Connections with Multiple Congested Gateways in Packet-Switched Networks
Part 2: Two-way Traffic

Sally Floyd*
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720
floyd@ee.lbl.gov

December 13, 1991

Abstract

In a previous paper [F91] we explored the bias in TCP/IP networks against connections with multiple congested gateways, for networks with one-way traffic. Using simulations and a heuristic analysis, we showed that in a network with the window modification algorithm in 4.3 tahoe BSD TCP and with Random Drop or Drop Tail gateways, a longer connection with multiple congested gateways can receive unacceptably low throughput. However, we showed that in a network with no bias against connections with longer roundtrip times and with no bias against bursty traffic, a connection with multiple congested gateways can receive an acceptable level of throughput.

In this paper we show that the the addition of two-way traffic to our simulations results in compressed ACKs at the gateways, resulting in much more bursty traffic in the network. We use Priority gateways to isolate the effects of compressed ACKs in networks with two-way traffic; these Priority gateways give priority to small packets such as ACK packets, thereby allowing two-way traffic without compressed ACKs. In our simulations with two-way traffic and multiple congested gateways with FIFO service and Drop Tail or Random Drop packet-drop algorithms, the throughput for the connection with multiple congested gateways suffers significantly. This loss of throughput can be reduced either by the use of Priority gateways, which reduces the burstiness of the traffic, or by the use of Random Early Detection (RED) gateways, which are designed to avoid a bias against bursty traffic.

*This work was supported by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

1 Introduction

In this paper we investigate the throughput of connections in TCP/IP networks with two-way traffic and multiple congested gateways.

2 Simulator algorithms

Our simulator is briefly described in [F91], along with the algorithms used in these simulations.

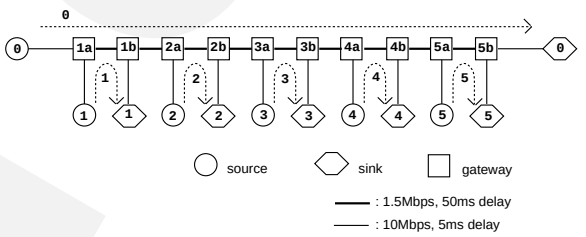

Figure 1: Simulation network with 5 congested gateways.

Figure 1 shows a simulation network for one-way traffic, with 6 connections, 10 gateways, and 5 congested gateways. The congested gateways in Figure 1 are gateways 1a, 2a, 3a, 4a, and 5a. The dotted lines show the connection paths; source i sends to sink i. Each connection has a maximum window just large enough so that, even when that connection is the only active connection, the network still occasionally drops packets.

For one-way traffic we use a family of simulation networks similar to Figure 1, where the number of congested gateways n ranges from 1 to 10. Figure 1 only shows the network for $n = 5$ congested gateways. For

Draft

a simulation network with n congested gateways for $n \geq 1$ there are $n + 1$ connections and $2n$ gateways. Connection 0 passes through multiple congested gateways, and connections 1 through n each pass through one congested gateway. Connection 0 is roughly $2n - 1$ times longer than the other n connections. In these simulations all connection paths have the same maximum bandwidth.

Definitions: Reduce-to-One, Increase-by-One. Some of the simulations in the paper use the Reduce-to-One window decrease algorithm and the Increase-by-One window increase algorithm from 4.3 tahoe BSD TCP [J88]. Our simulator does not use the 4.3-tahoe TCP code directly but we believe it is functionally identical. Briefly, there are two phases to the window-adjustment algorithm. In the slow-start phase the window is doubled each roundtrip time until it reaches a certain threshold. Reaching the threshold causes a transition to the congestion-avoidance phase where the window is increased by roughly one packet each roundtrip time. In this paper we call the increment algorithm used in the congestion-avoidance phase the Increase-by-One algorithm. Packet loss (a dropped packet) is treated as a "congestion experienced" signal. The source uses timeouts or "fast retransmit" to discover the loss (if four ACK packets acknowledging the same data packet are received, the source decides a packet has been dropped) and reacts by setting the transition threshold to half the current window, decreasing the window to one packet and entering the slow-start phase. In this paper we call this the Reduce-to-One algorithm. □

In order to achieve the highest throughput for longer connections, some of these simulations use the Fast Recovery algorithm in 4.3 reno BSD TCP, along with Selective Acknowledgements (or SACKs). In this paper the Fast Recovery algorithm implemented in 4.3 reno BSD TCP is called the Reduce-by-Half algorithm.

