Traffic Phase Effects in Packet-Switched Gateways

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

Abstract

Much of the traffic in existing packet networks is highly periodic, either because of periodic sources (e.g., real time speech or video, rate control) or because window flow control protocols have a periodic cycle equal to the connection roundtrip time (e.g., a network-bandwidth limited TCP bulk data transfer). Control theory suggests that this periodicity can resonate (i.e., have a strong, non-linear interaction) with deterministic estimation or control algorithms in network gateways.[1] In this paper we define the notion of *traffic phase* in a packet-switched network and describe how phase differences between competing traffic streams can be the dominant factor in relative throughput. Drop Tail gateways in a TCP/IP network with strongly periodic traffic can result in systematic discrimination against some connections. We demonstrate this behavior with both simulations and theoretical analysis. This discrimination can be eliminated with the addition of appropriate randomization to the network. In particular, analysis suggests that simply coding a gateway to drop a random packet from its queue (rather than the tail) on overflow is often sufficient.

We do not claim that Random Drop gateways solve all of the problems of Drop Tail gateways. Biases against bursty traffic and long roundtrip time connections are shared by both Drop Tail and Random Drop gateways. Correcting the bursty traffic bias has led us to investigate a different kind of randomized gateway algorithm that operates on the traffic stream, rather than on the queue. Preliminary results show that the Random Early Detection gateway, a newly developed gateway congestion avoidance algorithm, corrects this bias against bursty traffic. The roundtrip time bias (at least in TCP/IP networks) results from the TCP window increase algorithm, not from the gateway dropping policy, and we briefly discuss changes to the window increase algorithm that could eliminate this bias.

1 Introduction

In this first part of this paper we present fundamental problems resulting from the interaction between deterministic gateway algorithms and highly periodic network traffic. We define the notion of traffic phase for periodic traffic and show that phase effects can result in network performance biases. We show further that gateways with appropriate randomization, such as Random Drop gateways, can eliminate this bias. In the second part of this paper we discuss some of the advantages and shortcomings of Random Drop gateways that have been reported in the literature.

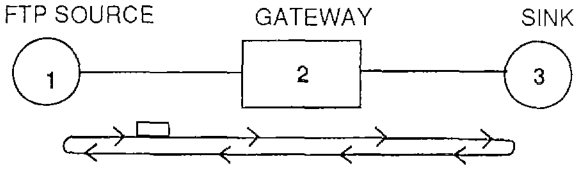

Figure 1: Periodic traffic.

Gateway algorithms for congestion control and avoidance are frequently developed assuming that incoming traffic is 'random' (according to some probability distribution). However, much real network traffic, such as bulk data transfer shown in Figure 1, has a strongly periodic structure. For a particular connection the number of outstanding packets is controlled by the current window. When the sink receives a data packet it immediately sends an acknowledgment (ACK) packet in response and when the source receives an ACK it immediately transmits another data packet. Thus the

*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] While gateway congestion control algorithms are almost nonexistent at present, there is one (particularly poorly behaved) algorithm in almost universal use: If a gateway's output queue is full it deterministically drops a newly arriving packet. In this paper, we refer to this algorithm as "Drop Tail" and examine its (mis-)behavior in some detail.

roundtrip time of the connection is the traffic "period". (This roundtrip time may vary as queueing delays vary.)

Most current network traffic is either bulk data transfer (i.e., total transfer is large compared to the bandwidth-delay product and throughput is limited by network bandwidth) or interactive (i.e., transfers small compared to bandwidth-delay product and/or infrequent relative to the roundtrip time). In this paper we refer to the former as "FTP traffic" and are concerned with its periodic structure. We refer to interactive traffic as "telnet traffic" and use Poisson sources to model it. By *random traffic* we mean traffic sent at a random time from a telnet source.

Consider FTP traffic with a single bottleneck gateway and a backlog at the bottleneck.[2] When all of the packets in one direction are the same size, output packet completions occur at a fixed frequency (determined by the time to transmit a packet on the output line).

For example, the following is a schematic of the packet flow in figure 1: Packets leaving the gateway are all the same size and occupy b seconds of bottleneck link time. The source-sink-source "pipe" is completely full (i.e., if the roundtrip time is r, there are $\lfloor r/b \rfloor$ packets in transit). A packet that departs the gateway at time D results in a new packet arrival at time $D + r$ (the time to take one trip around the loop). But queue lengths are incremented and decremented only at packet ends. Thus there will be a gap of $\phi = r \bmod b$ between the departure of a packet from the gateway queue and the arrival of the next packet at the queue. We call this gap the *phase* of the conversation relative to this gateway. (Phase is defined formally in section 2.2 and the general traffic case is illustrated in figures 6, 7, and 8.)

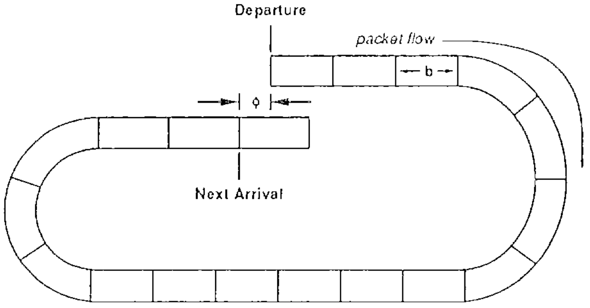

Figure 2: The phase (ϕ) of a simple packet stream.

Since the gateway queue length decrements by one for departures and increments by one for arrivals, the phase is simply the (average) time this particular connection leaves a vacancy in the queue. If, for example, the connection has filled the gateway queue, the probability that a (random) telnet packet will successfully grab the one vacancy created by a departure (thereby forcing the gateway to drop the next packet that arrives for the bulk-data connection) is simply ϕ/b times the telnet traffic intensity. Since ϕ is a function of the physical propagation time, r, small topology or conversation endpoint changes can make the gateway completely shut out telnets ($\phi \approx 0$) or always give them preference ($\phi \approx b$). (Section 2.6 describes this in detail.)

Phase effects are more common than the example above suggests. Whenever the gateway congestion management mechanism is driven by backlog, phase effects can cause a significant bias. In this paper, we concentrate on networks with TCP congestion management (where each source executes the 4.3BSD TCP congestion control algorithm described in [J88]) and Drop Tail gateways. A longer version of this paper [FJ91, in preparation] demonstrates phase effects in an ISO-IP/TP4 network using DECbit congestion management [RJ90].

Another type of periodic traffic, rate controlled or real-time sources, exhibits phase effects similar to those described in this paper. These effects have been described in the digital teletraffic literature and, more recently, in a general packet-switching context. For example, [RW90] discusses the periodicity of packetized voice traffic where each voice source alternates between talk spurts and silences. A small random number of packets (mean 22) is transmitted for each talk spurt and these packets arrive at the multiplexer separated by a fixed time interval. For the model in this paper, the packet stream from many conversations is multiplexed on a slotted channel with a finite buffer. The authors show that when a packet from a voice spurt encounters a full buffer there is a high probability that the next packet from that voice spurt also encounters a full buffer. Because packets arriving at a full buffer are dropped, this results in successive packet losses for a single voice spurt. In fact, with this model [L89] has shown that *any* position-based strategy of dropping packets results in successive packet losses for one voice spurt. [L89] shows that even though the beginning and endings of talk spurts break up the periodic pattern of packet drops, the periodic pattern is quickly reestablished. [RW90] shows that a "random drop" strategy works well in distributing the packet losses across the active conversations. [L90] describes in detail the periodicity of the packet queues for this model.

Section 2.1 contains basic simulations showing bias due to traffic phase in networks with Drop Tail gateways and 2.2 gives an analysis of this behavior. Section 2.3 discusses the extent to which this behavior would

[2] Since most current topologies consist of a high-speed LAN gatewayed onto a much lower speed WAN, this is a fair approximation of reality: The bottleneck is the LAN-to-WAN transition and, since current gateways rarely do congestion avoidance, it will probably have a sizable queue.

persist in a network with some random traffic. Section 2.5 shows the success of the Random Drop algorithm in eliminating this discriminatory behavior. Section 2.6 examines discrimination between telnet connections and FTP connections in networks with Drop Tail gateways.

The second half of the paper addresses some of the criticisms of Random Drop gateways. Section 3.1 summarizes previous research. Section 3.2 discusses the bias shared by Random Drop and Drop Tail gateways against bursty traffic. Section 3.3 discusses TCP's bias against connections with longer roundtrip times. Section 4 discusses areas for future research and presents conclusions.

2 Traffic phase effects

2.1 Simulations of phase effects

In this section we give the results of simulations showing the discriminatory behavior of a network with Drop Tail gateways and TCP congestion control. These simulations are of the network in Figure 3, with two FTP connections, a Drop Tail gateway and a shared sink. The roundtrip time for node 2 packets is changed slightly for each new simulation, while the roundtrip time for node 1 packets is kept constant. In simulations where the two connections have the same roundtrip time, they get equal throughput. However, when the two roundtrip times differ, the network preferentially drops packets from one of the two connections and its throughput suffers. This behavior is a function of the relative phase of the two connections and changes with small changes to the propagation time of any link.

