On 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

SUMMARY

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 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**

[*] An earlier version of this paper appeared in Computer Communication Review, V.21 N.2, April 1991.

[†] 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 non-existent 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.

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 on overflow, rather than dropping the tail, 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 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.

KEY WORDS Congestion control   Phase effects   Random drop gateways

1 Introduction

The first part of this paper presents 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 performance biases in networks and in network simulations. We show that gateways with appropriate randomization, such as Random Drop gateways, can eliminate the bias due to traffic phase effects.

The second part of this paper discusses the biases against bursty traffic and the biases against connections with longer roundtrip times that have been reported in networks with both Drop Tail and with Random Drop gateways. We show that the first bias results from the gateway congestion recovery algorithms, and that the second bias results from the TCP window modification algorithm. We show that these biases could be avoided by modifications to the gateway and to the TCP window modification algorithm respectively.

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. When the source receives an ACK it immediately transmits another data packet.

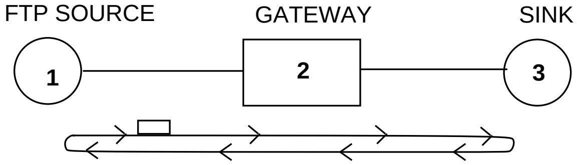

Figure 1: Periodic traffic.

Thus the roundtrip time (including queueing delays) of the connection is the traffic "period".

Most current network traffic is either bulk data transfer (i.e., the total amount of data is large compared to the bandwidth-delay product and throughput is limited by network bandwidth) or interactive (i.e., transfers are 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:

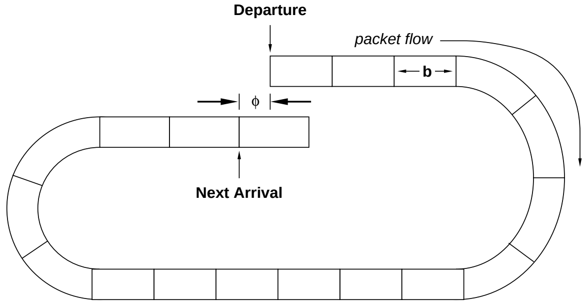

Figure 2: The phase (ϕ) of a simple packet stream.

Packets leaving the bottleneck gateway are all the same size and have a transmission time of b seconds. The source-sink-source "pipe" is completely full (i.e., if the roundtrip time including queueing delay 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). The queue length is decremented

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

at packet departures and incremented at packet arrivals. 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.

For a connection where the window is just large enough to fill the queue the phase is simply the (average) time this particular connection leaves a vacancy in the queue. If the connection has filled the gateway queue, the probability that a (random) telnet packet will successfully grab a 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. 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 2.7 describes this in detail.

Phase effects are more common than the example above suggests. When a deterministic gateway congestion management mechanism is driven by backlog, phase effects can cause a significant bias. In this paper, we concentrate on traffic phase effects in networks with Drop Tail gateways and TCP congestion management, where each source executes the 4.3BSD TCP congestion control algorithm (Jacobson, 1988). Section 2.3 demonstrates phase effects in an ISO-IP/TP4 network using DECbit congestion management (Ramakrishnan and Jain, 1990).

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. One example concerns the periodicity of packetized voice traffic where each voice source alternates between talk spurts and silences (Ramaswami and Willinger, 1990). 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. 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 *any* position-based strategy of dropping packets results in successive packet losses for one voice spurt (LaTouche, 1989) (LaTouche, 1990). Even though the beginning and endings of talk spurts break up the periodic pattern of packet drops, the periodic pattern is quickly reestablished. However, a "random drop" strategy works well in distributing the packet losses across the active conversations (Ramaswami and Willinger, 1990).

The first half of the paper contains simulations showing a bias due to traffic phase in networks with Drop Tail gateways, and analyzes this bias. The behavior in a small, deterministic simulation network is not necessarily characteristic of behavior in an actual network such as the Internet. The bias from traffic phase effects can be broken by adding sufficient randomization to the network, either in the form of random telnet traffic or in the form of random processing time at the nodes. The first half of the paper shows the success of Random Drop gateways in eliminating the bias due to traffic phase effects.

We believe that the pattern of bias discussed in this paper is noteworthy because it could appear in actual networks and because it shows up frequently in network simulations. Many simulations and measurement studies of networks with Drop Tail gateways are sensitive to small changes in network parameters. The phase interaction can be sufficiently large compared to other effects on throughput that simulations have to be designed with care and interpreted carefully to avoid a phase-induced bias.