Definitions: Reduce-by-Half window decreases and Selective Acknowledgements. With the Reduce-by-Half window decrease algorithm, when a packet loss is detected by the "fast retransmission" algorithm the connection reduces its window by half. For the purposes of this paper, the important feature of the Reduce-by-Half window decrease algorithm is that with the "fast retransmission" algorithm, the source retransmits a packet and reduces the window by half, rather than reducing its window to one packet. For the simulations in this paper, we use Selective Acknowledgement sinks [JB88] with the Reduce-by-Half algorithm. With Selective Acknowledgement sinks, each ACK acknowledges not only the last sequential packet received for that connection, but also acknowledges all other (non-sequential) packets received. □

Some of the simulations in this section use the TCP Increase-by-One window-increase algorithm for the congestion-avoidance phase of the window-increase algorithm. As shown in [FJ91a], this algorithm has a bias against connections with longer roundtrip times. In order to eliminate this bias, some of our simulations use the Constant-Rate algorithm instead in the congestion-avoidance phase. We are *not* proposing the Constant-Rate algorithm for current networks; we simply are using the Constant-Rate algorithm to explore throughput in networks with a window-increase algorithm with no bias in favor of shorter-roundtrip-time connections.

Definitions: Constant-Rate window increases. In the Constant-Rate window-increase algorithm, in the congestion-avoidance phase each connection increases its window by roughly $a * r^2$ packets each roundtrip time, for some fixed constant a, and for r the calculated average roundtrip time. Using this algorithm, each connection increases its window by a pkts/sec in each second. For the simulations in this paper, we use $a = 4$. □

In this paper we examine networks with Drop Tail, Random Drop, and RED gateways. As shown in [FJ91a], simulations and measurement studies with Drop Tail gateways are vulnerable to traffic phase effects; small changes in network parameters can result in large changes in the performance of the network. In order to avoid these phase effects in networks with Drop Tail gateways, in this paper the simulator adds a small random component to the roundtrip time for each packet in simulations using Drop Tail gateways. (This is discussed in [FJ91a].) Normally, our simulator charges zero seconds for the time required to process packets at the nodes. In this paper each source node adds a random time uniformly distributed in [0, b], for $b \approx 5.3$ ms. the bottleneck service time of the network, to the time required by the source node to process each ACK packet in the simulations with Drop Tail gateways. This is not intended to model any assumptions about realistic network behavior, but to eliminate problems with traffic phase effects. With this added random component the simulations with Drop Tail gateways give similar results to the simulations with Random Drop gateways.

To avoid the bias against bursty traffic common to Random Drop and to Drop Tail gateways, we also examine performance in simulations with Random Early Detection (RED) gateways, a modified version of Random Drop gateways that detect incipient congestion. RED gateways maintain an upper bound on the average queue size at the gateway.

Definitions: RED gateways. With our implementation of RED gateways [FJ91c], the gateway computes the average size for each queue using an exponential weighted moving average. When the average queue size exceeds a certain threshold, indicating incipient

Draft

congestion, the gateway randomly chooses a packet to drop and increases the threshold. As more packets arrive at the gateway, the threshold slowly decreases to its previous value. The gateway chooses a packet to drop by choosing a random number n in the interval 1 to $range$, where $range$ is a variable parameter of the gateway. The gateway drops the nth packet to arrive at the gateway. RED gateways are described in more detail in a paper currently in progress [FJ91c]. □

