

We also compared TCP performance under drop from front to that with RED. If these schemes are used by themselves at the cell level and are not used in conjunction with any form of frame drop, then drop from front results in higher performance than RED. Also, RED's performance is sensitive to the range of probabilities used in the drop function. RED's performance is considerably better when RED is used in conjunction with whole frame drop. Drop from front with partial frame drop gives almost similar performance. However, this slightly lower throughput may not be sufficient to warrant the use of RED since drop from front can be used for other traffic classes as well since it reduces the average latency [13]. So when using drop from front there is no explicit need for the switch to know which virtual circuits are carrying TCP traffic.

While our simulations were done for an ATM cell-buffer, we expect similar differences between the performances under Taildrop and Drop from Front in router based packet buffers. Even though the effectiveness of "Drop from Front" depends on the fast retransmit feature of the TCP protocol, the implementation of "Drop from Front" is entirely in the switch. No change to the TCP protocol is necessary.

Acknowledgements

We thank Mark Garrett for helpful discussions about activity in the ATM Forum. We thank Dave Sincoskie for helpful suggestions about efficient implementations of "drop from front".

References

[1] L. S. Brakmo, S. W. O'Malley and L. L. Peterson, " TCP Vegas: New Techniques for Congestion Detection and Avoidance" *Proceedings of ACM SIGCOMM '94*, May 1994.

[2] S. Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic," *Computer Communications Review*, vol.21, no.5, pp. 30-47, October 1991.

[3] S. Floyd and V. Jacobson, "On Traffic Phase Effects in Packet-Switched Gateways," *Internetworking: Research and Experience*, vol.3, no.3, pp.115-156, September 1992. (An earlier version of this paper appeared in *Computer Communication Review*, vol. 21, no. 2, April 1991.)

[4] S. Floyd and V. Jacobson, "Random Early Detection gateways for congestion avoidance", IEEE/ACM Transactions of Networking, vol. 1, no. 4, pp. 397-413, August 1993.