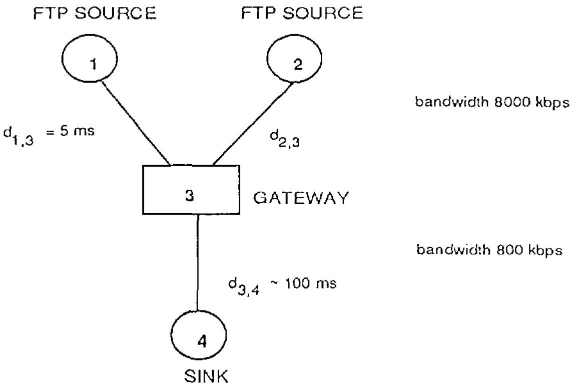

Figure 3: Simulation network.

Our simulator is a version of the REAL simulator [K88] (which is built on Columbia's Nest simulation package [BDSY88]) with extensive modifications and bug fixes made by Steven McCanne at LBL. The gateways use FIFO queueing, and in this section's simulation, always use Drop Tail on queue overflow. FTP sources always have a packet to send and always send a maximal-sized packet as soon as the window allows them to do so. A sink immediately sends an ACK packet when it receives a data packet.

Source and sink nodes implement a congestion control algorithm similar to that in 4.3-tahoe BSD TCP [J88].[3] Briefly, there are two phases to the window-adjustment algorithm. In slow-start phase the window is doubled each roundtrip time until it reaches a certain threshold. Reaching the threshold causes a transition to congestion-avoidance phase where the window is increased by roughly one packet each roundtrip time. Packet loss (a dropped packet) is treated as a "congestion experienced" signal. The source uses "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, then decreases the window to one and enters slow-start phase.

The essential characteristic of the network in Figure 3 is that two fast lines are feeding into one slower line. Our simulations use 1000-byte FTP packets and 40-byte ACK packets. The gateway buffer in Figure 3 has a capacity of 15 packets. With the parameters in Figure 3, for $d_{3,4} = 100$ ms., packets from node 1 have a roundtrip time of 221.44 ms. in the absence of queues. The gateway takes 10 ms. to transmit an FTP packet so a window of 23 packets is sufficient to "fill the pipe". (This means that when a connection has a window greater than 23 packets, there must be at least one packet in the gateway queue.)

Figure 4 gives the results of simulations where each source has a maximum window of 32 packets. Thus, each source is prepared to use all of the available bandwidth. Each dot on the graph is the result from one 100 sec. simulation, run with a different value of $d_{2,3}$, the propagation delay on the edge from node 2 to gateway 3. The x-axis gives the ratio between node 2's and node 1's roundtrip time for each simulation. The y-axis gives node 1's average throughput for the second 50-second interval in each simulation, measured as the percentage of the maximum possible throughput through the gateway (for all simulations, steady state was reached early in the first 50 seconds).

Figure 4 shows that this configuration is highly biased: For most values of Node 2's link delay, Node 1 gets 90% of the available bandwidth. But some Node 2 delay values cause the Node 1 bandwidth share to drop

[3] Our simulator does not use the 4.3-tahoe TCP code directly but we believe it is functionally identical.

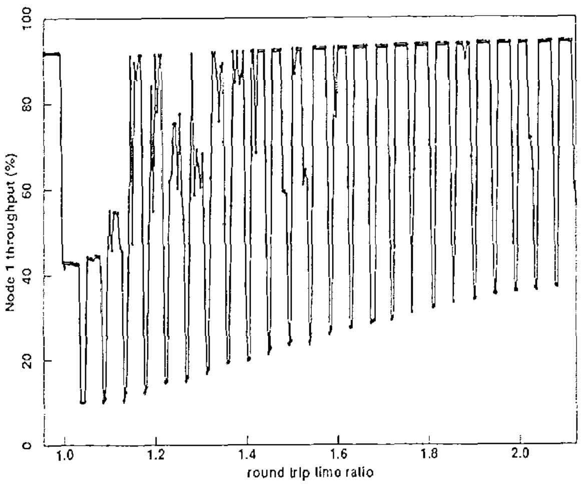

Figure 4: Node 1's average throughput vs. node 2's roundtrip time, for $d_{3,4} = 100$ ms.

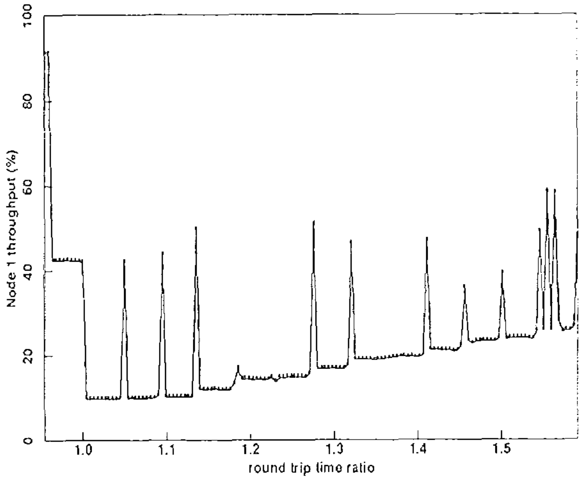

Figure 5: Node 1's throughput vs. node 2's roundtrip time, for $d_{3,4} = 103.5$ ms.

to only 10–20% and those values appear to be regularly spaced. As the next section explains in more detail, this behavior results from the precise timing of the packet arrivals at the gateway. The gateway takes 10 ms. to transmit one FTP packet, therefore during congestion packets leave the gateway every 10 ms. The structure in the graph (the space between the large throughput dips) corresponds to a 10 ms. change in node 2's roundtrip time.

Figure 5 shows the result of making a small (4%) change in the delay of the shared link, $d_{3,4}$. Note that there is still a huge bias but its character has changed completely: Now Node 2 gets 80–90% of the bandwidth at almost any value of its link delay and the bandwidth reversal peaks are much narrower (though still spaced at 10ms. intervals).

It is not necessary to have large maximum windows or few connections to get the bias in Figure 4. For example, we ran simulations of a network similar to that in Figure 3 but with eight FTP sources, each with a maximum window of 8 packets. For four of the sources the line from the source to the gateway had a delay of 5 ms. For the other four, the line from the source to the gateway had a delay varied over the same range as $d_{2,3}$ above. With $d_{3,4} = 100$ ms., the pattern is essentially identical to Figure 4. In some simulations, the first four connections receive most of the throughput, and in other simulations the second four connections receive most of the throughput.

2.2 Analysis of phase effects

In this section, we present a model for the timing of packet arrivals at the bottleneck gateway, define the notion of traffic phase, and describe the pattern of packet arrivals at the gateway for the network in Figure 3. Finally, this description is used to explain the simulation results in the previous section.

Let packets from node 1 have roundtrip time r_1 in the absence of queues. This means that, in the absence of queues, when node 1 transmits an FTP packet the ACK packet is received back at node 1 after r_1 seconds. For the network in Figure 3, the only possible nonempty queue is the output queue for the line from gateway 3 to node 4. Assume that the gateway begins transmission of an FTP packet from node 1 at time t. When this FTP packet arrives at the sink, the sink immediately sends an ACK packet, and when the ACK packet arrives at node 1, node 1 immediately sends another FTP packet. This new FTP packet arrives at the gateway queue exactly r_1 seconds after the old FTP packet left the gateway. (For this discussion, assume that when the last bit of a packet arrives at the gateway it is immediately added to the output queue and leaves the queue when the gateway begins transmission of the packet.) Thus, in the absence of window decreases, exactly r_1 seconds after a node 1 FTP packet leaves the gateway, another node 1 FTP packet arrives at the gateway.

Definitions: r_1, r_2, b, $maxqueue$, s. Packets from node 1 have roundtrip time r_1, and packets from node 2 have roundtrip time r_2, in the absence of queues. The gateway takes $b = bottleneck$ seconds to transmit an FTP packet, and has maximum queue size $maxqueue$. Node 1 and node 2 each take s seconds to transmit a packet on the line to the gateway. □

Defining the model: We give a model of gateway behavior for the network in Figure 3. The model starts with the time when the gateway queue is occasionally full, but not yet overflowing. Assume that initially the window for each connection is fixed (this period of fixed windows could be thought of as lasting less than one roundtrip time) then each connection is allowed to increase its window at most once. Assume that the gateway queue is never empty and that all FTP packets are of the same size. We are not concerned with how the windows reach their initial sizes.