The second half of the paper addresses some of the criticisms of Random Drop gateways from the literature. TCP/IP networks with either Drop Tail or Random Drop gateways share a bias against bursty traffic and a bias against connections with longer roundtrip times. The second half of the paper suggests that the bias against bursty traffic could be corrected by a gateway that detects incipient congestion, with the probability of dropping a packet from a particular connection proportional to that connection's share of the throughput.

The paper shows that the bias of TCP/IP networks (with either Drop Tail or Random Drop gateways) against connections with longer roundtrip times results from TCP's window modification algorithm. The second half of the paper investigates a modification to TCP's window increase algorithm that eliminates this bias. This modified window increase algorithm increases each connection's *throughput rate* (in pkts/sec) by a constant amount each *second*. In contrast, the current TCP window increase algorithm increases each connection's *window* by a constant amount each *roundtrip time*.

2 Traffic phase effects

2.1 Simulations of phase effects

This section gives 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, the two connections receive 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. Section 2.5 shows that this preferential behavior is absent in simulations where an appropriate random component (other than queueing delay) is added to the roundtrip time for each packet. This preferential behavior is also absent in simulations with Random Drop instead of Drop Tail gateways.

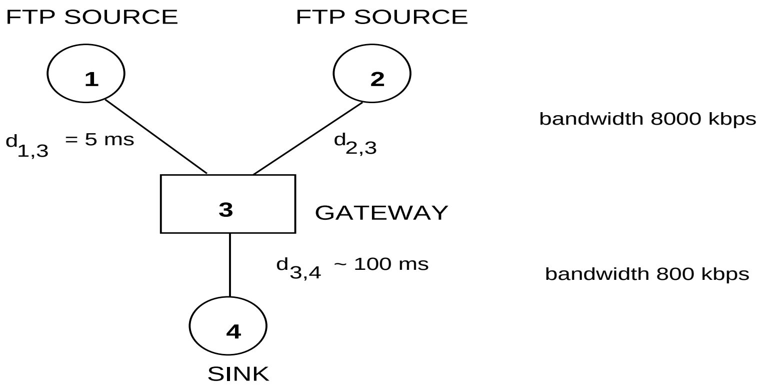

Figure 3: Simulation network.

Our simulator is a version of the REAL simulator (Keshav, 1988) built on Columbia's Nest simulation package (Bacon *et al.*, 1988), with extensive modifications and bug fixes made by Steven McCanne at LBL. The gateways use FIFO queueing, and this section's simulations 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 (Jacobson, 1988).[3] Briefly, there are two phases to the window-adjustment algorithm. A threshold is set initially to half the receiver's advertised window. The connection begins in slow-start phase, and the current window is doubled each roundtrip time until the window reaches the threshold. Then the congestion-avoidance phase is entered, and the current window is increased by roughly one packet each roundtrip time. The window is never allowed to increase to more than the receiver's advertised window, which is referred to as the "maximum window" in this paper.

In 4.3-tahoe BSD TCP, packet loss (a dropped packet) is treated as a "congestion experienced" signal. The source uses the *fast retransmit* procedure to discover a packet loss: if four ACK packets are received acknowledging the same data packet, the source decides that a packet has been dropped. The source reacts to a packet loss by setting the threshold to half the current window, decreasing the current window

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

to one, and entering the slow-start phase. (The source also uses retransmission timers to detect lost packets. However, for a bulk-data connection a packet loss is usually detected by the fast retransmit procedure before the retransmission timer expires.)

Because of the window-increase algorithm, during the slow-start phase the source node transmits two data packets for every ACK packet received. During the congestion-avoidance phase the source generally sends one data packet for every ACK packet received. If an arriving ACK packet causes the source to increase the current window by one, then the source responds by sending two data packets instead of one.

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, with propagation delay $d_{3,4} = 100$ 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 on the slow line, 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.) This small network is not intended to model realistic network traffic, but is intended as a simple model exhibiting traffic phase effects.

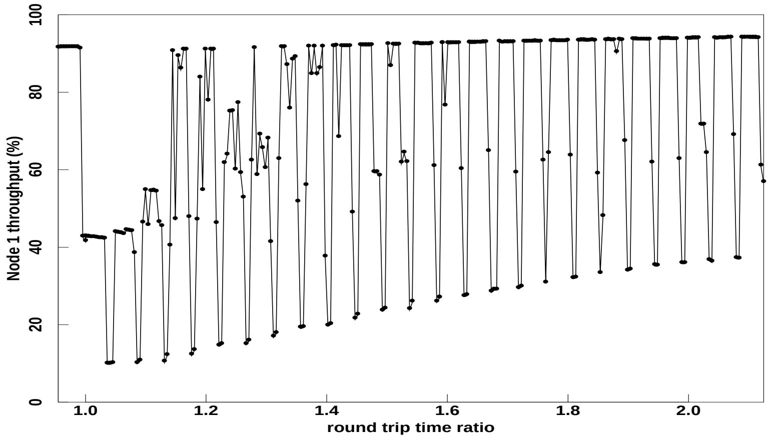

Figure 4: Node 1 throughput as a function of node 2's roundtrip time.