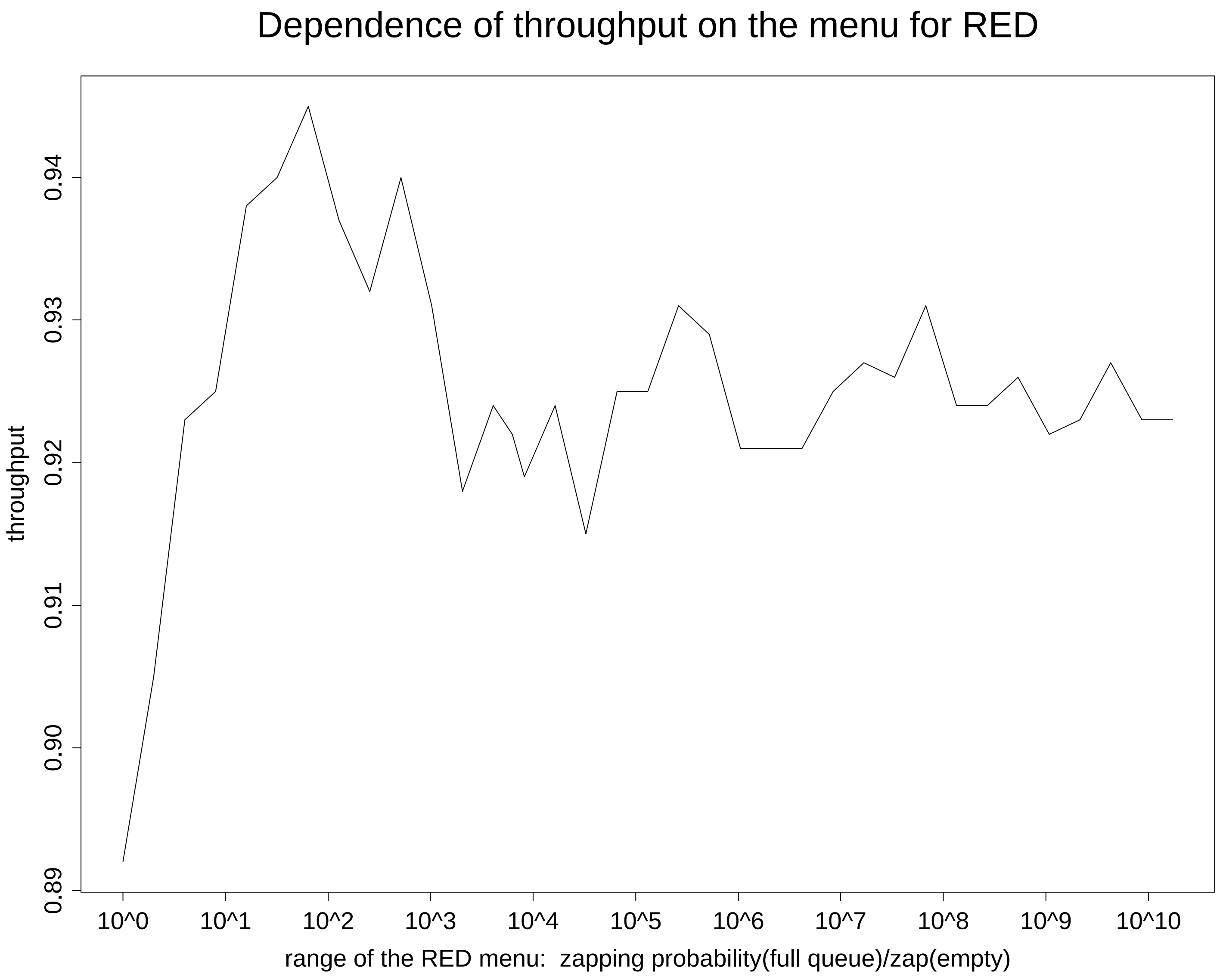

Figure 14: Robustness to parameters of exponential drop function

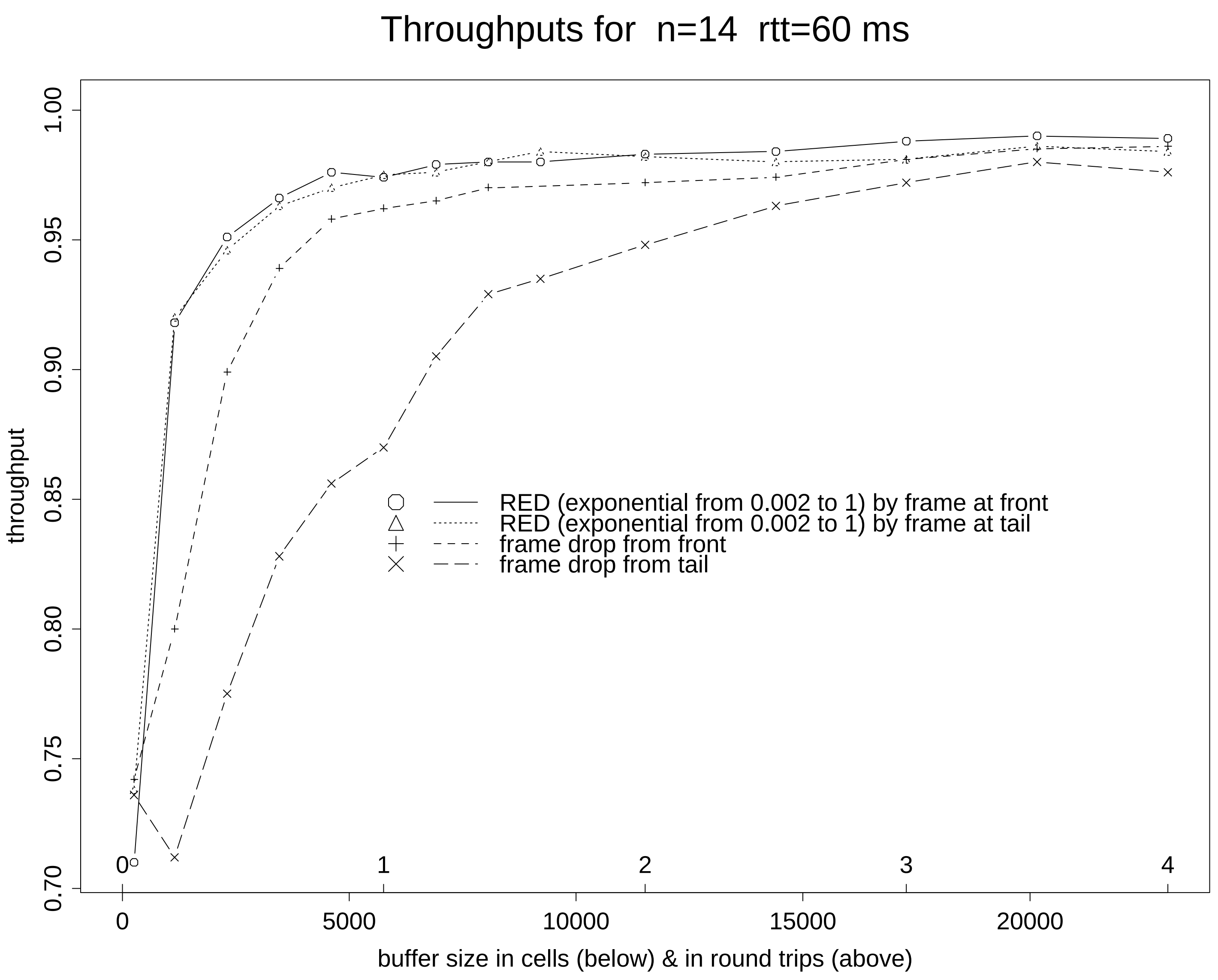

Figure 13: Performance of RED used at front and RED used at tail; applied to frames