One advantage of RED gateways is that, unlike Drop Tail and Random Drop gateways, RED gateways do not have a bias against bursty traffic. The bias of Drop Tail and of Random Drop gateways against bursty traffic and the correction of this bias in RED gateways are described in [FJ91a]. With Drop Tail or Random Drop gateways, the more bursty the traffic, the more likely it is that the queue will overflow and the Drop Tail or Random Drop gateway will drop a packet. This is because a burst of packets results in a temporary increase in the queue size at the gateway. With RED gateways the detection of congestion depends on the average queue size, not on the maximum queue size. Thus with RED gateways bursty traffic is less likely to result in the detection of congestion. With a RED gateway even when bursty traffic results in the detection of congestion at the gateway, the mechanism for dropping packets ensures that the bursty connection does not have a disproportionate probability of having a packet dropped. □

For the RED simulations in this paper the maximum queue size is 60 packets, and the RED gateways drop packets when the average queue size is between 10 and 20 packets. (The range from 10 to 20 packets for the average queue size for RED gateways is somewhat arbitrary; the optimum average queue size is still a question for further research.) For the two-way traffic simulations with Random Drop and Drop Tail gateways, the maximum queue size is 20 packets (I have to rerun these simulations for a maximum queue of 60 packets). The maximum windows for connections 1 to n are set sufficiently large to force occasional packet drops even in the absence of traffic from connection 0.

3 Simulations with two-way traffic

The simulations in Section 2 were of a network with one-way traffic. In this section, we explore simulations with two-way traffic. As discussed in [WRM91] and [ZC91], two-way traffic introduces the added complication of ACK-compression, caused by small ACK packets being queued at a congested gateway with no interleaving larger data packets. This ACK-compression increases the burstiness of the traffic, causing substantial problems for networks with Random Drop (or Drop Tail) gateways. We show that with Random Drop gateways, the addition of two-way traffic causes a substantial loss in throughput for the longer connection. With RED gateways, with their increased ability to accomodate bursty traffic, the addition of two-way traffic causes only a small change in the behavior of the network. We use gateways with Priority service, which give priority to small packets, to compare the throughput in networks with two-way traffic with and without compressed ACKs.

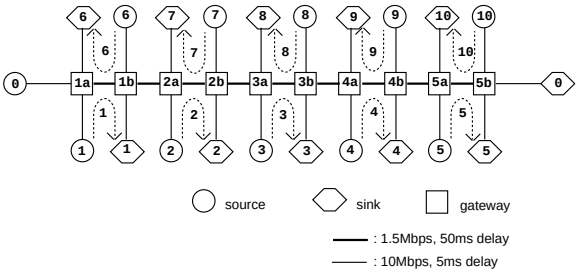

Figure 2: Simulation network with two-way traffic.

Figure 2 shows the simulation network with two-way traffic and with n congested forward gateways, for $n = 5$. Connection 0 goes from source 0 to sink 0. Gateways $1a$ to $5a$ are congested in connection 0's forward direction, and gateways $1b$ to $5b$ are congested in the reverse direction. With this two-way traffic, connection 0 suffers from significant ACK-compression. All of the connections in this network can experience compressed ACKs, but because connection 0 passes through multiple congested gateways, the problem of compressed ACKs is more severe for connection 0. All of the simulations in this section use Constant-Rate window increases and Reduce-by-Half decreases.

Draft

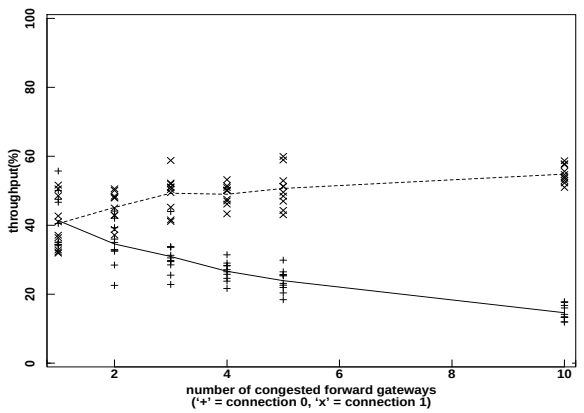

Figure 3: RED gateways, two-way traffic, FIFO service.