The results of simulations where each source has a maximum window of 32 packets are shown in Figure 4. Each source is prepared to use all of the available bandwidth. Each dot on the graph is the result from one 100 sec. simulation, each run with a different value for $d_{2,3}$, 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

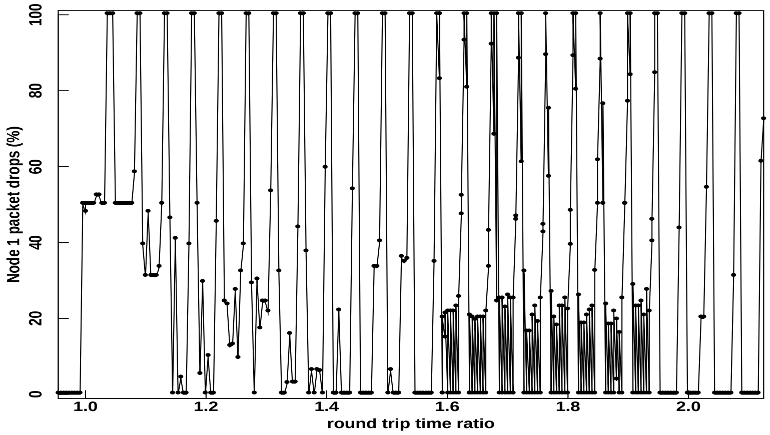

Figure 5: Node 1's share of the packet drops.

reached early in the first 50 seconds. The results for the first 50-second interval in each simulation differ slightly from Figure 4 because the two connections started at slightly different times. The results in Figure 4 would be evident even if each connection had only a few hundred packets to send.

Figure 5 shows node 1's share of the total packet drops for the second 50-second interval in each simulation. In some simulations only node 1 packets were dropped and in other simulations only node 2 packets were dropped. In each 50-second interval the gateway usually dropped between 20 and 40 packets.

Figures 4 and 5 show that this configuration is highly biased: for most values of node 2's link delay, node 1 gets few of the packet drops, and 90% of the available bandwidth. But some node 2 delay values cause node 1 to receive all of the packet drops and cause the node 1 bandwidth share to drop to only 10–20%. These node 2 delay 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 6 shows the result of making a small (4%) change in the delay of the shared link, $d_{3,4}$. 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). For these simulations, $d_{3,4}$ has been changed from 100 ms. to 103.5 ms. This changes node 1's roundtrip time from 221.44 ms. to 228.44 ms.

In most of the simulations in this paper, the sink sends an ACK packet as soon as it receives a data packet. For the simulations in Figure 7, a "delayed-ACK" sink is used, as in current TCP implementations. In other respects, the scenario is the same as that in Figure 4. A delayed-ACK sink sets a 100 ms. timer when a packet

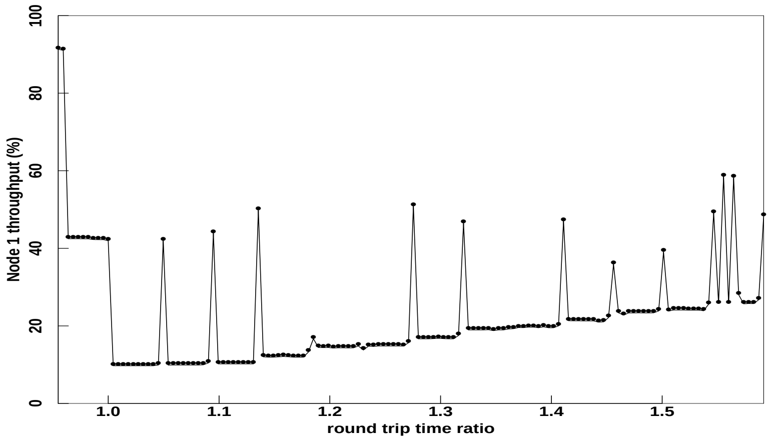

Figure 6: Node 1's throughput with a phase shift ($d_{3,4} = 103.5$ ms).