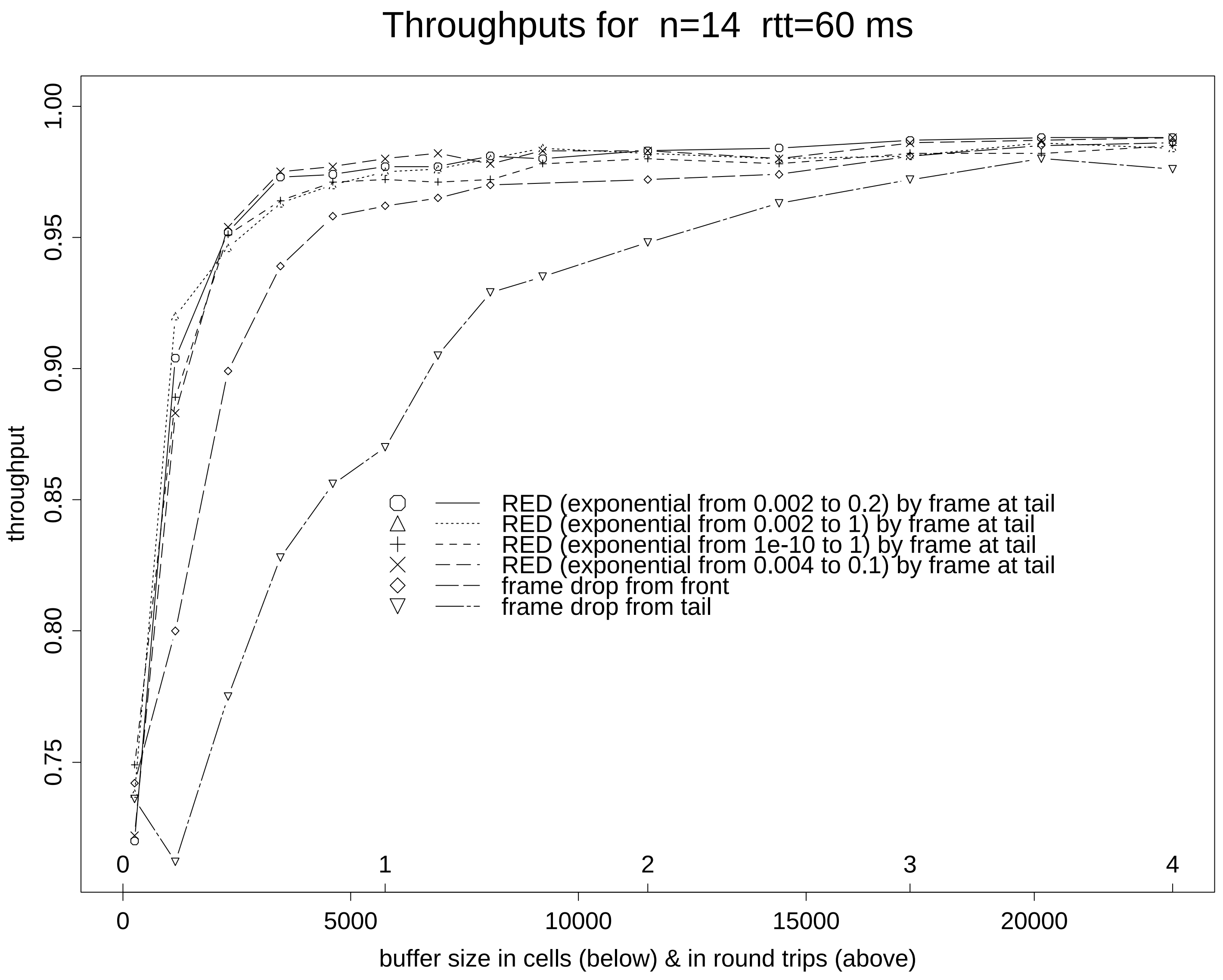

Figure 12: Drop from front with partial frame drop and RED with whole frame drop

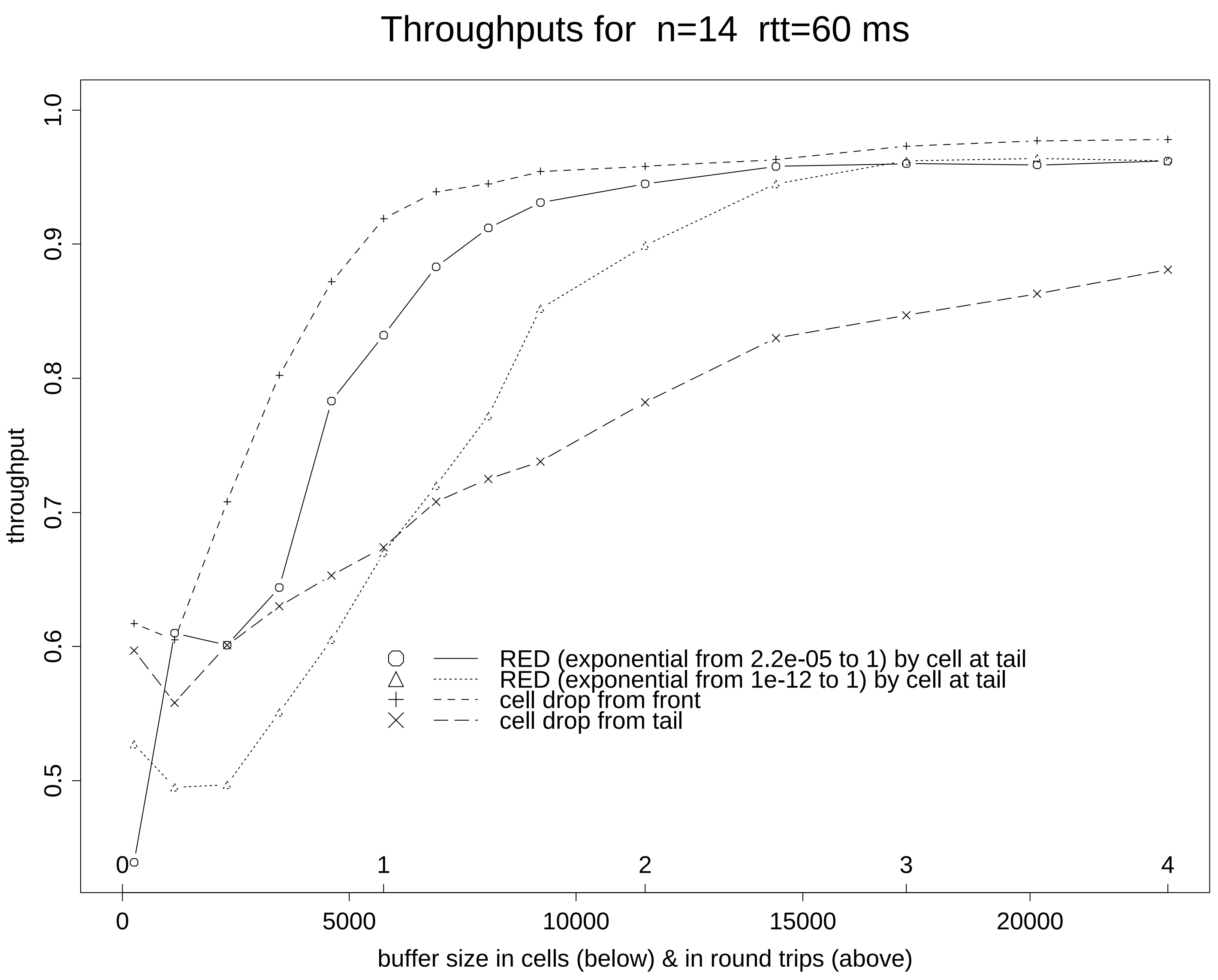

Figure 11: Drop from front and RED; no form of frame drop

5 Performance Comparison with RED

Figure 11 compares the performance of drop from front to that of RED. The version of RED used is as explained in Section 2. The drop probabilities increase exponentially with buffer occupancy. The legend in the figure indicates the extreme values of the drop probabilities corresponding to the buffer occupancy being equal to one cell and the buffer being full. The figure also shows the performance of drop from tail for reference. All the plots in figure 11 show results corresponding to the pure cell drop case – i.e. there is no form of frame drop used in conjuction with any of the schemes. The results are for the combined throughput of 14 connections with equal round trip times (60 ms in this case). The X-axis shows the buffer size in both cells and in multiples of round trip delay (the service rate, as for all simulations in this paper, is the DS-3 rate). From the figure it can be seen that drop from front has the best performance for all buffer sizes. The TCP performance under RED not only is less than that of drop of front but also is sensitive to the precise drop probabilities. Indeed, if the drop probabilities span a wide range (10^{-12} to 1 in the figure) then RED sometimes performs even worse than drop from tail.

Figure 12 again compares the performance of drop from front to that of RED except that now both are used in conjunction with frame drop. For drop from front, the frame drop used is partial frame drop and for RED the frame drop scheme used drops whole frames (as explained in Section 2). When RED is applied at the frame level, the performance improvement is more impressive than when RED is applied at the cell level. Also, when applied at the frame level the throughputs are fairly robust to the specific parameters of the exponential drop function as can be seen from the 4 plots for RED throughput in Figure 12. The performance is slightly better than that of even frame dropping from the front. As can be seen from Figure 12, either RED by the frame or dropping frames from the front gives a large improvement over dropping frames from the tail (which was already significantly better than dropping cells from the tail). Visually, the curves for RED and for dropping from the front give nearly the same impression of dramatic improvement in using the bottleneck resource efficiently. However, drop from front has the advantage of not having to know that the connection is a TCP connection since even for other sources it reduces the average latency.

RED can be used at the front of the buffer or at the tail of the buffer. Figure 13 shows that this does not make much of a difference.

Figure 14 illustrates the robustness of RED applied at the frame level to the parameters of the exponential function used for the drop probabilities. The X-axis is the ratio REDfull/REDone where REDfull is the drop probability when the buffer is full and REDone is the drop probability when the buffer has occupancy of one cell. So the X-axis indicates the range of probabilities. The throughput results are for six connections with equal round trip times of 60 ms and a bottleneck buffer size of 2304 cells. The graph shows that range doesn't really affect the throughput for RED at the frame level.

6 Conclusions

In this paper, we proposed using "Drop from Front" buffer management policies to improve TCP performance in high bandwidth-delay product networks. The motivation was to further exploit the fast retransmit mechanism in TCP. We studied by simulation the value of "Drop from Front" as a performance enhancing mechanism for TCP over ATM. "Drop from Front" gives sources earlier notification of starting congestion than "Tail Drop" mechanisms do. Through the TCP end-to-end

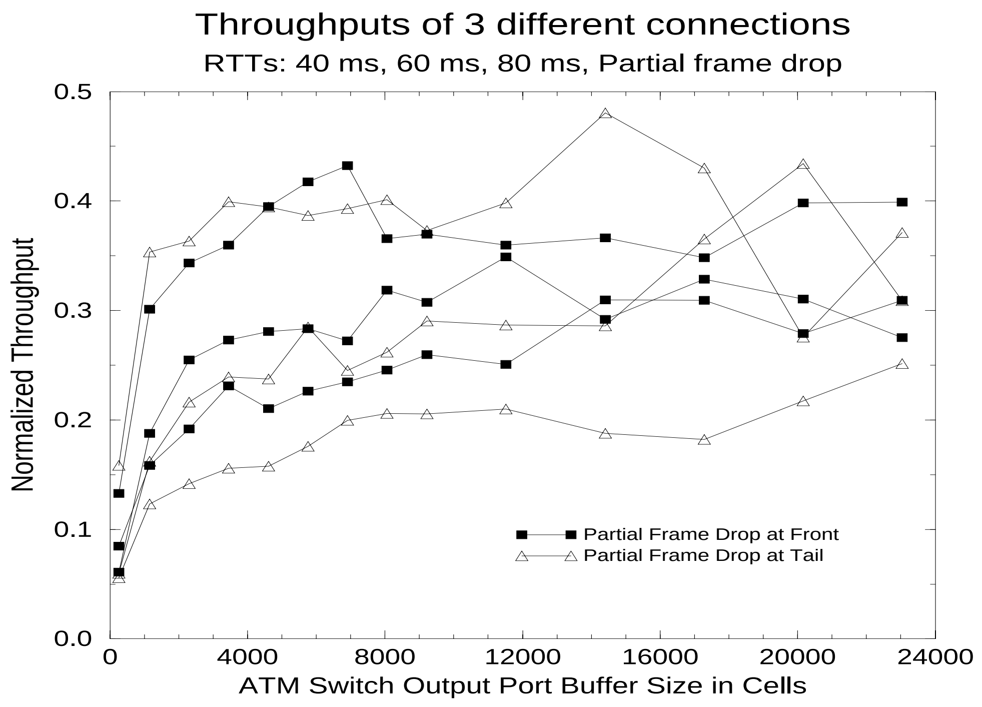

Figure 9: Individual throughputs, Different RTTs,$B = 5760$ cells for $\beta = 1$