The model specifies that a source can only increase its window immediately after the arrival of an ACK packet. When the source receives this ACK packet, it immediately transmits an FTP data packet and increases the current window by one. When the output line becomes free s seconds later, it sends a second data packet. Without the additional packet, the gateway queue occasionally would have reached size $maxqueue$. Because of the additional packet, the queue eventually fills, some packet arrives at a full queue and is dropped. The pattern of packet arrivals at the gateway determines which packet will be dropped. □

Definitions: phases t_1, t_2. Now we describe the timing of packet arrivals at the gateway. Every b seconds the gateway processes a packet and decrements the output queue by one. (This number b equals the size of the FTP data packet divided by the speed of the output line.) Using queueing theory terminology, a new *service interval* begins each time the gateway processes a new packet. After the gateway begins transmission of a packet from node 1, another FTP packet from node 1 arrives at the gateway exactly r_1 seconds later. This new packet arrives exactly $t_1 = r_1 \bmod b$ seconds after the beginning of some service interval. (We define $a \bmod b$ as the positive remainder from dividing a by b.) Similarly, when the gateway transmits a node 2 packet, another node 2 packet arrives at the gateway after r_2 seconds, or $t_2 = r_2 \bmod b$ seconds after the beginning of some service interval. The time intervals t_1 and t_2 give the *phases* of the two connections. Notice that if $t_1 > t_2$, then when a node 1 and a node 2 packet arrive at the gateway in the same service interval, the node 1 packet arrives at the gateway after the node 2 packet. □

In this section we give the intuition explaining the behavior of the model; [FJ91] gives formal proofs. We discuss three cases, when $r_1 = r_2$, when r_1 is slightly greater than r_2, and when r_2 is slightly greater than r_1. These are shown in Figures 6, 7, and 8. Node 1 has the same roundtrip time in all three figures, and the same value for t_1. For node 2, however, the roundtrip time is different in the three figures, and the value for t_2 changes.

Case 1: In this model, when $r_1 = r_2$, then $t_1 = t_2$, and a new packet arrives at the gateway every b seconds. The order of the packet arrivals depends on the order of the packet departures one roundtrip time earlier. Each new arrival increases the gateway queue to $maxqueue$. The queue is decremented every b seconds, at the end of each service interval. Line D of Figure 6 shows the service intervals at the gateway. Line C shows the node 1 packets arriving at the gateway. Line B shows node 2 packets arriving at the gateway. Line A shows the queue when $r_1 = r_2$. The x-axis shows time, and for line A the y-axis shows the queue size.

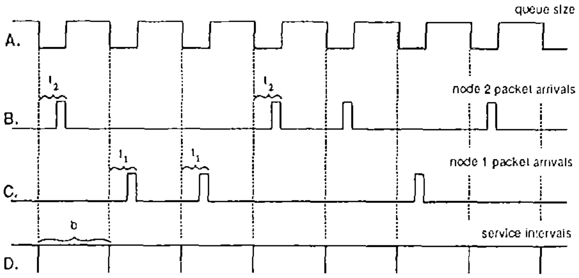

Figure 6: Phase of packet arrivals at the gateway, for $r_1 = r_2$.

For this informal argument, assume for simplicity that $s = 0$. In this case, when some node increases its window by one, two packets from that node arrive at the gateway simultaneously. Thus, when $r_1 = r_2$, the second packet arrives at a full queue, and is dropped. Thus for $r_1 = r_2$, when a node increases its window, a packet from that node will be dropped at the gateway.

Case 2: Now consider a network where r_1 and r_2 differ slightly from each other. We have two periodic processes with slightly different periods. Decrease r_2 slightly, so that $r_1 - b \le r_2 \le r_1 - t_1$. The packet arrivals are shown in Figure 7. It is no longer true that exactly one packet arrives at the gateway in each service interval. In Figure 7, the packets from node 2 arrive slightly earlier than their arrival time in Figure 6. When a node 2 packet arrives at the gateway following a node 1 packet, the two packets arrive in the same service interval.

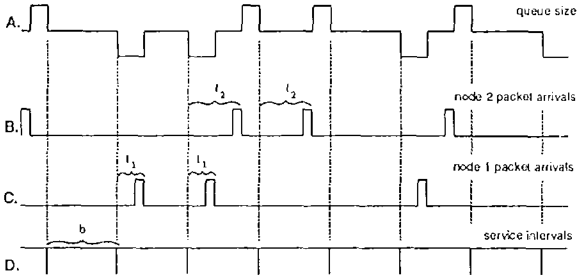

Figure 7: Phase of packet arrivals at the gateway, for $r_2 < r_1$.

Definitions: blank, node 1, node 2, and double service intervals. A *node 1 interval* is a service interval with only a node 1 packet arrival at the gateway. A *node*

2 interval is a service interval with only a node 2 packet arrival. A *blank interval* is a service interval with no packet arrivals, and *double interval* is a service interval with two packet arrivals. □

Now we describe the sequence of packet arrivals at the gateway, when $r_1 - b \leq r_2 \leq r_1 - t_1$. In a double service interval, a node 1 packet arrives at time t_1, followed at time $t_2 > t_1$ by a node 2 packet. Each double or node 2 interval is followed by a node 2 or blank service interval. (If a node 2 interval were followed by a node 1 interval, this would mean that one roundtrip earlier, a node 1 packet and a node 2 packet were transmitted by the gateway at the same time. This is not possible.) There cannot be two consecutive blank service intervals. (This would mean that one roundtrip earlier, there was a service interval during which no packets were transmitted by the gateway, violating the assumptions of the model.) Following each blank service interval, there is a (possibly empty) sequence of node 1 service intervals, followed by a double service interval, followed by a (possibly empty) sequence of node 2 service intervals, followed by another blank service interval. A rigorous proof is included in [FJ91].

As a result of this well-defined pattern of packet arrivals at the gateway, only node 2 packets cause the queue size to increase to $maxqueue$. As a result, regardless of which connection first increases its window, the gateway responds by dropping a packet from node 2. If node 2 increases its window, the additional node 2 packet arrives to a full queue, and is dropped. If node 1 increases its window, the additional node 1 packet increases the queue size to $maxqueue$. The next node 2 packet that arrives at the gateway will be dropped.

Case 3: A similar case occurs if r_2 is slightly greater than r_1, so that $r_1 + b - t_1 \leq r_2 \leq r_1 + b$. The packet arrivals are shown in Figure 8. When a node 1 packet arrives at the gateway after a node 2 packet, both packets arrive in the same service interval. In this case, only node 1 packets cause the gateway queue to increase to $maxqueue$. When some connection's window is increased, the gateway always drops a node 1 packet.

Thus, with a slight change in node 2's roundtrip time, the pattern of packet arrivals at the gateway can change completely. The network can change from unbiased behavior to always dropping packets from a particular connection. The pattern of packet arrivals is slightly more complex when r_1 and r_2 differ by more than b, but the performance results are similar. This is discussed in [FJ91]. In the next few paragraphs we use the results in this section to explain the simulation results in the previous section.

Definitions: drop period. The model that we have described concerns the *drop period* in a simulation, the period that begins when the queue first reaches size

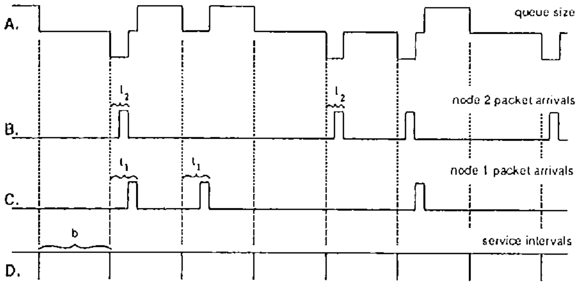

Figure 8: Phase of packet arrivals at the gateway, for $r_2 > r_1$.

maxqueue and that ends when one of the connections reduces its window, decreasing the rate of packets arriving at the gateway. This is similar to the *congestion epoch* defined in [SZC90]. If the maximum windows have not all been reached, then after the queue first reaches size *maxqueue*, it takes at most one roundtrip time until some node increases its window, and some packet is dropped. It takes one more roundtrip time until the rate of packets arriving at the gateway is decreased. Therefore, the drop period lasts for between one and two roundtrip times. □

When both roundtrip times are equal and neither node 1 nor node 2 have reached their maximum windows, node 1 and node 2 both increase their windows in each drop period. In this case both node 1 and node 2 packets are dropped by the gateway in each drop period. As a result, node 1 and node 2 each get roughly half of the total throughput. This is shown in Figure 4 when the roundtrip time ratio is 1.

When $r_1 - b \leq r_2 \leq r_1 - t_1$, as in Case 2, only node 2 packets are dropped at the gateway. Even when both node 1 and node 2 increase their windows during a drop period, for each increase a node 2 packet is dropped. This is shown in Figure 4 when r_2 ranges roughly from 211.44 ms. to 220 ms., corresponding to roundtrip ratios from 0.955 to 0.994. (The exact range for this behavior in Figure 4 is slightly different because in the simulation network $s \neq 0$. This is explained in more detail in [FJ91].)

When $r_1 + b - t_1 \leq r_2 \leq r_1 + b$, as in Case 3, only node 1 packets are dropped at the gateway. This is shown in Figure 4 when r_2 ranges roughly from 230 ms. to 231.44 ms., corresponding to roundtrip ratios from 1.039 to 1.045. Note that even when only node 1 packets are dropped, node 1's throughput is still nonzero. Node 1's window is allowed to increase each time until the queue overflows and node 1 packets are dropped.

In the simulations, for $r_2 > r_1 + b$ there is a repeating pattern, shown in the simulations in Figure 4. For each nonnegative integer i, for $r_1 + i * b < r_2 < r_1 + (i+1) * b$, first there is a range for r_2 in which node 2 packets are dropped in every drop period and node 1 packets might or might not be dropped. This is followed by a range for r_2 in which node 1 packets are dropped in every drop period and node 2 packets might or might not be dropped. This behavior depends on whether node 1 or node 2 packets arrive first during a double service interval. In [FJ91] we give a more complete set of formal proofs explaining this behavior.