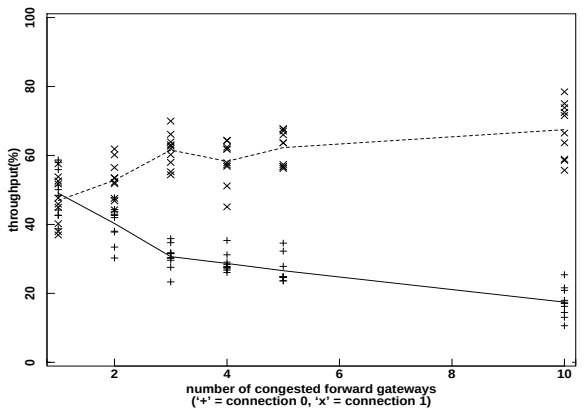

Figure 4: RED gateways, two-way traffic, Priority service.

Simulations with two-way traffic and RED gateways, with Constant-Rate increases and Reduce-by-Half decreases.

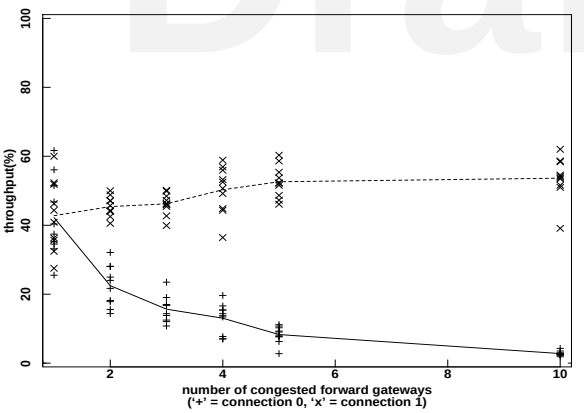

Figure 5: Random-Drop gateways, two-way traffic, FIFO service.

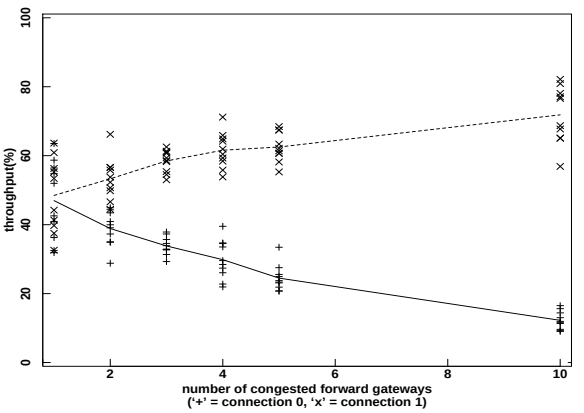

Figure 6: Random-Drop gateways, two-way traffic, Priority service.

Simulations with two-way traffic and Random-Drop gateways, with Constant-Rate increases and Reduce-by-Half decreases.

Draft

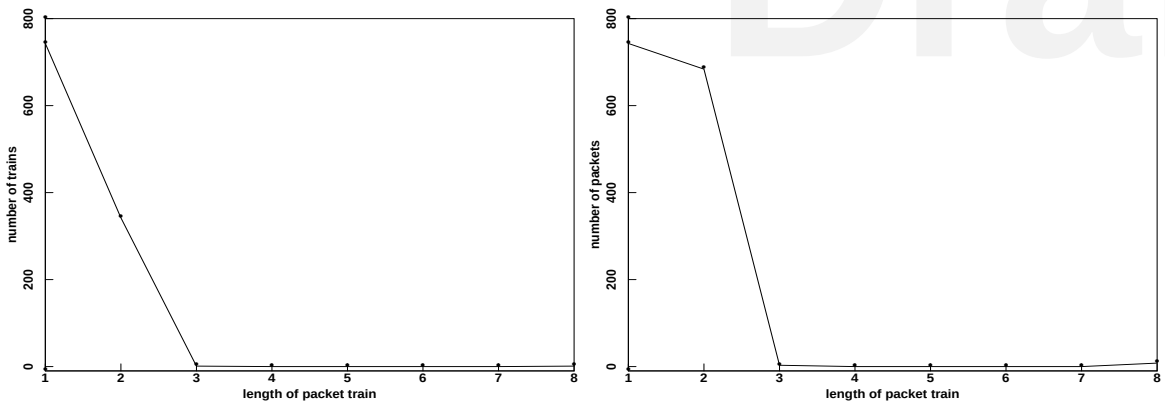

Figure 7: Packet trains with one-way traffic.