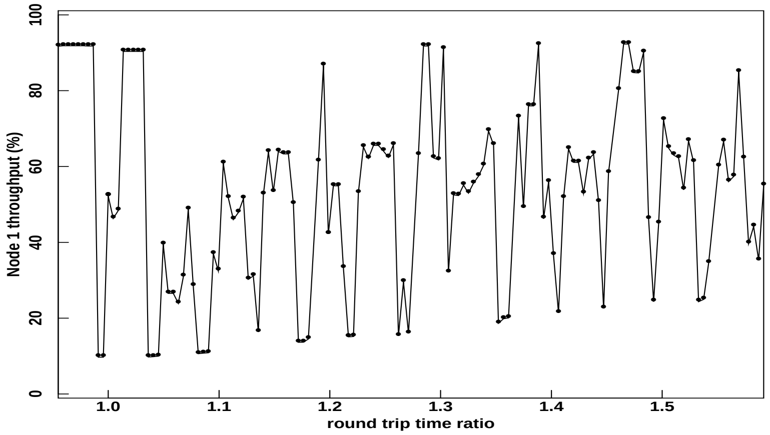

Figure 7: Node 1's throughput with a "delayed-ACK" sink.

is received. If a new packet from the same connection arrives at the sink before the timer expires then an ACK packet is sent immediately acknowledging the packet with the higher sequence number. Otherwise, the ACK packet is sent when the 100 ms. timer expires. As Figure 7 shows, the discriminatory behavior persists in simulations with a "delayed-ACK" sink. For the remaining simulations in this paper, we use sinks that send an ACK packet as soon as they receive a data packet, because the analysis in this case is more straightforward.

It is not necessary to have large maximum windows or few connections to get the bias in Figure 4. The pattern of phase effects remains essentially the same in a simulation with many FTP connections, each with a maximum window of 8 packets. For all of the simulations in this section, a small change in roundtrip times can result in a large change in network performance. Section 2.4 shows traffic phase effects in slightly more complex TCP/IP networks with multiple connections and gateways.

As the simulations in this section show, the pattern of bias does not depend on a particular choice of window sizes, of roundtrip times, or of sink protocols. Other simulations show that this pattern of bias does not depend on details of the TCP transport algorithm; bias due to phase effects is still present with a TCP implementation where the window is decreased by half rather than decreased to one after a packet drop, for example. The pattern of segregation requires a network with a congested gateway that occasionally drops packets and that uses a deterministic strategy in choosing a packet to drop based only on that packet's position in the gateway queue. As section 2.5 shows, this pattern of segregation is likely to be absent in a network with a strong mix of packet sizes, or with an appropriate random component (other than queueing delay) added to the roundtrip time for each packet.

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. 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 queueing delay. 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 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 a packet *arrives* at the gateway queue when the last bit of a packet arrives at the gateway, and a packet *leaves* the gateway 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: roundtrip times r_1 and r_2, bottleneck service time b, queue size max, transmission time 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 max. 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) and 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. This model is 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. In a mild abuse of terminology, we say that this FTP packet "increased" the source window. When the output line becomes free s seconds later, the source sends a second data packet. Without the additional packet, the gateway queue occasionally would have reached size max. Because of the additional packet, the queue at some point fills, and 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: service intervals, 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. Each time the gateway begins transmitting 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. [4] Similarly, when the gateway transmits a node 2 packet, another node 2 packet arrives at the gateway r_2 seconds later, 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. □

This section gives the intuition explaining the behavior of the model; the appendix contains more formal proofs. The three cases discussed correspond to node 2's roundtrip time r_2 being equal to, slightly less than, or slightly greater than node 1's roundtrip time r_1. Node 1 has the same roundtrip time in all three cases, and the same value $t_1 = r_1 \bmod b$. However, node 2's roundtrip time r_2 is different in the three cases, and as a result the value for t_2 changes.

Case 1: In this case the two roundtrip times and the two phases are the same. 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 max. The queue is decremented every b seconds, at the end of each service interval. Line D of Figure 8 shows the service intervals at the gateway. Line C shows the node 1 packets arriving at the gateway, line B shows node 2 packet arrivals, and line A shows the queue. The x-axis shows time, and for line A the y-axis shows the queue size.

[4] $A \bmod b$ is the positive remainder from dividing A by b.