For the simulations in Figure 4, node 1 packets arrive at the gateway early in the current service interval, after .144 of the current service interval. However, for the simulations in Figure 5 node 1 packets arrive at the gateway quite late in the current service interval. In this case, for a wide range of roundtrip times, packets from node 2 arrive at the gateway earlier in the service interval than node 1 packets, forcing a disproportionate number of drops for node 1 packets.

The behavior in a small, deterministic network is not necessarily characteristic of behavior in an actual network such as the Internet. The bias illustrated in Figure 4 is due to the fixed relationship of packet arrivals to departures at the gateway. It can be broken by adding sufficient randomization to the network, either in the form of random traffic (discussed in section 2.3) or in the form of random processing time at the nodes (discussed in section 2.4). Section 2.5 shows that the pattern of bias can be corrected with the Random Drop gateways, which are less sensitive than Drop Tail gateways to the exact timing of packet arrivals at the gateway. As discussed in [FJ91], patterns of bias can still be present when the network contains three or more FTP connections, all with different round trip times, or when the network contains multiple gateways.

We believe that this pattern of bias is noteworthy both because it appears in real networks and because it shows up frequently in network simulations. Sections 2.6 and 3.2 show that simulations and measurement studies of networks with Drop Tail gateways are sensitive to small changes in network parameters. As figures 4 and 5 suggest, the phase interaction is so large compared to other effects on throughput, simulations have to be designed with care and interpreted carefully to avoid a phase-induced bias.

2.3 Adding random traffic

In this section, we explore the extent to which patterns of bias persist in the presence of randomly-timed traffic. Telnet nodes in the simulation network send fixed-size packets at random intervals (drawn from an exponential distribution). The pattern of bias described above is strongest when all of the packets in the gateway queue are of the same size. Significant bias remains when roughly 15% of the traffic consists of random 1000-byte packets, and also when roughly 3% of the traffic consists of random 40-byte packets. However, when 15% of the traffic consists of random 40-byte packets, the pattern of bias is largely eliminated. These results are described briefly below, and are discussed in more detail in [FJ91].

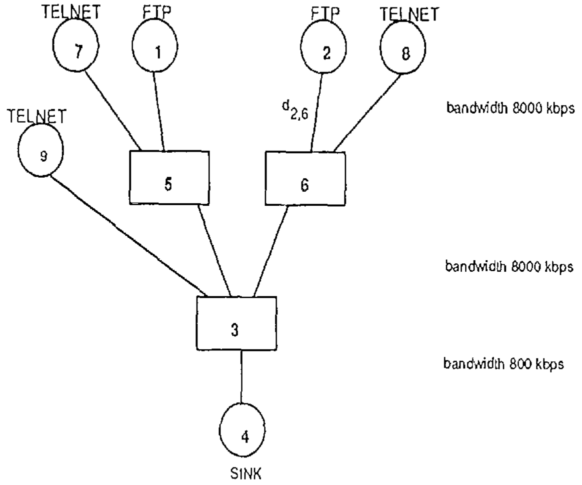

Figure 9: Simulation network with telnet and FTP nodes.

Figure 9 shows the simulation network with both FTP and telnet nodes. The delays on each edge are set so that, in the absence of queues, packets from node 1 have the same roundtrip time as in the network in Figure 3.

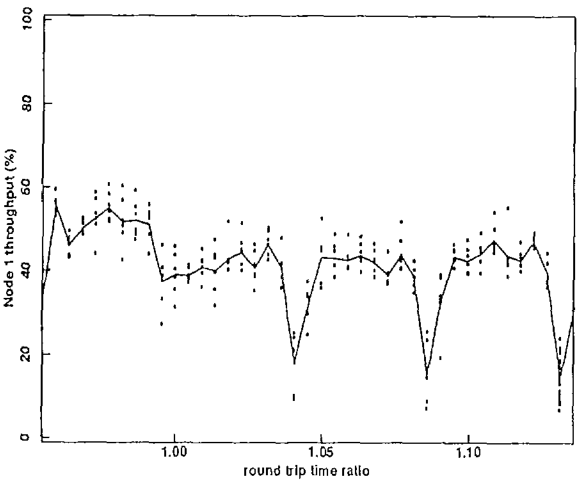

Figure 10: Node 1's throughput, with 1000-byte random packets as 15% of throughput.

Figure 10 shows results from simulations where each telnet node sends on the average five 1000-byte packets per second. (This is not meant to reflect realistic sizes for telnet packets, but simply to add a small number of randomly-arriving 1000-byte packets to the network.)

In these simulations, in 50 seconds the gateway processes roughly 750 random packets, and roughly 3700-4000 FTP packets. Because the network behavior depends on the precise timing of the randomly-sent telnet packets, and because this timing varies from one telnet packet to the next, for each set of parameters we show the results from several 50-second periods of a longer simulation. Each dot gives node 1's average throughput from one 50-second period of a simulation. The solid line gives the average throughput for node 1, averaged over all of the simulations. As Figure 10 shows, there is still discrimination for some roundtrip ratios even from simulations where roughly 15% of the packets through the gateway are random 1000-byte packets.

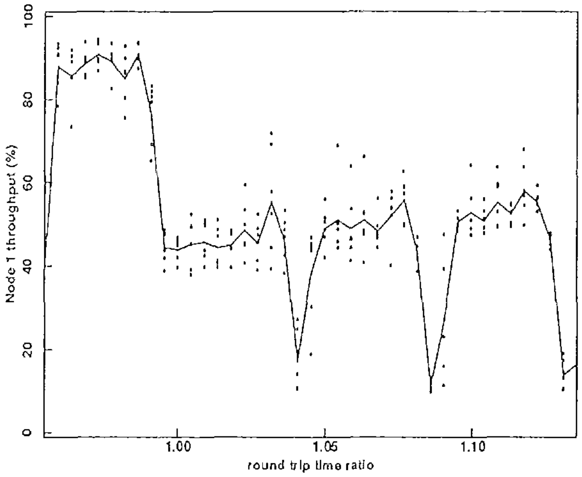

Figure 11: Node 1's throughput, with 40-byte random packets as 3% of throughput.

The results are different when each telnet node sends 40-byte packets. For the simulations in Figure 11, each telnet node sends on the average one 40-byte packet per second. In 50 seconds the gateway processes roughly 150 telnet packets, and roughly 4200-4700 FTP packets. Figure 11 shows that in simulations where roughly 3% of the packets at the gateway are random 40-byte packets, the pattern of discrimination still holds. However, in simulations where roughly 15% of the packets at the gateway are random 40-byte packets, the pattern of bias is broken. (These results are shown in [FJ91].)

When all of the packets in the gateway queue are the same size, then the gateway queue requires the same time b to transmit each packet. In this case, given congestion, each FTP packet from node i arrives at the gateway at a fixed time r_i mod b after the start of some service interval. This is no longer true when the gateway queue contains packets of different sizes. Thus, this pattern of bias is most likely to occur when most of the packets in the gateway queue are of the same size.

2.4 Adding randomness in the nodes

The node processing times in the simulations described so far has been deterministic. Each node is charged zero seconds of simulation time for the CPU time to process each packet. What if each node spends a random time processing each packet? In this case, the roundtrip time for each packet would have a random component apart from time waiting in queues. This could help to break up the fixed pattern of packet arrivals at the gateway.

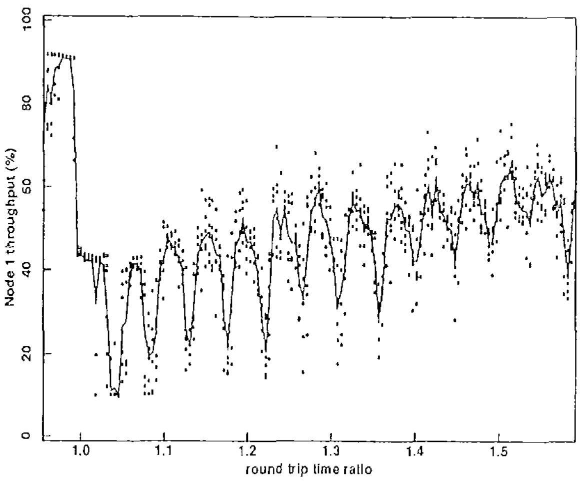

Figure 12: Node 1's throughput, with random processing time from 0 to 5 ms.

For the simulations in Figure 12, node 1 and node 2 each use a time uniformly chosen between 0 and 5 ms., half the bottleneck service time, to prepare each FTP packet after an ACK packet is received. This is not intended to reflect any assumptions about the behavior of actual networks. As Figure 12 shows, the pattern of discrimination is changed somewhat, but is still present. However, when node 1 and node 2 each use a time uniformly chosen between 0 and 10 ms., the bottleneck service time, to prepare each FTP packet, the pattern of discrimination is significantly reduced. (These results are shown in [FJ91].) The conclusion is that the pattern of discrimination remains when the total random component of the processing time in one roundtrip time is less than half the bottleneck service time. However, when the total random processing time in the nodes is as large as the bottleneck service time, the pattern of discrimination is likely to be broken.