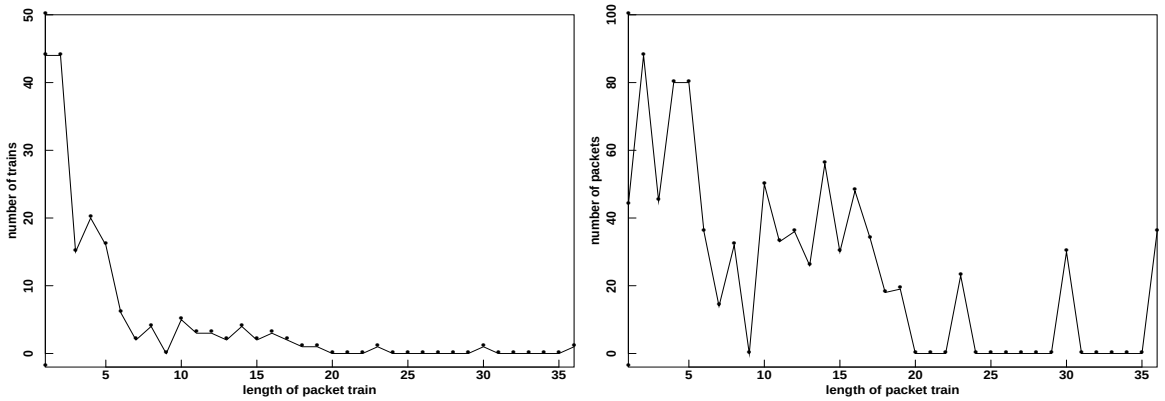

Figure 8: Packet trains with two-way traffic, FIFO service.

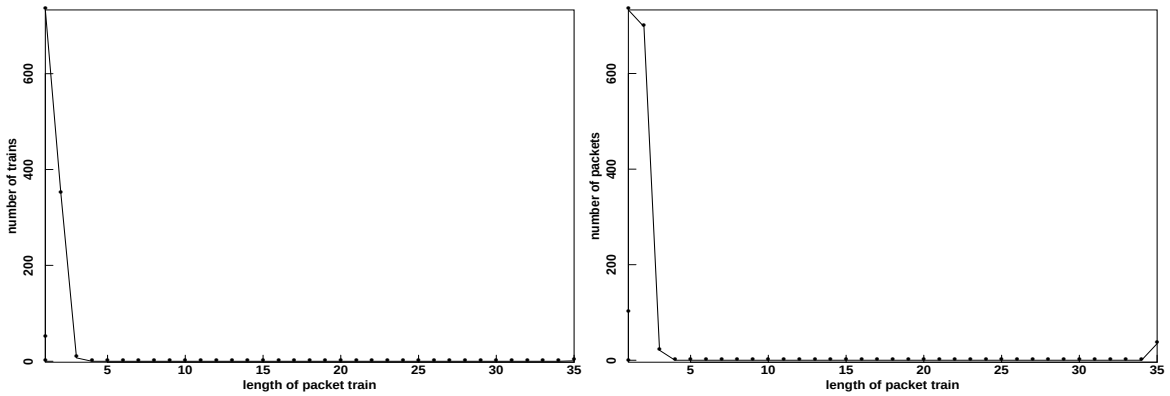

Figure 9: Packet trains with two-way traffic, Priority service.