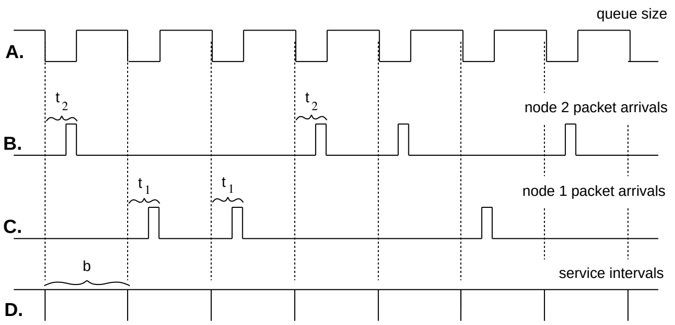

Figure 8: Phase of packet arrivals at the gateway, for $r_1 = r_2$.

For this informal argument, assume for simplicity that s, the time for nodes 1 and 2 to transmit packets on the output line, is zero. When some node increases its window by one, two packets from that node arrive at the gateway back-to-back. The second packet arrives at a full queue and is dropped. Thus with the two equal roundtrip times, after some node increases its window a packet from that node will be dropped at the gateway. $\square$

Case 2: Now consider a network where node 2's roundtrip time r_2 is slightly smaller than r_1. Assume that roundtrip time r_2 is smaller than r_1 by at least t_1 and by at most b, the bottleneck service time. We have two periodic processes with slightly different periods. The packet arrivals are shown in Figure 9. (The labels for Line D are explained in the proofs in the appendix.) It is no longer true that exactly one packet arrives at the gateway in each service interval. In Figure 9, the packets from node 2 arrive slightly earlier than their arrival time in Figure 8. When a node 2 packet arrives at the gateway following a node 1 packet, the two packets arrive in the same service interval.

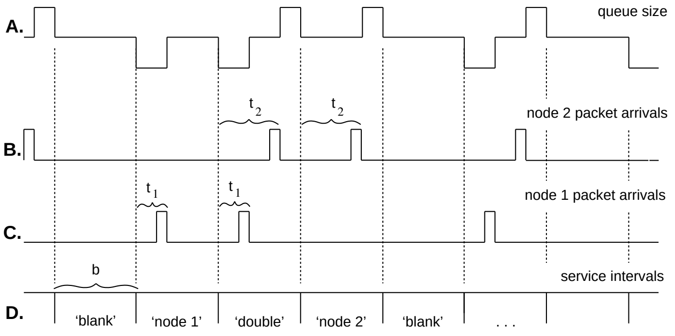

Figure 9: Phase of packet arrivals at the gateway, for $r_2 < r_1$.

From Figure 9, in a service interval with both a node 1 and a node 2 packet arrival, a node 1 packet arrives at time t_1, followed at time $t_2 > t_1$ by a node 2 packet. During the period when windows are fixed and the queue occasionally reaches size max, only node 2 packets increase the queue size to max. 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 max. The next node 2 packet that arrives at the gateway will be dropped. Claim 1 describes this behavior in detail in the appendix. □

Case 3: A similar case occurs if roundtrip time r_2 is slightly greater than r_1. Assume that roundtrip time r_2 is larger than r_1 by at least $b - t_1$ and by at most b, the bottleneck service time. The packet arrivals are shown in Figure 10. When a node 1 packet arrives at the gateway after a node 2 packet, both packets arrive in the same service interval. During the period when windows are fixed and the queue occasionally reaches size max, only node 1 packets cause the gateway queue to increase to max. When some connection's window is increased, the gateway always drops a node 1 packet. □

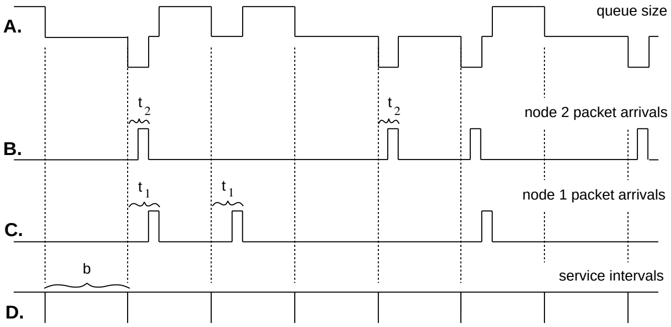

Figure 10: Phase of packet arrivals at the gateway, for $r_2 > r_1$.

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.

Claims 1 through 6 in the appendix describe the behavior of the model when the two roundtrip times differ by at most b, the bottleneck service time, as is discussed in the Cases 1 through 3 above. There is a range for r_2 where the gateway only drops node 2 packets, followed by a range for r_2 where the gateway drops both node 1 and node 2 packets, followed by a range for r_2 where the gateway only drops node 1 packets.

Claims 7 through 9 in the appendix describe the pattern of packet drops when the two roundtrip times differ by more than b. For $t_2 < t_1 - s$, the gateway always drops a node 1 packet when node 1 increases its window, and for $t_1 + s < t_2 < b - s$, the gateway always drops a node 2 packet when node 2 increases its window.