2.5 Phase effects and Random Drop gateways

We show that with Random Drop gateways, the network bias shown in Figure 4 is eliminated. With Random Drop gateways, when a packet arrives at the gateway and the queue is full, a packet from the gateway queue is randomly chosen to be dropped. One goal for a randomized gateway is that the probability that the gateway drops a packet from a particular connection should be proportional to that connection's share of the total throughput. As we show in the following sections, Random Drop gateways do not achieve this goal in all circumstances. Nevertheless, Random Drop gateways are an easily-implemented, low-overhead, stateless mechanism that samples over some range of packets in deciding which packet to drop. The probability that the gateway drops a packet from a particular connection is proportional to that connection's share of the packets in the gateway queue when the queue overflows.

Consider a gateway with a maximum queue of $maxqueue$. When a packet arrives to a full queue, the gateway uses a pseudo-random number generator to choose a pseudo-random number n between 1 and $maxqueue+1$. (To save time, the pseudo-random number could be chosen in advance.) The nth packet in the gateway queue is dropped. Consider a queue that overflows because a node 1 packet arrives at the gateway immediately after a node 2 packet. With Random Drop gateways, the node 1 packet and the node 2 packet are equally likely to be dropped, along with any of the other packets in the queue at that time.

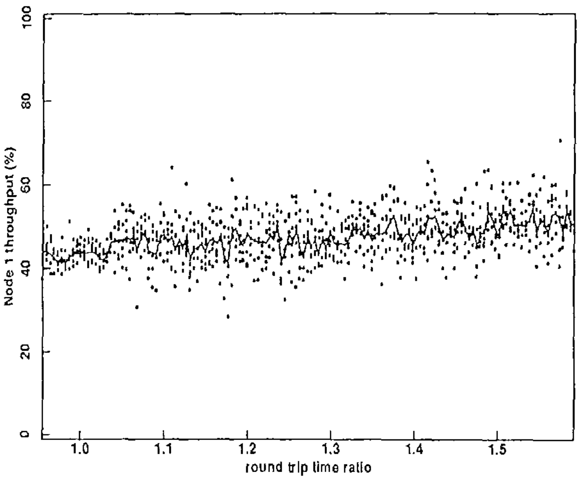

Figure 13: Node 1's throughput with Random Drop gateways.

Figure 13 shows the results from simulations using a Random Drop gateway along with the network shown in Figure 3. These simulations differ from the simulations in Figure 4 only in that the network use a Random-Drop instead of the Drop-Tail gateway. In Figure 13, each dot represents the throughput for node 1 in one 50-second interval of simulation. For each node 2 roundtrip time, six 50-second intervals are shown. The solid line shows the average throughput for node 1 for each roundtrip time ratio. As Figure 13 shows, Random Drop eliminates the bias observed in simulations with the Drop Tail gateway.

For the simulations in Figure 13 there are roughly 30 packet drops in each 50-second interval of simulation. If the queue contained an equal numbers of packets from node 1 and node 2 each time it overflowed, the probability that one node received all 30 packet drops would be 2^{-29} (roughly one in a billion). The statistical nature of the Random Drop algorithm is a good protection against systematic discrimination against a particular connection.

Random Drop gateways are not the only possible gateway mechanism for correcting the bias caused by traffic phase effects. However, the use of randomization allows Random Drop gateways to break up this pattern of bias with a stateless, low-overhead algorithm that could be easily implemented in present networks and that would scale well to networks with many connections.

The simulations in Figure 13 work well because, for the parameters in these simulations, the contents of the gateway queue at overflow are fairly representative of the average contents of the gateway queue. Nevertheless, it is possible to construct simulations with Random Drop gateways where this is not the case. When two connections have roundtrip times that differ by more than the bottleneck service time, then the packets from the two connections tend to get "shuffled together" in the gateway queue. Even for those simulations in Figure 13 where the two connections have similar roundtrip times, the packets from the two connections tend to intermix whenever one of the connections has a current window greater than the pipe size. Nevertheless, it is possible to construct simulations where the maximum windows are less than the pipe size, and the roundtrip times for the two connections are the same. In this case, the gateway always transmits a window of node 1 packets followed by a window of node 2 packets [SZC90]. In this case there is no mechanism to break up clumps of packets. and the contents of the gateway queue at overflow are seldom representative of the average contents. Thus, the use of randomization in Random Drop gateways is not sufficiently powerful to break up all patterns of packet drops.

2.6 Bias against telnet nodes

In this section we examine possible discrimination against telnet nodes, in a network where all connections have the same roundtrip times. We show that discrimination against telnet nodes is a possibility in networks with Drop Tail gateways and this discrimination can be affected by small changes in the phase of the FTP connections or in the maximum queue size at the bottleneck. We show that the use of Random Drop gateways eliminates discrimination against telnet traffic.

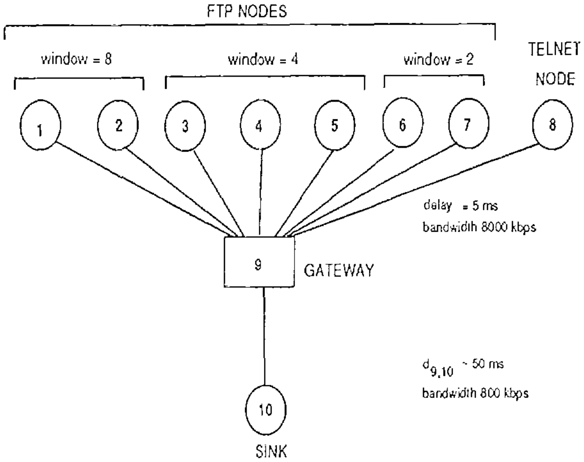

Figure 14: Network with FTP and telnet nodes.

Figure 14 shows the simulation network. There is one telnet connection, and seven FTP connections, with maximum windows ranging from 2 to 8 packets. The telnet connection sends an average of one packet per second, for an average of 50 packets in 50 seconds of simulation. All connections have the same roundtrip time.

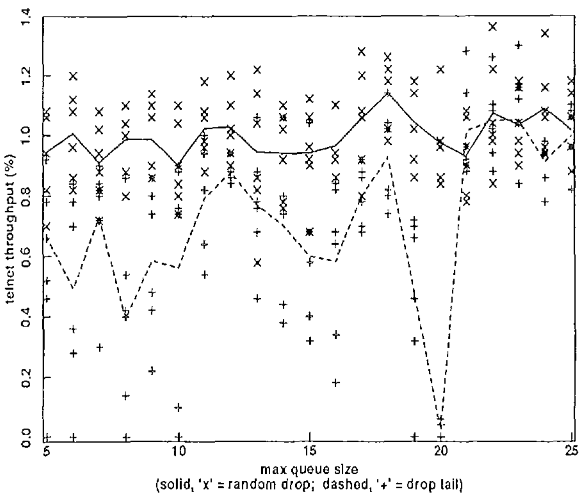

Figure 15: Telnet throughput, for $d_{9,10} = 50$ ms.

We compare simulations with Random Drop and with Drop Tail gateways. Figure 15 shows simulations with $d_{9,10} = 50$ ms., for a roundtrip time, in the absence of queues, of 121.44 ms. Each set of simulations was run with the maximum queue size ranging from 5 to 25 packets. For each choice of parameters, three 100-second simulations were run. Each “x” or “+” shows the telnet node’s average throughput in one 50-second period of simulation. The solid line shows the telnet node’s average throughput with Random Drop gateways and the dashed line shows it with Drop Tail gateways.

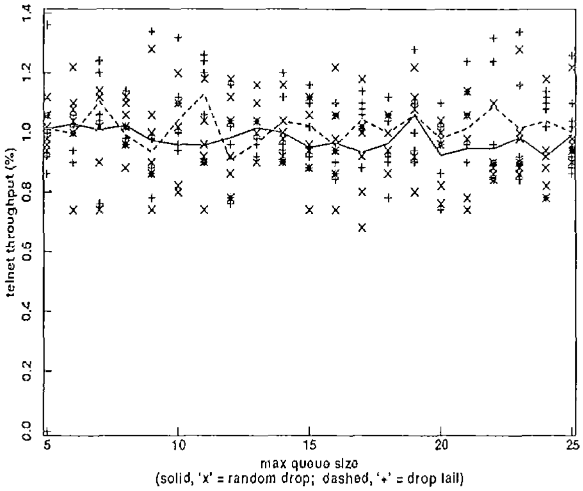

Figure 16: Telnet throughput, for $d_{9,10} = 53.7$ ms.

For the simulations in Figure 15, when the maximum queue is 20 packets, the FTP connections fill but don’t overflow the gateway queue. FTP packets arrive at the gateway 1.44 ms. after the start of the current service interval, or after 14.4% of the current service interval has been completed. With Drop Tail gateways, a telnet packet arriving at the gateway at a random time has an 85.6% chance of arriving at a full queue and being dropped. For these parameters, the telnet node is easily shut out. When the maximum queue is greater than 20 packets no packets are dropped and the telnet node’s throughput is limited only by the rate at which telnet packets are generated. When the maximum queue is less than 20 packets, even for a fixed set of parameters, the throughput for the telnet node can vary widely from one simulation to the next. In some simulations with Drop Tail gateways, some of the FTP connections get shut out, allowing the queue to fill up, shutting out the telnet node. In other simulations, the FTP connections continually adjust their windows as a result of packet drops and the queue is often not full. In these simulations, the telnet node’s throughput is relatively high.