Draft

Figures 3 and 5 show the results of simulations with the network in Figure 2. The simulations in Figure 3 use RED gateways. These differ from the simulations in Figure 6 in Part 1 only with the addition of traffic in the reverse direction. For the simulations in Figure 3, connections 0 and 1 both receive somewhat reduced throughput, due to the effects of the two-way traffic, but the throughput for connection 0 is still substantial. For the simulations in this section, both the RED and the Random-Drop gateways measure their queues in packets rather than in bytes. Because of this, the addition of two-way traffic adds significantly to the congestion in the network, If our gateways measured queue-length in bytes rather than in packets, this would reduce the impact of two-way traffic.

The simulations in Figure 5 use Random Drop gateways, and differ from the simulations in Figure 6 in Part 1 only with the addition of traffic in the reverse direction. For the simulations in Figure 5, the throughput for connection 0 is significantly reduced by the addition of two-way traffic, even for a small number of congested gateways. Because of the compressed ACKs for connection 0, the pattern of data packets transmitted by connection 0 is fairly bursty. With Random Drop or Drop Tail gateways, this burstiness increases connection 0's chances of having packets dropped at the gateways. With RED gateways, which are designed to accomodate bursty traffic as much as possible, this burstiness has less of an effect on network behavior.

3.1 Packet trains with two-way traffic

Definitions: packet train. For the network in Figure 1, the bottleneck service time is $b = 5.33$ ms., and it takes a source node $s = 0.8$ ms. to transmit a data packet to the gateway. With one-way traffic, ACK packets arrive at the source node at least b ms. apart, and the source node therefore sends data packets at least b ms. apart, in the absence of window increases. However, the source node is capable of sending data packets at s-ms. intervals. This can occur as a result of a window increase, or as a result of compressed ACKs that arrived at the source node close together. For the purposes of this paper, we define a *packet train* as a sequence of packets that leave the source node at s-ms. intervals, faster than the bottleneck service rate. □

Figure 7 shows the packet train lengths for connection 0 for the first 50 seconds of a simulation similar to Figure 6 in Part 1, with one-way traffic and 10 congested gateways. There are many packet trains of length 1, with single packets, and also many packet trains of length 2, which occur each time connection 0 increases its window by one packet. There is also one packet train of length 3, and one packet train of length 8. These result from the window opening up after the Reduce-by-Half window decrease algorithm.

With the Reduce-by-Half window decrease algorithm, a connection effectively waits half a roundtrip time after retransmitting a dropped packet, and reduces its window to half of its previous value. If only one packet has been dropped, then the data packets that are transmitted are all clocked by incoming ACK packets. However, due to details of the Reduce-by-Half algorithm, if many packets have been dropped in one roundtrip time, then the Reduce-by-Half window decrease algorithm might open its window to half of its previous value all at once when the ACK packet for the retransmitted packet is received. This results in a burst of packets sent by the source node. Connection 0 in Figure 2 has a large maximum window, and therefore is likely to have many packet drops during the initial slow-start phase of doubling the window each roundtrip time. This accounts for the long packet train in Figure 7 (and Figure 9).

Figure 8 shows the packet train lengths for connection 0 for the first 50 seconds of the simulation as in Figure 3, with two-way traffic and 10 congested forward gateways. There are many packet trains with from 10 to 20 packets, and a few longer packet trains. The lefthand chart of Figure 8 shows the number of packet trains of length m, and the righthand chart shows the number of packets in packet trains of length m. As the righthand chart shows, only a small fraction of the packets are in packet trains of length one or two. Almost all of the packet trains of length greater than two are caused by compressed ACKs.

Two ACK packets are compressed when the first ACK arrives at a gateway to a queue, and the second ACK packet arrives before the first ACK has been transmitted and also before any other intervening packets. Once two ACK packets have been compressed, it is fairly difficult for them to become separated again; another packet has to arrive at some succeeding gateway in the short time between the arrival of the first ACK packet and the arrival of the second ACK packet. Therefore, as the number of gateways with occasional queues increases, the problem of compressed ACKs also increases.

3.2 Priority gateways

One way to demonstrate the effects of compressed ACKs is to run simulations with gateways that give priority to small packets, and to compare these results to those from simulations with gateways with FIFO service.