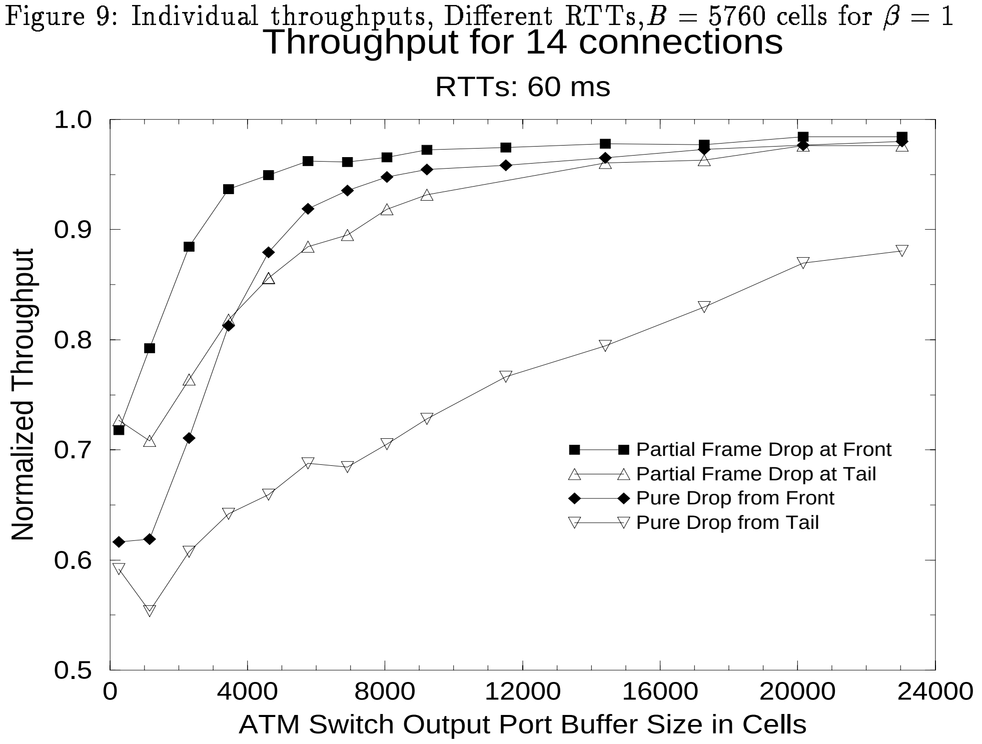

Figure 10: Total throughputs, Equal RTTs, $B = 5760$ cells for $\beta = 1$

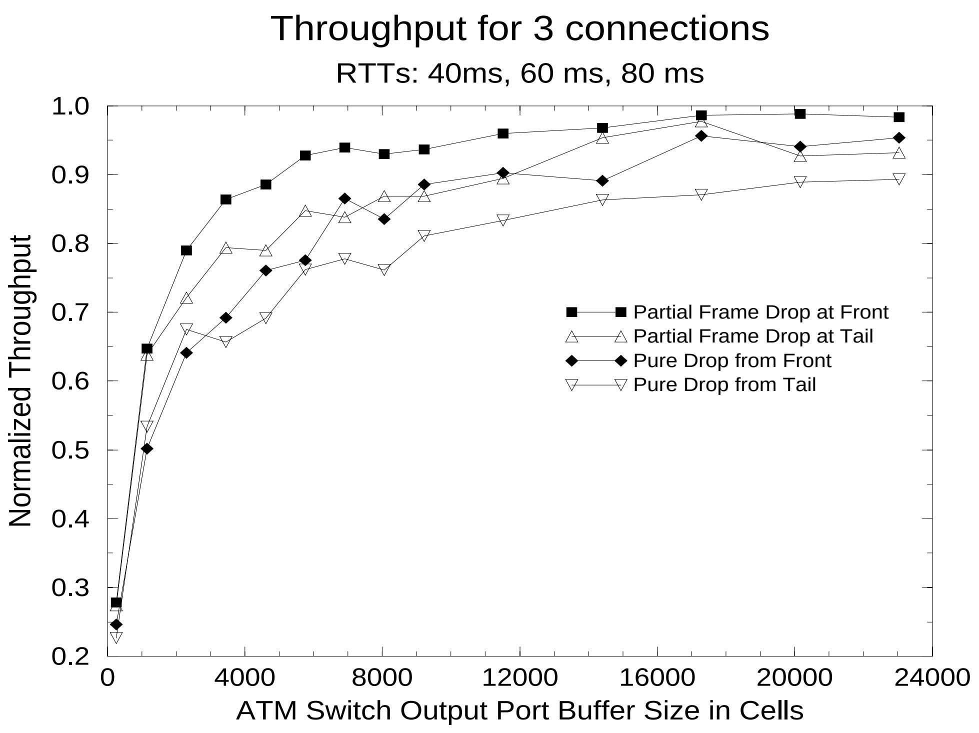

Figure 7: Total throughputs, Different RTTs, $B = 5760$ cells for $\beta = 1$

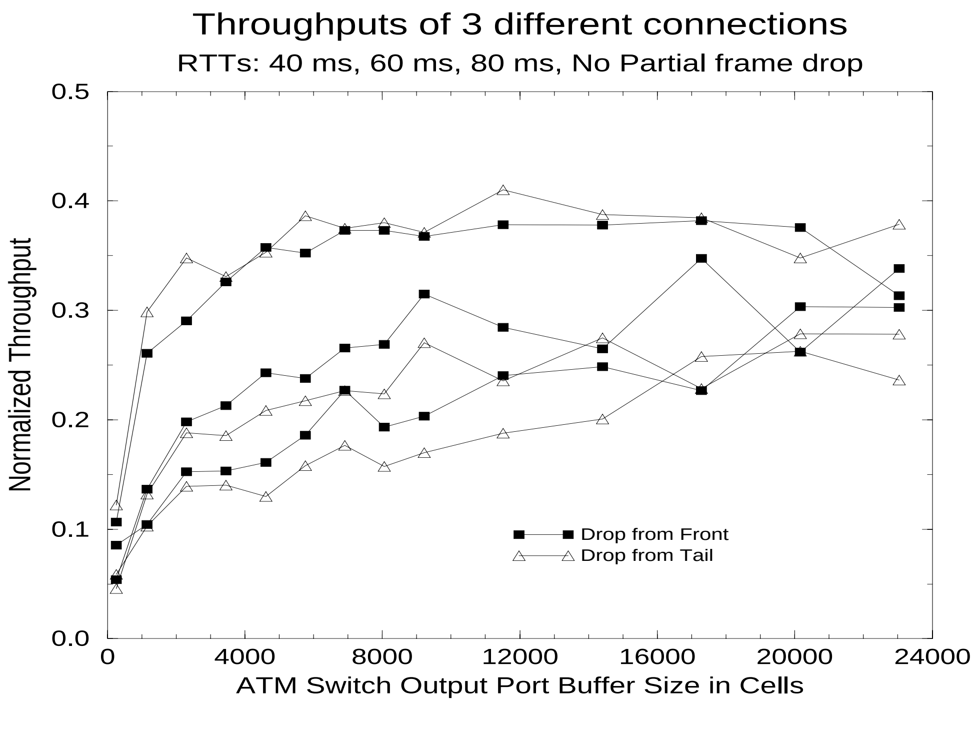

Figure 8: Individual throughputs, Different RTTs, $B = 5760$ cells for $\beta = 1$

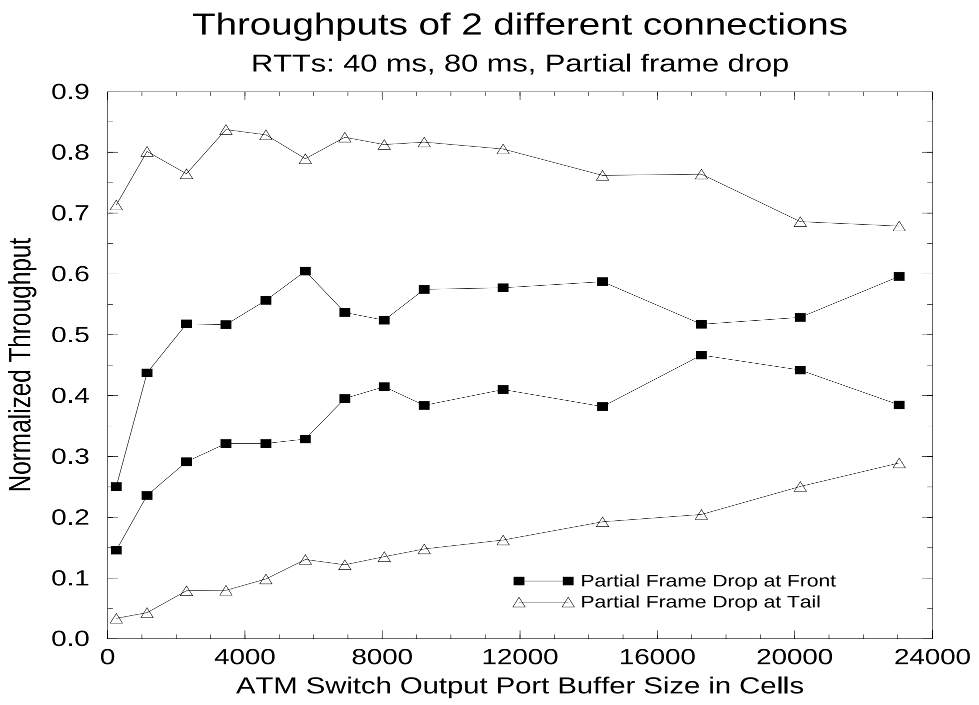

Figure 5: Individual throughputs, Different RTTs,$B = 5760$ cells for $\beta = 1$