Figure 16 shows the results of simulations for $d_{9,10} = 53.7$ ms. In this case, the roundtrip time in the absence of queues is 128.84 ms. and FTP packets arrive at the gateway after 88.4% of the current service interval has been completed. Even with Drop Tail gateways and a maximum queue size of 20 packets, randomly-arriving telnet packets have only an 11.6% chance of arriving at the gateway after some FTP packet, and of being dropped. For the simulations with $d_{9,10} = 53.7$ ms., telnet nodes are never shut out, regardless of the maximum queue size.

Figures 15 and 16 show that with Drop Tail gateways, it is possible for telnet nodes to be shut out by FTP connections. This behavior is affected by small changes in the network parameters, and this behavior can also change drastically from one simulation to the next, for a fixed set of parameters. The simulation in [DKS90] showing two telnet connections shut out by six FTP connections, for example, should be interpreted with this sensitivity to the exact network parameters in mind.

As Figures 15 and 16 show, the throughput for the telnet node is consistently high in all of the simulations with Random Drop gateways. The randomization in Random Drop gateways is sufficient to overcome any pattern of discrimination against the telnet nodes.

3 A discussion of Random Drop gateways

3.1 Previous research on Random Drop gateways

In [H89], Hashem evaluates the Random Drop gateway algorithm. The benefits of Random Drop gateways over Drop Tail gateways reported in [H89] include fairness to late-starting connections, and slightly improved throughput for connections with longer roundtrip times. [H89] reports on simulations of a network with two connections, one local and one long-distance, with large maximum windows and a shared gateway. In these simulations, the long-distance connection receives higher throughput with Random Drop gateways than with Drop Tail gateways. Nevertheless, in both cases, the local connection receives higher throughput than the long-distance connection.

The shortcomings of the Random Drop algorithm discussed in [H89] include the preferential treatment reported above for connections with shorter roundtrip times, a higher throughput for connections with larger packet sizes, and a failure to limit the throughput for connections with aggressive TCP implementations. These shortcomings are shared by networks with Drop Tail gateways.

[H89] investigates Early Random Drop gateways as a mechanism for congestion avoidance as well as for congestion control. In the implementation of Early Random Drop gateways in [H89], the gateway randomly drops packets when the queue length exceeds a certain level. Because Early Random Drop gateways have a broader view of the traffic distribution than do Random Drop gateways, [H89] suggests that they have a better chance that Random Drop gateways of targeting aggressive users. [H89] further suggests that Early Random Drop gateways might correct the tendency of Drop Tail and Random Drop gateways of synchronously dropping many connections during congestion. In [H89], additional work on Early Random Drop gateways is recommended. In the conclusions on Random Drop gateways, [H89, p.103] reports that "In general, ... Random Drop has not performed much better than the earlier No Gateway Policy (Drop Tail) approach. It is still vulnerable to the performance biases of TCP/IP networks." We examine these performance biases in more detail in the next two sections.

[Z89] uses simulations to evaluate Random Drop gateways. [Z89] concludes that Random Drop does not correct Drop Tail's problem of uneven throughput given uneven path lengths, and that neither Random Drop nor a version of Early Random Drop is successful at controlling misbehaving users. For the simulations in [Z89], Zhang remarks that the bias against traffic with longer roundtrip times results because "after a period of congestion, connections with a shorter path can reopen the control window more quickly than those with a longer path [Z89, p.99]." We examine this problem in more detail in Section 3.3.

[M90] presents a measurement study of a network with local and long distance traffic, with several congested gateways. The Random Drop and the Drop Tail gateway algorithms are compared. Three topologies are explored, with one, two, and three congested gateways, respectively. For each topology, there was one longer connection, and many shorter connections, each with a maximum window of eight packets. For some of the simulations, the throughput for the longer connection was better with Random Drop gateways, and for other simulations the throughput was better with Drop Tail gateways. As Section 3.2 explains, we believe that these results should be interpreted keeping traffic phase effects in mind. [M90, p.6] reports that "Random Drop Congestion Recovery improves the fairness of homogeneous connections that have the same bottleneck, but beyond that, it has limited value."

The June 1990 draft of the Gateway Congestion Control Survey by the IETF Performance and Congestion Control Working Group [MR90] discusses the results from [M90]. The suggestion is that "Random Drop Congestion Recovery should be avoided unless it is used within a scheme that groups traffic more or less by roundtrip time. [MR90, p.8]" In this paper, we suggest that, in comparison to the current Drop Tail gateways, Random Drop gateways offer significant advantages and no significant disadvantages.

[DKS90] briefly compares Fair Queueing gateways with Random Drop. They report that Random Drop gateways "greatly alleviate" the problem of segregation with Drop Tail gateways, but that they do not provide fair bandwidth allocation, do not control ill-behaved sources, and do not provide reduced delay to low-bandwidth conversations. A comparison of Random Drop gateways with rate-based gateway algorithms such as Fair Queueing, or an examination of traffic phase effects in Fair Queueing gateways, is beyond the scope of this paper.

3.2 Bursty traffic

One objection to Random Drop gateways in the literature is that with Random Drop gateways (as with Drop Tail gateways), connections with a longer roundtrip time receive reduced throughput. In this section, we look at the problems of bursty traffic [4] resulting from connections with longer roundtrip times and small windows. In the following section, we look at the effect of the window increase algorithm on connections with longer roundtrip times and large windows.

We explore the performance of Drop Tail and Random Drop gateways in networks with a range of roundtrip times in some detail, for several reasons. First, the poor performance of Random Drop gateways for connections with longer roundtrip times has been cited as one reason to avoid the use of Random Drop gateways with mixed traffic. Second, we emphasize the danger of interpreting results from simulations or measurement studies with Drop Tail gateways without considering the effect of small changes in the network parameters on network performance. This includes the effect caused by changes in traffic phase. Third, issues of networks with a range of roundtrip times are becoming more important in high-speed networks.

In this section we give an example of a configuration where the contents of the gateway queue when the queue overflows are not necessarily representative of the average throughput. In networks with Drop Tail or Random Drop gateways, connections with longer roundtrip times and small windows can receive a disproportionate number of dropped packets, as reported in [M90]. The simulations in this section compare the performance of Drop Tail and Random Drop gateways, and show that the performance with Drop Tail gateways can be influenced by traffic phase.

We consider simulations of the network in Figure 17. For a node with maximum window W and roundtrip time R, the throughput is limited to W/R packets per second. A node with a long roundtrip time and a small window receives only a small fraction of the total throughput. In our configuration, when node 5 has a small window, the packets from node 5 often arrive at the gateway in a loose cluster. (By this, we mean that considering only node 5 packets, there is one long interarrival time, and many smaller interarrival times.) If the gateway queue is only likely to overflow when a cluster of node 5 packets arrives at the gateway, then, even with Random Drop gateways, node 5 packets have a disproportionate probability of being dropped.

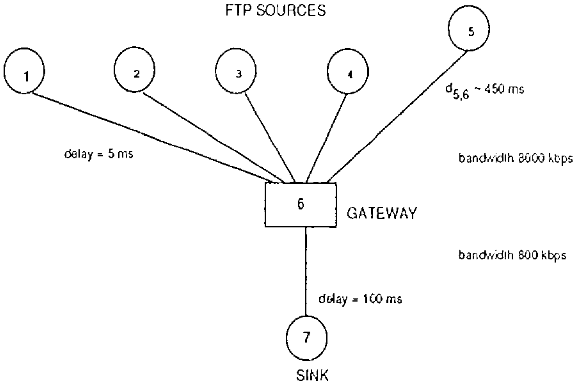

Figure 17: A simulation network with five FTP connections.

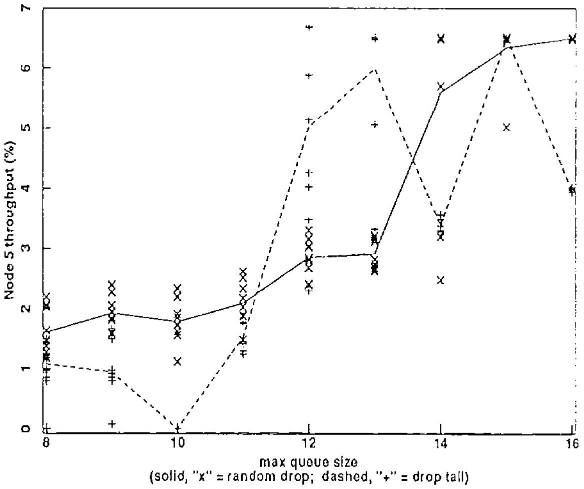

Figure 18: Node 5's throughput, with $d_{5,6} = 449.4$ ms.

Figures 18 and Figure 19 show the results of simulations for the network in Figure 17. The simulations were run for Drop Tail and for Random Drop gateways, for a range of queue sizes, and for two slightly different choices for node 5's roundtrip time. For each set of parameters, the simulation was run for 500 seconds. Each mark represents one 50-second period, excluding the first 50-second period. The x-axis shows the queue size, and the y-axis shows node 5's average throughput. For each figure, the solid line shows the average throughput with Random Drop gateways, and the dashed line

[4] By bursty traffic we mean traffic from connections where the current window is small compared to the delay-bandwidth product or connections where the amount of data generated in one roundtrip time is small compared to the delay-bandwidth product.