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 *max* and that ends when one of the connections reduces its window, decreasing the rate of packets arriving at the gateway. The drop period is similar to the *congestion epoch* defined elsewhere (Shenker *et al.*, 1990). If the maximum windows have not all been reached, then after the queue first reaches size *max*, 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 between one and two roundtrip times. □

For the simulations in Figure 4, node 1 packets arrive at the gateway early in the current service interval, after 0.144 of the current service interval. However, for the simulations in Figure 6 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.

For simulations with a “delayed-ACK” sink, the proofs in this section no longer hold. In this case, the ACK for some packets is delayed at the sink for 100 ms. For the simulations in Figure 7, this delay happens to be an integer multiple of the bottleneck time b. In these simulations, the use of a “delayed-ACK” sink changes the exact pattern of packet arrivals at the gateway, but node 1 packets still arrive at the gateway at a fixed time t_1 after the start of some service interval, and node 2 packets still arrive at the gateway at a fixed time t_2 after the start of some service interval. The pattern of segregation is changed slightly with a “delayed-ACK” sink, but the segregation still varies sharply as a function of the roundtrip times. Traffic phase effects can still be observed in simulations with a “delayed-ACK” sink where the delay is not an integer multiple of b.

2.3 Phase effects with DECbit congestion avoidance

In this section we demonstrate phase effects in an ISO TP4 network using DECbit congestion avoidance (Ramakrishnan and Jain, 1990). In the DECbit congestion avoidance scheme, the gateway uses a *congestion-indication* bit in packet headers to provide feedback about congestion in the network. When a packet arrives at the gateway the gateway calculates the average queue length for the last (busy + idle) period plus the current busy period. (The gateway is *busy* when it is transmitting packets, and *idle* otherwise.) When the average queue length is greater than one, then the gateway sets the congestion-indication bit in the packet header of arriving packets.

The source uses window flow control, and updates its window once every two roundtrip times. If at least half of the packets in the last window had the congestion indication bit set, then the window is decreased exponentially. Otherwise, the window is increased linearly.

Figure 11 shows the results of simulations of the network in Figure 3. Node 2’s

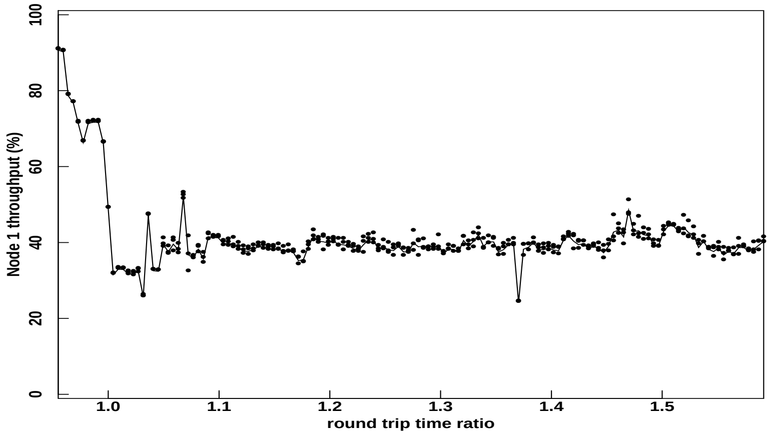

Figure 11: Node 1's throughput with the DECbit scheme.

roundtrip time is varied by varying $d_{2,3}$. (The roundtrip time for node 1 packets is still 221.44 ms.) These simulations use the implementation of the DECbit scheme in the REAL simulator (Keshav, 1988). Each simulation was run for 200 seconds. Figure 11 represents each 50-second interval (excluding the first 50-second interval) by a dot showing node 1's throughput for that interval. The line shows node 1's average throughput.

For the simulations where the two roundtrip times differ by less than the bottleneck service time, the total throughput for nodes 1 and 2 is close to 100% of the link bandwidth. For the simulations where the two roundtrip times differ by more than the bottleneck service time, the throughput for node 2 is similar to the throughput for node 1, and the total throughput is roughly 80% of the link bandwidth. In this case, when the gateway drops packets, it generally drops packets from both node 1 and from node 2.