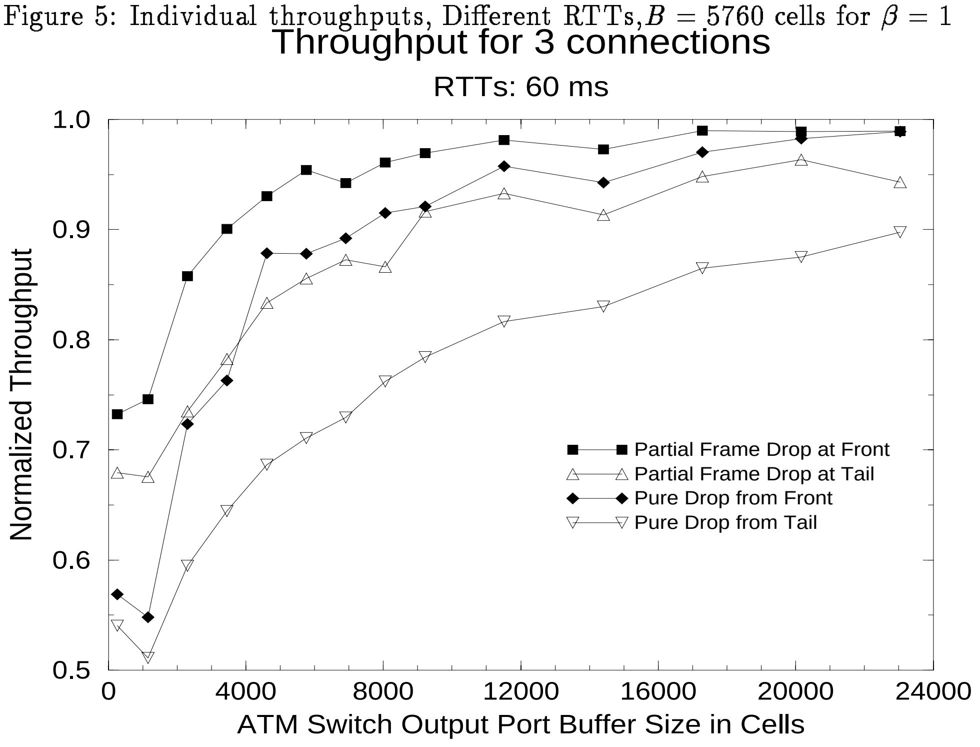

Figure 6: Total throughputs, Equal RTTs,$B = 5760$ cells for $\beta = 1$

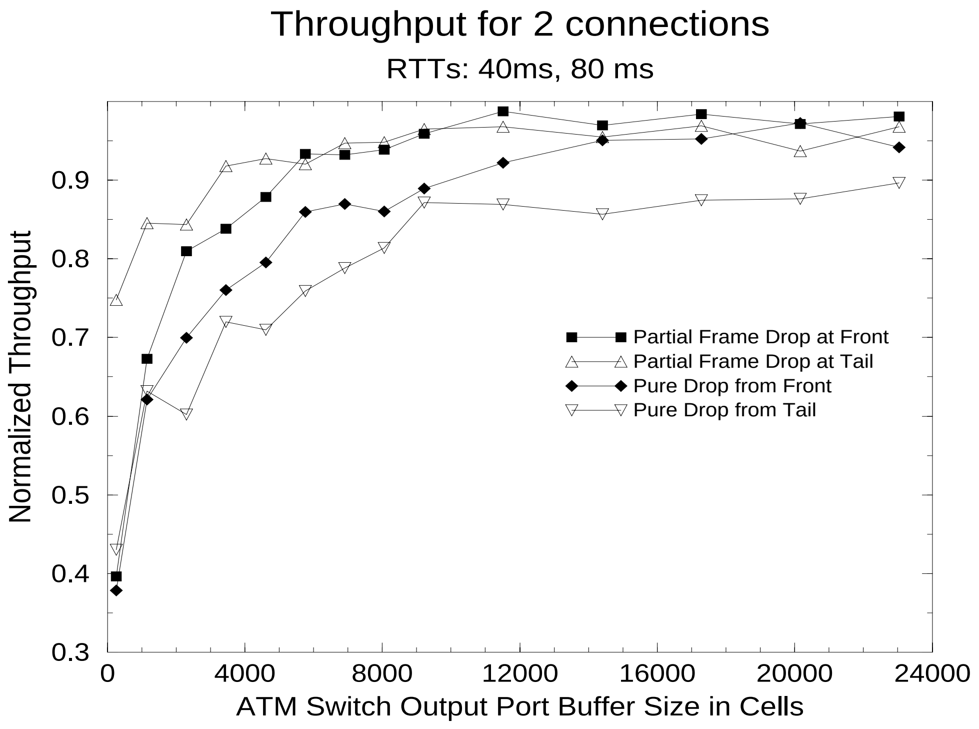

Figure 3: Total throughputs, Different RTTs, $B = 5760$ cells for $\beta = 1$

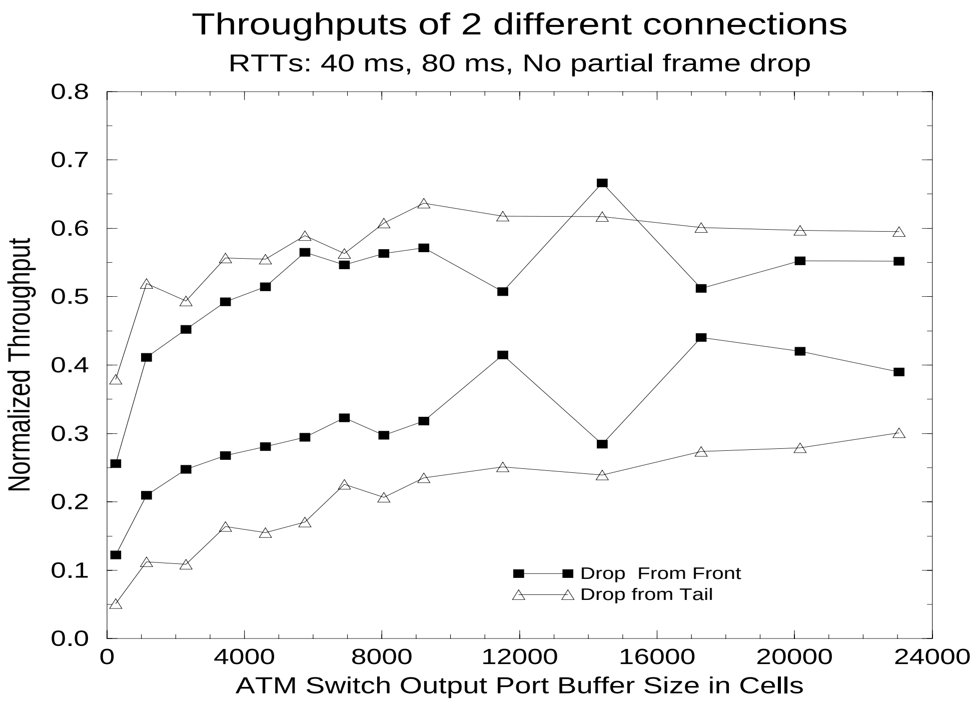

Figure 4: Individual throughputs, Different RTTs, $B = 5760$ cells for $\beta = 1$

"Front" policies. For the explanation, consider only the case of two active sources. Under "Tail" policies, the two sources tend to lose about the same numbers of packets. For Cell-Tail, this is true because whenever two packets collide while the buffer is full, both get damaged. For Frame-Tail this is true because whenever two packets are colliding while the buffer is full, at random one of the two connections loses all cells from the point where the buffer gets full. On the other hand, under "Front" policies, the number of packets a source loses is roughly proportional to its throughput. This alleviates the unfairness somewhat.

Before final conclusions can be drawn on buffer management schemes, a wider variety of situations must be investigated. For the time being, the following conclusions are warranted:

For Pure Cell Drop at the Tail, β should be at least in the range 3 or 4 to get "decent" throughput for a reasonable range of "number of sources - RTT combinations". For Pure Cell Drop at the Front, a value of $\beta = 1$ is acceptable, although $\beta = 1.25$ might be better choice. Partial Frame Drop at the Tail may need much larger values of β because of larger unfairness for β as high as 2 or 3 if the number of active sources is small and the RTT differential is high (for an explanation of why unfairness decreases somewhat with larger buffer size, see [9]).

Partial Frame Drop at the Front comes out best, allowing values of β as low as 1 (maybe even a bit lower) and having the least amount of unfairness to connections with larger round-trip times.