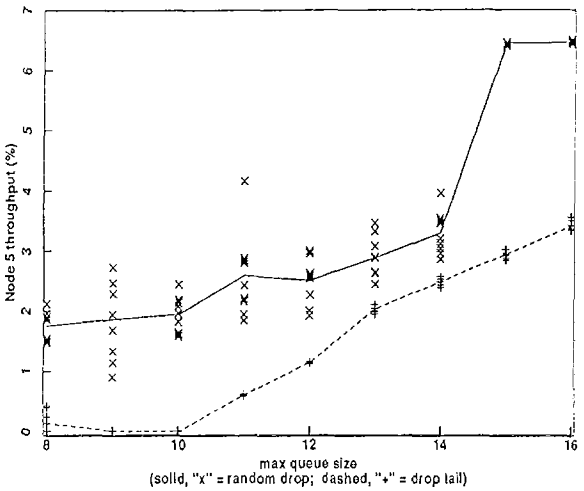

Figure 19: Node 5's throughput, with $d_{5,6} = 453$ ms.

shows the average throughput with Drop Tail gateways.

With Drop Tail gateways, the throughput for node 5 is affected by small changes in phase for node 5 packets. We show simulations for delay $d_{5,6} = 449.4$ ms. and for $d_{5,6} = 453$ ms. This corresponds to roundtrip times for node 5 of 1110.24 ms. and 1117.44 ms., respectively. Packets from nodes 1-4 have a roundtrip time of 221.44 ms.

For the network in Figure 17, when the maximum queue is ten or greater, congestion is low, and the gateway queue only overflows when the queue contains packets from node 5. In this case, with Random Drop gateways, the node 5 packets have a disproportionate probability of being dropped, because the queue contents when the queue overflows are not representative of the average queue contents. The performance of the Random Drop gateway is not significantly affected by small changes in traffic phase.

With Drop Tail gateways, with a maximum queue greater than 10, the probability that a node 5 packet arrives to a full queue depends on the precise timing of packet arrivals at the gateway. For simulations with $d_{5,6} = 449.4$ ms., node 5 packets arrive at the gateway at the start of a service interval, and these packets are unlikely to arrive at a full queue. For simulations with $d_{5,6} = 453$ ms., node 5 packets arrive towards the end of the service interval, and are more likely to be dropped. For the simulations with $d_{5,6} = 449.4$ ms., the performance of the network is also affected by small changes in the maximum queue size.

Note that with Random Drop gateways, node 5 is never completely shut out. This is in contrast to simulations with Drop Tail gateways for a maximum queue of 10. With this queue size, the gateway queue is full but not overflowing before packets from node 5 arrive. For simulations with $d_{5,6} = 453$ ms., node 5 packets are always dropped when they arrive at the gateway. For simulations with $d_{5,6} = 449.4$ ms., the explanation is slightly more complicated, but the results are the same. In this case, because of the phase of the shorter FTP connections, node 5 packets are likely to be dropped if they arrive at the gateway at a somewhat random time after a timeout.

In general, when running simulations or measurement studies with Drop Tail gateways in small deterministic networks, it is wise to remember that a small change in traffic phase, or in the level of congestion, might result in a large change in the performance results. Thus, the results in this section are not inconsistent with the results in [M90], which show that for a particular network with one congested gateway, the throughput for the longer connection was higher with Drop Tail gateways than with Random Drop gateways.

In summary, for some set of parameters, Drop Tail gateways give better throughput for node 5, and for other sets of parameters, Random Drop gateways give better throughput for node 5. In both cases, the throughput for node 5 is fairly low. The performance problems for nodes with long roundtrip times and small windows are neither cured, nor significantly worsened, by Random Drop gateways.

We suggest that the throughput for bursty traffic can be improved with Random Early Detection (RED) gateways, where incipient congestion is detected early, and the packet to be dropped is selected from a broad range of packets. We are in the initial stages of an investigation of Random Early Detection gateways. We include only a cursory discussion of RED gateways in this paper. They will be discussed in more detail in a future article.

With our implementation of RED gateways, the gateway computes the average size for each queue using a weighted exponential running average. When the average queue size exceeds a certain threshold, the gateway randomly chooses a packet to drop and increases the threshold. The threshold then slowly decreases to its previous value. The packet drop choice is made by choosing a random number n in the interval 1 to $range$, where $range$ is a parameter of the gateway. The nth packet to arrive at the gateway is then dropped. In moderate congestion, $range$ is large, and the probability that a packet from some node is dropped is roughly proportional to that node's average share of packets through that queue. In high congestion, $range$ is decreased, decreasing the feedback time to the network. With an RED gateway under moderate congestion, a node that transmits packets in a cluster does not have a disproportionate probability of being dropped.

Figure 20 shows the result of simulations with RED gateways, for $d_{5,6} = 450$ ms. The x-axis shows the

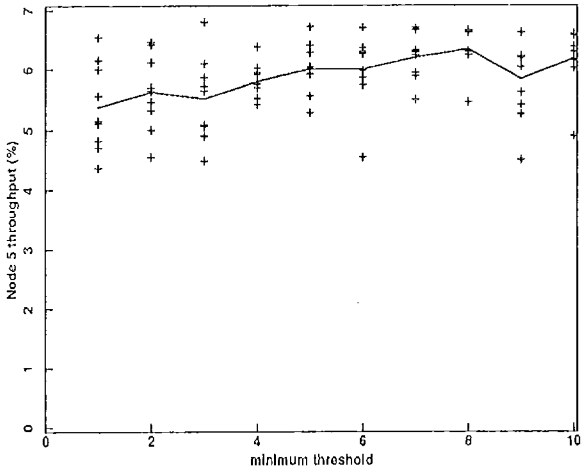

Figure 20: Simulation with RED gateways

minimum threshold for the RED gateway, and the y-axis shows the average throughput for node 5. The throughput for node 5 is close to the maximum possible throughput, given node 5's roundtrip time and maximum window. For these simulations, the maximum queue is 60 and the average queue size ranges from 5 to 11 packets. For the simulations in Figure 18 and 19, the average queue size ranges from 4 to 12 packets. This chart simply suggests that the problems of reduced throughput for connections with long roundtrip times and small windows could be cured by a gateway where the probability of a packet drop for a connection is roughly proportional to that connection's fraction of the throughput.

3.3 Interactions with window adjustment algorithms

In this section we discuss the bias against connections with longer roundtrip times in networks with TCP congestion avoidance. We show that this bias is similar for Drop Tail or for Random Drop gateways and suggest that it results from the end-node TCP window increase algorithm, not from the gateway algorithm. With the current algorithm in 4.3 BCD TCP, in the absence on congestion each connection increases its window by one packet each roundtrip time. This algorithm is attractive because it is simple and time-invariant, but has the result that throughput increases at a faster rate for connections with a shorter roundtrip time. This results in a bias against connections with longer roundtrip times. In this section we examine this bias and we discuss possible alternatives to the window increase algorithm.

We ran simulations for the configuration in Figure 3 with two FTP connections and one shared gateway. In these simulations, each source has a maximum window equal to the delay-bandwidth product. For the simulations, node 1's roundtrip time is fixed and node 2's roundtrip time ranges up to more that eight times node 1's. Thus node 2's maximum window ranges from 22 packets to more than 180 packets. Figure 21 shows the result of simulations with Drop Tail gateways, and Figure 22 shows the result of simulations with Random Drop gateways. The x-axis shows node 2's roundtrip time as a multiple of node 1's roundtrip time. The solid line shows node 1's average throughput, and the dashed line shows node 2's average throughput. The gateway has a maximum queue size of 15 packets.

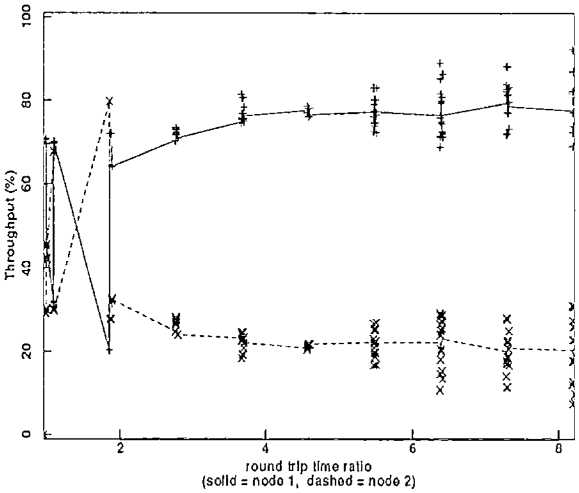

Figure 21: Node 1 and node 2 throughput with Drop Tail gateways.