Definitions: Priority gateways. For gateways with Priority service, we defined *small* packets as being at

Draft

most 50 bytes, and we defined *large* packets larger than 50 bytes. In our simulations, all data packets are defined large, and all ACK packets are defined as small. The Priority gateway transmits packets using FIFO service, with one exception. After a Priority gateway sends a large n-byte packet, then if there are any small packets, the Priority gateway sends at most n bytes of small packets before proceeding with the next large packet in the queue. The assumption is that after the current large packet is served, a Priority gateway will generally send all outstanding small packets before sending the next large packet. Thus, small packets that arrive at the gateway queue when the server is busy will usually be transmitted after the gateway finishes transmitting the current large packet. Large packets will have their waiting time in the queue increased by at most a factor of two. Priority gateways are easily implemented, and don't require separate state for each connection. For our simulator, either Priority or FIFO service can be chosen with either Drop Tail, Random Drop, or RED gateways. □

Compressed ACKs are much less likely to occur with Priority service than with FIFO service. With Priority service, each time an ACK packet arrives at a gateway to a busy server, the ACK packet waits at least until the current packet has been served before it can be transmitted. Thus, each Priority gateway adds a certain jitter to the relative timing of the ACK packets. This is quite different from the behavior of FIFO service, where each ACK packet might wait a substantial time in a queue.

Figure 4 shows the results of simulations with RED gateways with Priority service. Connection 0's throughput is somewhat better in Figure 4 than it was in Figure 3. Thus, we can see that the compressed ACKs in Figure 3 did have a (small) effect on throughput in the network. Figure 9 shows the lengths of packet trains for connection 0 for a 50-second simulation as in Figure 4. With Priority service and two-way traffic, almost all packet trains are of length one or two. The one packet train of length 35 is the result of the Reduce-by-Half window decrease algorithm, as explained above.

Figure 6 shows the result of simulations with Random Drop gateways with Priority service. Because the use of Priority service decreases the burstiness of the traffic, the use of Priority service increases the throughput for connection 0 significantly, with Random-Drop gateways. Because the gateways in these simulations use queues measured in packets rather than in bytes, the addition of two-way traffic still significantly increases the congestion in the network, even with Priority service. This accounts for the difference in performance from Figure 6 from Part 1 with one-way traffic, and Figure 6 with two-way traffic with Priority gateways.

The main conclusions of this section are that with Random Drop (or Drop Tail) gateways, the introduction of two-way traffic can lead to a significant increase in burstiness for traffic with multiple congested gateways, and this increase in burstiness can result in a decrease in throughput. For gateways such as RED gateways that avoid a bias against bursty traffic, the effects of an introduction of two-way traffic are much more moderate. Because Priority gateways allow us to look at networks with two-way traffic but without compressed ACKs, Priority gateways are a useful tool for exploring the effects of compressed ACKs in networks.

We do not necessarily propose implementing Priority gateways in current networks. For a network with Priority gateways, extra care would have to be given to the problems caused by gateways transmitting packets from one connection out-of-order. Packet-switched networks do not guarantee in-order delivery of packets in any case, but the problems of out-of-order packets could be intensified by the use of Priority gateways. This could occur, for example, when some data packets from a connection are small, and other data packets from the same connection are large. (This does not occur in our simulations, because in our simulations all data packets are defined as large, and all ACK packets are defined as small.) At the moment, we are using Priority gateways simply to examine the effects of compressed ACKs at the gateway.

4 Conclusions and Future Work

For our simulations with multiple congested gateways and one-way traffic, the performance of Random Drop gateways and of RED gateways are fairly similar. For our simulations with two-way traffic, however, the traffic for the connection with multiple congested gateways is quite bursty, due to compressed ACKs. In this case, there is a significant difference in performance between Random Drop and RED gateways. Because of their bias against bursty traffic (a bias shared by Drop Tail gateways), the use of Random Drop gateways results in reduced throughput for the connection with multiple congested gateways. For RED gateways, which are designed to accomodate bursty traffic, the introduction of two-way traffic, and the corresponding increase of burstiness, does not result in a significant reduction in throughput for the connection with multiple congested gateways.