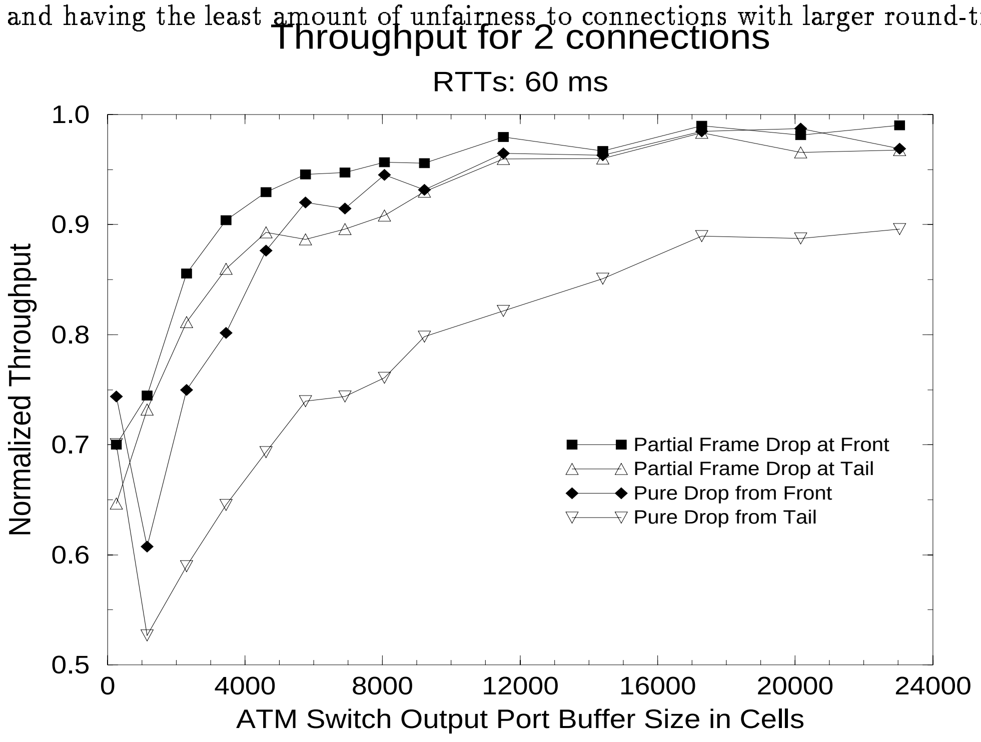

Figure 2: Total throughputs, Equal RTTs, $B = 5760$ cells for $\beta = 1$

Only transmitted cells that are not part of a damaged packet, or of a later repeated packet, count toward throughput. The plots show normalized throughput with 1 being perfect throughput (no unused cell slots and all cell slots utilized by good cells). All plots use output buffer sizes of 256, 1152, 2304, 3456, 4608, 5760, 6912, 8064, 9216, 11520, 14400, 17280, 20160, and 23040 cells. At an "external" round trip time (RTT) of 60 msec (external to the output buffer), a buffer size of 5760 cells corresponds to $\beta = 1$ (max delay in the buffer equals RTT outside the buffer), and a buffer size of 23040 corresponds to $\beta = 4$. The RTTs listed in the figures are the RTTs excluding the delay in the buffer. As mentioned in Section 3, we used two packet sizes in the simulation: 4388 bytes for data packets and 64 bytes for acknowledgement (ACK) packets. These translate to 92 cells per data packet and 2 cells per ACK packet.

The Figures 2, 3, 6, 7, and 10 give total normalized throughput for respectively the cases of: 2 active sources, each (external) RTT 60 msec; 2 active sources, RTTs 40 and 80 msec; 3 active sources, each RTT 60 msec; 3 active sources, RTTs 40, 60, 80 msec; 14 active active sources, each RTT 60 msec. In all those cases each source has its own DS-3 channel into the switch and all the competing streams use the same output port. In the cases of equal RTTs, β is well-defined, but in the cases of unequal RTTs (Figures 3 and 7) β is no longer well-defined: The average of the RTTs weighted by throughput is less than 60 msec, because of the bias, by TCP, in favor of sessions with short RTTs. For all simulations, the TCP sources always have data to send and the only limiting mechanism is the windowing mechanism.

Each of the Figures 2, 3, 6, 7, and 10 contains four plots of normalized total throughput vs buffer size. These correspond to the four drop disciplines studied: Pure Cell Drop at the Tail (abbreviated as Cell-Tail); Pure Cell Drop at the Front (Cell-Front); Partial Frame Drop at the Tail (Frame-Tail); and Partial Frame Drop at the Front (Frame-Front). We see that "Partial Frame Drop" helps improve throughput: consistently, Frame-Front has better throughput than Cell-Front, and consistently Frame-Tail has better throughput than Cell-Tail. In addition, with rare exceptions, Cell-Tail is the poorest performer. The exceptions all occur for very low buffer sizes, where "lockout" due to phase effects play a role despite the randomization. (Lockout occurs when one connection grabs all the bandwidth and virtually squeezes out all other sources). Lockout generally leads to good throughput because it causes behavior as if there is only one source which in our simulations is not capable of causing congestion. We also see that with rare exceptions throughput increases when the buffer size increases. Again, the rare exceptions occur for very small buffer sizes, where throughput is high due to lockout.

By and large, Frame-Front is better, often much better, than Cell-Front and also better than Frame-Tail. Frame-Tail is better than Cell-Front for small buffer sizes but Cell-Front becomes better than Frame-Tail for larger buffer sizes. This is in fact the case whenever all connections have the same RTT (Figures 2, 6, and 10). When RTTs are unequal (Figures 3 and 7), Frame-Tail starts doing better. This is particularly the case when the number of sources is very small (Figure 2) where for small buffer sizes Frame-Tail even does better than Frame-Front.

The Figures 5 (2 sources) and 9 (3 sources) show that this relative improvement of Frame-Tail for unequal RTTs is, to a large degree, due to lock out and unfairness: Under all circumstances the connection with the shorter RTT has an advantage, and under Frame-Tail this advantage is much larger than under Frame-Front. This effect is stronger for 2 sources than for 3 sources.

The Figures 4 (2 sources) and 8 (3 sources) show that a similar, but weaker, difference in bias is present between Cell-Tail and Cell-Front.

Next we give an explanation for this stronger degree of lock out under "Tail" policies than under to the buffer in the simulation) cells in the end of the buffer (this is only used to create buffer space, FIFO ordering is not violated). Drop from front does not suffer from phase effects and so no randomization is needed.

The link propagation delays are set to vary in the range 100 μsecs to 2800 μsecs. The propagation delay is fixed for a specific link but differs from link to link. The switch samples the switch input ports (where the DS-3 links terminate) at a sampling rate of once every 2.8 μsecs. The internal switch timing is simulated accurately as also the architecture of a popular ATM switch. For brevity, we do not describe in detail the internal timing and architecture of the switch used in the simulations. For our performance studies, it is sufficient to know that there is no queueing or blocking internal to the switch and contention occurs only at the switch output ports. Congestion is caused by routing many connections to the same output port (port 0 in the simulations).

The drop from front, and the partial frame drop variants, were implemented at all switch output port buffers. The cell picked for transmission with partial frame drop from front is always a cell that does not belong to a dropped frame. Cells from damaged frames at the front of ther buffer are dropped till a cell from an undamaged frame is found.

The output links of the switch terminate at the ADSUs. Cells arriving at the ADSU are assembled into packets and put in the queue at the destination side. Packets with lost cells which cannot be successfully reassembled at the ADSU are dropped at the ADSU. The TCP receiver on the destination side, ACKs every packet. The ACK is delayed by a specified value and in our simulations this delay is set to 60 ms for modeling US transcontinental delays. This delay could well have been added to the propagation delay of the links and was added in the destination only for convenience. Note that this is not the same as using the "delayed ACK sending" used in some TCP implementations where cumulative ACKs are sent (in the absence of reverse traffic) only upon expiration of an interval (typically 200 ms) timer. We did not use this "delayed ACK sending" mode in the simulations for the results presented. ACKs are sent back through the switch from the destination to source in the same manner as data packets. Since we were only interested in situations where the bottleneck is in the network, we simulated systems where end-system buffers never overflowed. Send window sizes were strictly determined by network buffering and capacity, and not due to end system limitations. Because we are interested in networks with high-bandwidth delay products, window scaling as proposed in [7], may be necessary. We also assumed the time-stamp option. We used a timer with much finer granularity than the typical 500 ms. This is not significant since most losses (in all cases except during periods when windows were less than 3 packets) are detected by the fast retransmit procedure which does not use timers. Round trip times were estimated ignoring ambiguous ACKs as in [8].