The traffic phase effects are present in Figure 11 only for those simulations where node 1 and node 2's roundtrip times differ by less than 10 ms., the bottleneck service time. For other roundtrip times with this scenario, the DECbit congestion avoidance scheme avoids the phase effects in simulations with TCP and Drop Tail gateways. When the two roundtrip times differ by less than the bottleneck service time, the network bias is in favor of the connection with the slightly longer roundtrip time. When node 1 and node 2's roundtrip times differ by less than the bottleneck service time and the current windows are both less than the pipe size then node 1 and node 2 packets are not interleaved at the gateway. The gateway transmits a window of node 1 packets, followed by a window of node 2 packets. In the next few paragraphs we give some insight into the phase effects exhibited by the DECbit congestion avoidance scheme under these conditions.

Case 1: Figure 12 shows packet arrivals at the gateway when the roundtrip times r_1 and r_2 are equal. Assume that one roundtrip time earlier, the gateway transmitted a window of two node 1 packets, immediately followed by a window

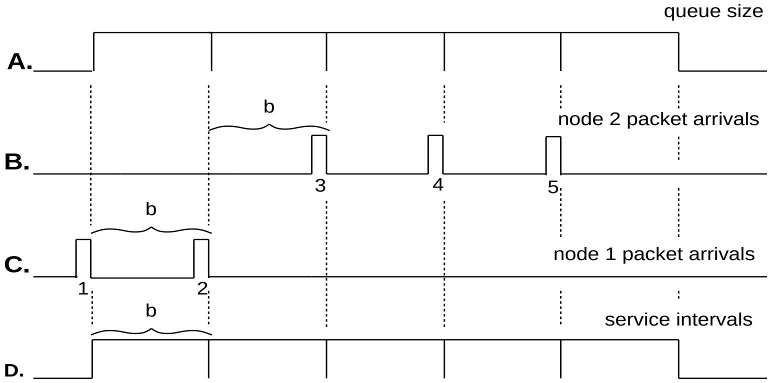

Figure 12: Packet arrivals at the gateway, for $r_1 = r_2$.

of three node 2 packets. (These small window sizes are chosen for illustration purposes only.) The five packets leave the gateway at regular intervals of b seconds, and one roundtrip time later, assuming that the window has not changed, five more packets arrive at the gateway at regular intervals of b seconds. Line C in Figure 12 shows the node 1 packet arrivals, and line B shows the node 2 packet arrivals. Packet numbers are shown for each packet. The gateway is idle when packet #1 arrives at the gateway. Packet #1 is immediately serviced, the queue goes to size one, and the current busy period begins. (In the DECbit algorithm the queue is considered to be of size one when a packet is being transmitted on the output line.) The gateway takes b seconds to transmit each packet on the output line; Line D shows these b-second service intervals for the five packets. Packets also arrive at the gateway once every b seconds. Line A shows the queue size. For each packet, the average queue size during the current busy period is 1. □

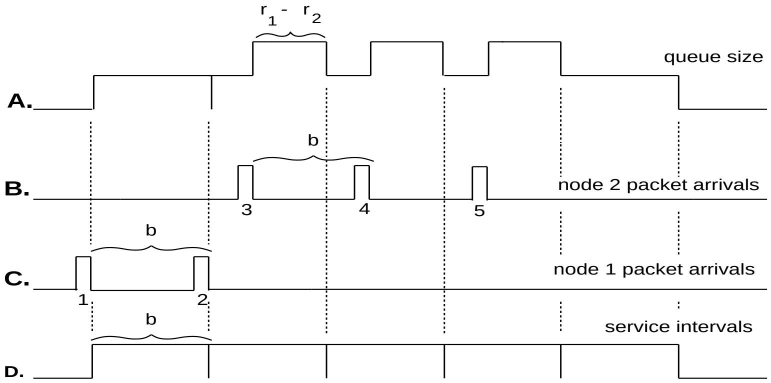

Figure 13: Packet arrivals at the gateway, $r_2 < r_1$.

Case 2: Figure 13 shows the packet arrivals at the gateway when roundtrip time r_2 is less than r_1 by at most b, the bottleneck service time. Two packets from node 1 arrive at the gateway, followed $b - (r_1 - r_2)$ seconds later by the first of three packets from node 2. For packet #2 and packet #3, the average queue size in the current busy period is 1. For packet #i, for $i \geq 4$, the average queue size is

$$\frac{(i-1)*b+(i-4)(r_1-r_2)}{(i-1)*b-(r_1-r_2)} > 1.$$

The average queue size for the current busy period increases as the packet number increases.