In order to explore the effect of compressed ACKs in simulations with two-way traffic, we have introduced gateways with Priority service, which give priority to smaller packets such as ACK packets. With the traditional FIFO service, ACK packets arriving at a congested gateway wait their turn in the queue, and there-

Draft

fore it is possible to two successive ACK packets to be "compressed" in the queue. With Priority service, after each data packet is transmitted, (almost all) waiting ACK packets are transmitted before the next data packet in the queue. In this way, with Priority service compressed ACKs are avoided. We show that with Random Drop gateways the throughput for a connection with multiple congested gateways is acceptable in simulations with two-way traffic with Priority service (and no compressed ACKs), but that the throughput is not acceptable in simulations with two-way traffic with FIFO service in the gateway (with compressed ACKs). It is not the two-way traffic itself that degrades performance for the connection with multiple congested gateways, but specifically the compressed ACKs, with the resulting increased burstiness of the traffic.

References

[BDSY88] Bacon, D,. Dupuy, A., Schwartz, J., and Yemimi, Y., "Nest: a Network Simulation and Prototyping Tool", *Proceedings of Winter 1988 Usenix Conference*, 1988, pp. 17-78.

[BJ81] Bharath-Kumar, K., and Jeffrey, J., *A New Approach to Performance-Oriented Flow Control*, IEEE Transactions on Communications, Vol.COM-29 N.4, April 1981.

[CDJM91] Cáceres, R., Danzig, P., Jamin, S., and Mitzel, D., "Characteristics of Wide-Area TCP/IP Conversations", to appear in SIGCOMM "91.

[DKS90] Demers, A., Keshav, S., and Shenker, S., "Analysis and Simulation of a Fair Queueing Algorithm", Internetworking: Research and Experience, Vol. 1, 1990, p. 3-26.

[F91] Floyd, S., *Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic*, submitted to Computer Communication Review.

[FJ91a] Floyd, S., and Jacobson, V., *Traffic Phase Effects in Packet-Switched Gateways*, Computer Communication Review, V.21 N.2, April 1991.

[FJ91c] Floyd, S., and Jacobson, V., *Random Early Detection gateways for congestion avoidance*, in preparation.

[HG86] Hahne, E., and Gallager, R., *Round Robin Scheduling for Fair Flow Control in Data Communications Networks*, IEEE International Conference on Communications, June, 1986.

[J88] Jacobson, V., *Congestion Avoidance and Control*, Proceedings of SIGCOMM '88, August 1988.

[JB88] Jacobson, V., and Braden, R., "RFC 1072: TCP Extensions for Long-Delay Paths", October 1988.

[JCH84] Jain, R., Chiu, D.M., and Hawe, W., "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems." DEC TR-301, Littleton, MA: Digital Equipment Corporation.

[K88] Keshav, S., "REAL: a Network Simulator", *Report 88/472*, Computer Science Department, University of California at Berkeley, Berkeley, California, 1988.

[M90] Mankin, A., "A Measurement Study of Random Drop and Gateway Interaction", 1990.

[M90a] Mankin, A., *Random Drop Congestion Control*, Proceedings of SIGCOMM 90, Sept. 1990.

[MS90] Mitra, D. and Seery, J., *Dynamic Adaptive Windows for High Speed Data Networks: Theory and Simulations*, AT&T Bell Laboratories report, April 1990.

[RJ90] Ramakrishnan, K.K., and Jain, Raj, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks", ACM Transactions on Computer Systems, V. 8, N. 2, p. 158-181, 1990.

[R83] Ross, S., Stochastic Processes, John Wiley & Sons, 1983.

[WRM91] Wilder, R., Ramakrishnan, K.K., and Mankin, A., "Dynamics of Congestion Control and Avoidance in Two-Way Traffic in an OSI Testbed", SIGCOMM '91, April 1991, p.43-58.

[ZC91] Zhang, L., and Clark, D., "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic", to appear in SIGCOMM '91.