For the simulations, we used a version of TCP-Reno, which includes the fast retransmit option and a method for reducing the incidence of slow start, proposed by Van Jacobson in 1990 [6]. Even though we did not simulate TCP-Tahoe with fast retransmit, nor the recently proposed Vegas enhancements [1] to TCP, we expect the earlier congestion warning to translate to throughput increases for these cases also.

4 Performance Comparison with Tail Drop

The figures in this section present the simulation results comparing the performance of drop of front with that of tail drop. All figures show throughput as a function of the congested ATM switch output port buffer size. Throughput is defined as the "good throughput" seen by TCP sources:

3 The Simulated System

The simulated system is shown in Figure 1, TCP sources are connected to routers and ATM Data

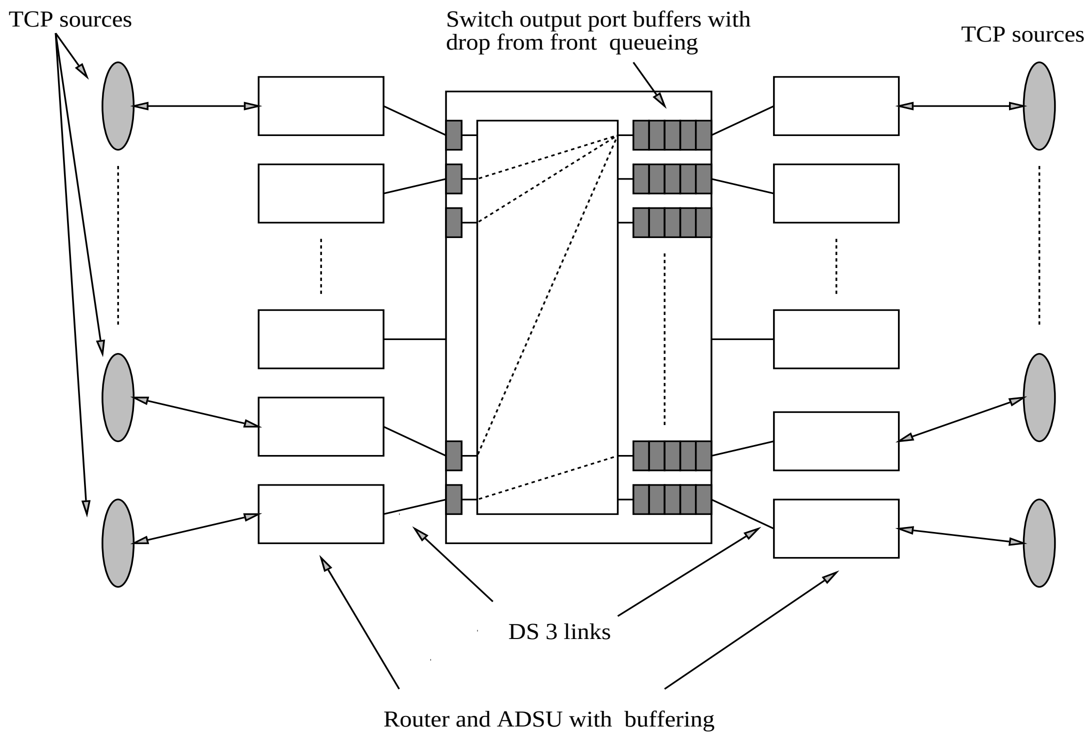
TCP sources
Switch output port buffers with drop from front queueing
TCP sources
DS 3 links
Router and ADSU with buffering

Figure 1: Simulated system with TCP sources, Routers, ADSUs, and ATM switch

Service Units (ADSUs). The router receives packets from TCP sources and passes them to the ADSU which converts the packets to ATM cells. We used only two TCP packet sizes in the simulation: 4388 bytes for data packets and 64 bytes for acknowledgement (ACK) packets. These translate to 92 cells per data packet and 2 cells per ACK packet. Routers do not become bottlenecks in our simulations and buffering at the routers is kept sufficiently high so as not to cause losses[2].

On the sending side, the ADSU splits TCP packets into cells, assigns the right virtual circuit for transport (virtual circuits are set up statically during initialization) and transmits the cells on the outgoing DS-3 links from the ADSUs. The DS-3 links (using the Physical Layer Convergence Procedure, PLCP, over DS-3) typically operate in a slotted manner with slot duration 10.4 μsecs. However, on the input side to the switch, we did not use slotting on the DS-3 links and so cell transmissions from the ADSU do not have to wait until a slot boundary. The links connected to the switch output port operate in a slotted manner. The mix of slotted and unslotted links did not alleviate "winner take all" type phasing effects and we had to resort to randomization. For tail drop, when a cell arrives to a full buffer instead of dropping the incoming cell (slotting and phase effects ensure then that one source will always be the winner in entering the buffer), the cell dropped is equiprobably either the incoming cell or one of the first few (we used the fan-in factor

[2]No real routing functions are performed. The routers at present just add one more non-bottlenecking hop before and after the switch.

2 Buffer Management Strategies for TCP

This section gives a short description of relevant buffer management strategies that have been proposed to improve TCP performance.

2.1 Partial Frame or Packet Drop

Whenever a cell is dropped (from the front, tail, or anywhere else in the buffer) the TCP packet to which the cell belongs will be retransmitted entirely. So there is no point in transmitting cells from already damaged packets to the destination and further cells from a damaged packet may as well be dropped (not let into the buffer) to prevent wasted buffer and bandwidth usage. This strategy is called ”Partial Packet Discard” in [11]. In this paper, packet and ATM AAL5 frames are used synonymously and so we refer to this strategy as partial frame drop. The frame drop is partial because at the time of cell drop, some cell from the same frame may already have been transmitted. Also, no search of the buffer is done to eliminate all cells from damaged frames.

For partial frame drop from tail, whenever a cell is dropped a bit is set so that until the end of that frame all arriving cells from that frame are also discarded (the remainder of the frame is discarded). For partial frame drop at front, whenever (because a cell attempts to enter a full buffer) the cell at the front of the buffer is dropped, a bit is set and all remaining cells of the frame are dropped as they reach the front of the buffer.

Though Romanow and Floyd [11] propose partial frame drop as a mechanism for throughput improvement, for the WAN environment that we consider, partial frame drop by itself will not improve performance sufficiently. This is because wastage of bandwidth due to transmission of cells from damaged packets is only a secondary factor in throughput degradation. Partial frame drop must be used in conjunction with another scheme such as RED or drop from front.

2.2 Random Early Detection

In the form of RED that we consider, we associate a drop probability with each buffer occupancy level. When an item (such as a cell) arrives, it is dropped with probability equal to the drop probability for the current buffer occupancy. So when RED is applied at the cell level, the test of whether to drop an item or not is performed upon every cell arrival. To apply RED on frames as the droppable items, we perform the drop test only upon arrival of the first cell of a frame. If the test causes this cell to be dropped then all subsequent cells of the frame are also dropped. So RED applied at the frame level drops whole frames.

For the simulation results in Section 5, we used an exponential function to associate drop probabilities with queue lengths. We experimented with linear and inverse-square drop probabilities and found that they do not work well. Note that the RED that we use is different in details from the scheme proposed by Floyd and Jacobson [4]. They first estimate the average queue size using exponential averaging over 500 packet arrivals. This average queue size is compared against minimum and maximum queue thresholds. If the average is greater than the maximum queue threshold, the incoming packet (Floyd and Jacobson's description is for packet TCP) is dropped. If the estimated average queue size is between the minimum and maximum thresholds then a linear drop function modified to attempt to space drops at regular intervals is used.