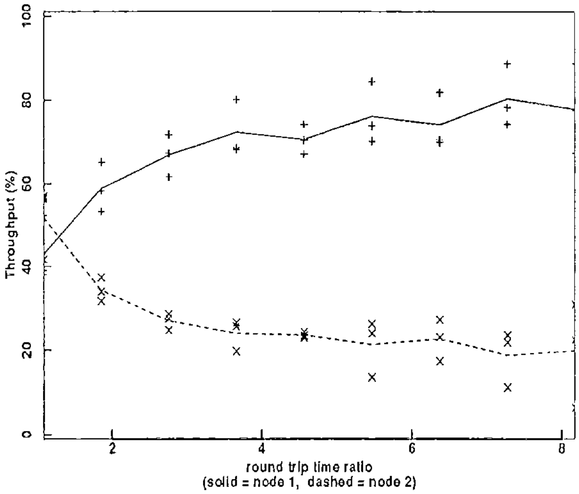

Figure 22: Node 1 and node 2 throughput with Random Drop gateways.

Figure 21 shows the results of simulations with Drop Tail gateways. For each cluster of simulations, we varied node 2's roundtrip time over a 10 ms. range to consider phase effects. In these simulations, phase changes significantly affect performance only when node 2's roundtrip time is less than twice node 1's. Figure 22 shows the results of simulations with Random Drop gateways. For both sets of simulations, as node 2's roundtrip time increases node 2's throughput decreases significantly. We suggest that this behavior is a result of the TCP window modification algorithms. In our simulations, the performance is only somewhat improved by the use of RED gateways.

For the moment, let r_i denote node i's average roundtrip time including queueing delays. In the congestion avoidance phase of TCP, node i's window is increased by 1 packet roughly every r_i seconds. That is, in each second, node i's throughput is increased by $1/(r_i)^2$ pkts/sec. Therefore, when a packet from node 2 is dropped and node 2's window decreased, it takes node 2 significantly longer than node 1 to recover its former throughput rate. This accounts for the reduced throughput.

Note that if each node i increased its window by $c * (r_i)^2$ packets each roundtrip time, for some constant c, then each node would increase its throughput by c pkts/sec in one second, regardless of roundtrip time. Since each source already has an estimate for the roundtrip time for each connection, such an algorithm is easily implemented. Our simulations show that with such a TCP window-increase algorithm, in a network with RED gateways that do not discriminate against bursty traffic, node 1 and node 2 each receive roughly half of the total throughput, regardless of roundtrip time. (This result, along with analysis, will be discussed in more detail in a future paper.)

This is in accord with the results in [CJ89]. In this paper, linear algorithms for increasing and decreasing the load are considered, where the load can be considered either as a rate or as a window. It is shown that a purely additive increase in the load gives the quickest convergence to fairness. For the model in [CJ89], this increase occurs at discrete time intervals. For a network with connections with different roundtrip times, comparable rates and comparable windows are quite different things. If the fairness goal is to provide comparable rates for connections with different roundtrip times, then the quickest convergence to fairness should occur with an additive increase in the rate for each fixed time interval. This is accomplished if every source increases its rate by c pkts/sec each second, for some constant c. This is equivalent to each connection increasing its window by $c * (r_i)^2$ packets each roundtrip time. If the fairness goal is to allocate equal network resources to different connections, a connection traversing n congested gateways uses n times the resources of one traversing one gateway. To be 'fair', the long connection should get only $1/n$th the bandwidth of the short. This would require a different window increase algorithm. With a window increase of $c * r_i$ packets each roundtrip time, for example, each connection increases its window by c packets in one second, and increases its throughput by c/r_i pkts/sec each second.

We are currently investigating alternatives to the current TCP window modification algorithms. There are many open questions: If the goal is for each connection to increase its rate by c pkts/sec each second, how do we choose c? What would be the impact of connections with large maximum windows increasing their window much more rapidly than they do now? Instead of using the average roundtrip time to calculate window increases, would it be better to use the average window size, averaged over a rather long period of time, or some other measure? And the ultimate difficult question: What is the meaning of "fair"? At the moment, this section is intended only to suggest that the current network bias in favor of connections with shorter roundtrip times is a result of the TCP window increase algorithm, and not of the performance of Random Drop or of Drop Tail gateways.

4 Conclusions and future research

In this paper we have considered the behavior of networks with highly periodic traffic and deterministic gateways. In particular, we have demonstrated that the performance of networks with periodic traffic and Drop Tail gateways can change significantly with small changes in traffic phase. The use of Drop Tail gateways can result in systematic discrimination against a particular connection. This performance depends on the phase relationship between connections, and is therefore sensitive to small changes in the roundtrip times for the connections. We have discussed the extent to which this pattern of discrimination can persist in the presence of random traffic in the network or in the presence of random CPU processing time.

We do not feel this pattern of discrimination is a significant problem in current networks (the present NSFNet backbone is too lightly loaded to suffer greatly from this problem) but there is certainly evidence that it exists. However, we do believe that this pattern of discrimination is a significant problem in the interpretation of simulation results or of measurement studies of networks using Drop Tail gateways.

We have argued that phase-related biases can be eliminated with the use of appropriate randomization in the gateways. In particular, Random Drop gateways are a stateless, easily-implemented gateway algorithm that does not depend on the exact pattern of packet arrivals at the gateway. The use of Random Drop gateways eliminates the pattern of bias due to traffic phase described in this paper.

There are several areas in which both Random Drop and Drop Tail gateways give disappointing performance. This includes a bias against connections with longer roundtrip times, a bias against bursty traffic, a bias against traffic with multiple gateways, and an inability to control misbehaving users. We have discussed several of these biases in this paper and we are currently investigating the other biases listed above. We are aware of no significant disadvantages to Random Drop gateways in comparison to Drop Tail gateways. This is in contrast to some earlier reports in the literature [MR90].

We have shown in Section 3.2 that the bias against connections with bursty traffic is slightly different for Random Drop and for Drop Tail gateways. With Drop Tail gateways, the performance is sensitive to small changes in traffic phase or in the level of congestion. Thus, in some cases, Drop Tail gateways give better performance for bursty traffic and in other cases Random Drop gateways give better performance. In our opinion this is not an argument against Random Drop gateways. Our current research suggests that this bias against connections with bursty traffic can be corrected with RED gateways, which provide for congestion avoidance as well as congestion control.

We have suggested in Section 3.3 that the bias against connections with longer roundtrip times and large windows results from the TCP window increase algorithm. We are currently investigating the implications of the bias against traffic with multiple congested gateways, and we are exploring possible modifications of the RED gateway to identify misbehaving users. It is our belief that RED gateways in general are a promising area for further research.

There are still many open questions. In our opinion, more research is needed in order to evaluate the implications of the competing goals for network performance. Maximizing fairness and maximizing total throughput are examples of possibly competing goals. Given congestion, do we want existing networks to provide the same throughput for connections with multiple congested gateways as for connections that use only one congested gateway? What would be the consequences of changing the window increase algorithm so that connections with longer roundtrip times increased their throughput at the same rate as connections with shorter roundtrip times? Can we develop a mechanism for controlling misbehaving users that is easy to implement and requires low overhead? These are all questions for future research.

This paper has been focused on understanding the behavior of existing networks, and on possible changes to existing networks, rather than on designing high-speed networks for the future. Nevertheless, many of the issues discussed in this paper could still be of concern in future networks. Such issues include the use of randomization in gateways to cope with patterns in network traffic, the design of gateways to accommodate bursty traffic, and the adaptation of window modification algorithms for networks containing connections with a broad range of roundtrip times.

5 Acknowledgements

We thank Scott Shenker and Lixia Zhang for useful conversations about these matters. We thank Srinivasan Keshav, Steven McCanne, and Chris Torek for helpful comments on an early draft. This work would have been impossible without Steven McCanne, who made all of the necessary modifications to our simulator.

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.

[CJ89] Chiu, D.-M., and Jain, R., "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", Computer Networks and ISDN Systems, V. 17, pp. 1-14, 1989.

[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.

[FJ91] Floyd, S., and Jacobson, V., *On Traffic Phase Effects in Packet-Switched Gateways*, in preparation.

[H89] Hashem, E., "Analysis of random drop for gateway congestion control", *Report LCS TR-465*, Laboratory for Computer Science, MIT, Cambridge, MA, 1989.

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

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

[L89] LaTouche, Guy, "A Study of Deterministic Cycles in Packet Queues Subject to Periodic Traffic", Bellcore Technical Memorandum, 1989.

[L90] LaTouche, Guy, "Sample Path Analysis of Packet Queues Subject to Periodic Traffic", Computer Networks and ISDN Systems, V. 20, pp. 409-413, 1990.

[M90] Mankin, A., *Random Drop Congestion Control*, Proceedings of SIGCOMM 90, Sept. 1990.

[MR90] Mankin, A. and Ramakrishnan, K. K., editors for the IETF Performance and Congestion Control Working Group, "Gateway congestion control survey", *Draft RFC*, June, 1990.

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

[RW90] Ramaswami, W., and Willinger, W., "Efficient Traffic Performance Strategies for Packet Multiplexors", Computer Networks and ISDN Systems, V. 20, pp. 401-412, 1990.

[SZC90] Shenker, S., Zhang, L., and Clark, D., "Some Observations on the Dynamics of a Congestion Control Algorithm", Computer Communication Review, V.20 N.5, October 1990, p. 30-39.

[Z89] Zhang, L., "A New Architecture for Packet Switching Network Protocols", MIT LCS TR-455, Laboratory for Computer Science, Massachusetts Institute of Technology, August 1989.