Madhow [9] show that for TCP Tahoe without ATM cellification, to prevent throughput collapse due to losses occurring in the slow-start phase itself, β must be at least 1/3. For our studies, we simulated TCP with two-way traffic over a router and an ATM switch with the ATM switch output port being the only bottleneck. The simulations indicate that for "Reno with taildrop" β must be at least three to get decent throughput, while for "Reno with drop from front" the minimal value of β is closer to one (see Section 4 for throughput plots). For links with large bandwidth-delay products, this means that the buffer sizes must be large. The early-warning effect due to "Drop from Front" is then considerable.

An important mechanism that has been proposed to avoid throughput collapse is Floyd and Jacobson's [4] Random Early Detection (RED) and related mechanisms. We compared the perfomance of drop from front to variants of RED. A summary of RED is in Section 2 and detailed simulation results are in Section 5. We find that drop from front performs better than RED when both schemes drop cells without trying to drop all further cells from already damaged packets, i.e. without partial packet discard. When the schemes pay attention to the packet level, drop from front (with partial packet discard) performs almost as well as RED (with complete packet discard). However, drop from front has the advantage that the switch does not need to maintain a table of drop probabilities and does not have to know the traffic type being carried. This is because drop from front also reduces latencies for succesfully transmitted packets and hence is a sensible policy to use for delay sensitive non-feedback controlled traffic as well.

This reduction in latency has been shown by Yin and Hluchyj [13] who considered a "drop from front" scheme for a very different problem where none of the sources are feedback controlled. They found that drop from front results in shorter average delay in the buffer for eventually transmitted packets and recommended its use for time-constrained traffic. It is easily seen that this shorter average delay is due to the increased effective service rate when some packets at the front of the buffer are dropped instead of transmitted, and that the decrease in delay is roughly proportional to the fraction of packets dropped.

While decreased cell delay is roughly proportional to the loss rate, "Drop from Front" creates holes one buffer drain time further forward in the packet stream than "Taildrop", and the amount of earlier warning always is about one buffer drain time, independent of the loss rate. What we see here is an application of the idea that if a switch informs sources of congestion by modifying the packet stream, it should always attempt to make the modifications as far forward in the buffer as possible. If it does not matter which source gets the information, it is the most forward packet or cell that should be modified. Dropping is an extreme form of modifying, and in the case of TCP the only modification the source understands. The same idea implies that if "Forward Explicit Congestion Notification" (FECN) or "Backward Explicit Congestion Notification" (BECN) gets implemented, the decision to set the congestion bit should be made shortly before the cell is transmitted.

The remainder of this paper is organized as follows: Section 2 briefly reviews the salient background information regarding TCP, RED, frame dropping, etc. Section 3 describes the TCP over ATM system simulated in our performance studies. Section 4 discusses simulation results comparing drop from front to tail drop. Section 5 discusses simulation results comparing drop from front to RED. Conclusions are in Section 6.

variations. The performance of drop from front is also comparable to that obtained using variations of Random Early Detection (RED) and it outperforms RED when RED is implemented at the cell level.

In this paper, we study Wide Area Networks (WANs) with high bandwidth-delay products where the buffering at bottleneck links is at most (and often less) a small multiple of the bandwidth-delay product. In this situation, it is well known (see eg [14], [12], [15], [9]) that if a bottleneck buffer in the network is not large enough, the "over-reaction" of the TCP sources to congestion can cause link starvation and throughput collapse. This problem is even more serious in the ATM-part of a network than in the classical router based part of the network. Also, in this WAN environment, the throughput loss in TCP over ATM [11] due to transmission of cells from already damaged packets is only a secondary effect and a solution such as Partial Packet Discard [11] does not by itself give sufficient throughput improvements.

We propose "Drop from Front" as a partial solution to the problem of throughput collapse in networks where TCP represents a sizeable part of the load. Drop from Front can be used in conjunction with other strategies such as Partial Packet Discard. We will show that moving to a "Drop from Front" strategy considerably improves performance and allows use of smaller buffers than is possible with "Taildrop". While our analysis is for an ATM cell-buffer carrying only TCP traffic, we expect our conclusions to remain valid for packet-buffers in routers, and for buffers where TCP is only a sizeable part of the load.

TCP assumes packet losses to be symptomatic of congestion and TCP sources cut back their window sizes upon detection of packet loss [5]. The TCP fast retransmit mechanism, introduced by Van Jacobson, detects packet loss by counting duplicate acknowledgements and provides a means for quick detection of packet loss[1] without waiting for time-outs. These duplicate acknowledgements are received by the source when the destination detects missing packets in the stream delivered to it by the network. During congestion episodes when buffers are full, drop from front causes the destination to "see" missing packets in its stream approximately one buffer drain time earlier than would be the case under tail drop. The sources correspondingly receive earlier duplicate acknowledgements causing earlier reduction in window sizes.

This earlier reaction results in considerably higher throughput because of TCP's drastic reduction in window sizes upon detection of loss. Though the specific reduction dynamics depend on the TCP version, in all versions the window is at least halved upon loss detection (for Reno the window is halved for each packet loss, the proposed Vegas modification aims to only halve the window for the first packet loss in a congestion episode, and for Tahoe the window is set to 1 and then increased rapidly to half the value before loss). Halving the window causes the source to stop sending packets for approximately half a typical Round Trip Time. With drop from front, a few sources (with packets near the front at congestion onset) receive earlier congestion notification. The drastic rate reduction from these sources allows cells from other sources to successfully enter (and leave) the buffer. The shortened congestion episode with fewer sources losing packets greatly reduces or eliminates later overreaction by more sources.

This throughput-collapse due to the TCP over-reaction to packet loss decreases in severity if the bottleneck buffer is increased in size (see, for instance, the simulation results in Section 4). Let β denote the the normalized buffer size at a bottleneck link defined as the number of buffers divided by the product of the link speed and the "typical" roundtrip time outside the buffer. Lakshman and

[1]except when window sizes are extremely small

The Drop from Front Strategy in TCP and in TCP over ATM

T. V. Lakshman[†]Arnold Neidhardt[*] Teunis J. Ott[**]

† AT&T Bell Labs
ATM Networks Research Dept.
101 Crawfords Corner Road
Holmdel NJ 07733 USA

* Bellcore
331 Newman Springs Road
Red Bank, NJ 07701
USA

** Bellcore
445 South Street
Morristown, NJ 07962
USA

Abstract

This paper proposes the use of a *Drop from Front* scheme for improving TCP performance in high bandwidth-delay product networks. In particular, for "TCP over ATM" we compare the performance when drop from front is used at the output port of ATM switches with the performance under tail drop, its variations, and with variations of Random Early Detection (RED). In drop from front, when a cell arrives at a full buffer, the cell closest to being transmitted is dropped, thus creating space for the arriving cell. This causes duplicate acknowledgements to be sent one whole buffer drain time earlier than is the case under tail drop. These quicker duplicate acknowledgements cause TCP with Fast Retransmit to recognize losses faster and invoke congestion control actions earlier than would be the case under tail drop. This earlier reaction translates into considerable performance improvement. Hence, Drop from front successfully utilizes the ability of TCP with Fast Retransmit to quickly recognize and react to congestion information (at the third repeat acknowledgement, as opposed to time-out). Roughly, the earlier action by the sources causes the congestion not to grow quite as severe, which prevents later over-reaction by the sources, and thus increases throughput. Our simulations show that, for the same buffer size, drop from front results in considerably higher TCP throughput than tail drop, and for all but very small buffers even higher than tail drop combined with partial frame drop. Without partial frame drop, drop from front performs better than RED and with partial frame drop its performance is very close to that of RED with complete frame drop. Drop from front is also fairer than tail drop because it partially counteracts TCP's bias against connections with larger round trip times.

1 Introduction

In packetized communication, whenever a packet attempts to enter a full buffer, packet loss is inevitable. The prevalent custom is to discard the packet (or cell) attempting to enter the queue. Following Romanow and Floyd [11], and Floyd and Jacobson [4] we will call this "tail dropping". However, there are no compelling reasons why tail dropping should be the method of choice. We advocate use of a drop from front policy under which when a packet (or cell) arrives to a full buffer, the arriving packet is allowed in, with space being created by discarding the packet (or cell) at the front of the buffer. We show that for networks using TCP, the Internet transport protocol, a drop from front policy results in better performance than is the case under tail dropping and its