The gateway decision to set the congestion indication bit for a packet is based on the average queue size for the last (busy + idle) cycle as well as on the average queue size for the current busy cycle. If the gateway sets the congestion indication bit for one packet in a busy cycle, then the gateway will set the congestion indication bit for all succeeding packets in that busy cycle. Therefore, node 2 packets are more likely to have the congestion indication bit set than are node 1 packets. As a result, node 2 is more likely than node 1 to decrease its current window. □

It is not our intention in this paper to consider whether these traffic phase effects are likely to occur in actual networks. Our intention is to show that traffic phase effects can occur in unexpected ways in packet-switched networks (or in network simulations) with periodic traffic and a deterministic gateway driven by the gateway backlog. The phase effects in this section are similar to the unfairness observed in a testbed running the DECbit congestion scheme with two competing connections (Wilder *et al.*, 1991).

The behavior with the DECbit congestion scheme in Figure 11 differs from the behavior with TCP and Drop Tail gateways in Figure 4 in part because the two schemes use different methods to detect congestion in the network. A TCP/IP network with Drop Tail gateways detects congestion when a packet is dropped at the gateway; as this paper shows, this can be sensitive to the exact timing of packet arrivals at the gateway. A network using the DECbit congestion avoidance scheme detects congestion by computing an average queue size over some period of time. This is less sensitive to the exact timing of packet arrivals at the gateway. As Section 2.6 shows, phase effects are also avoided in simulations using TCP with Random Drop gateways instead of with Drop Tail gateways. In Section 3.2 we briefly discuss the performance of TCP/IP networks with Random Early Detection gateways, which are similar to the gateways in the DECbit scheme in that they detect congestion by computing the average queue size.

2.4 Phase effects in larger TCP/IP networks

In this section we show that traffic phase effects can still be present in TCP/IP networks with three or more connections or with multiple gateways. The phase effects in networks with three or more connections are somewhat more complicated than the phase effects in networks with only two connections, and we do not attempt an analysis. In the section we discuss one network with multiple connections and multiple gateways where a change in propagation delay along one edge of the network significantly changes the throughput for a connection in a different part of the network.

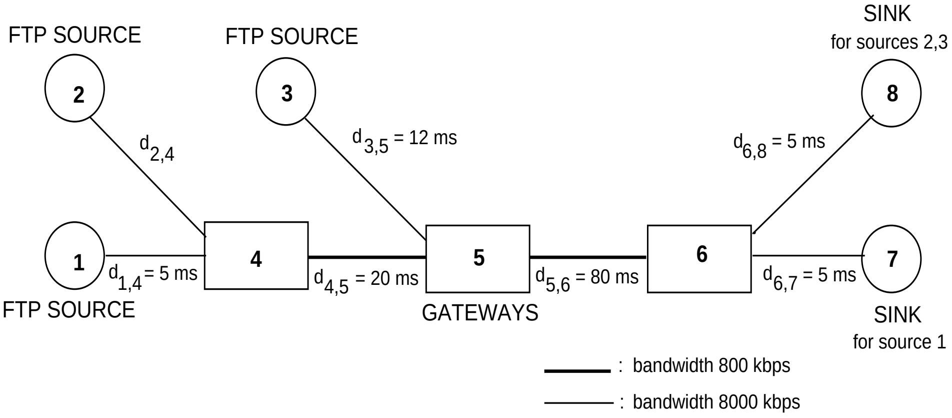

Figure 14: Simulation network with multiple gateways.

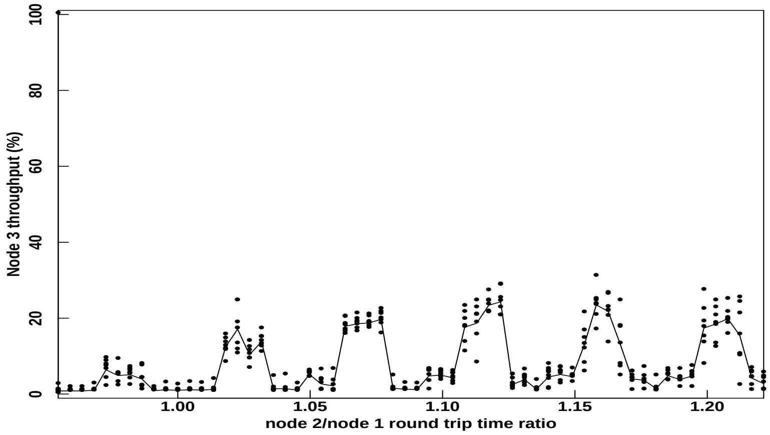

Figure 15: Node 3's throughput as a function of node 2's roundtrip time.

Figure 14 shows a network with three connections and multiple gateways. Figure 15 shows the throughput for node 3 as the propagation delay $d_{2,4}$ is varied, varying node 2's roundtrip time. Node 3's throughput is plotted as a percentage of the maximum possible throughput through gateway 5. Changing node 2's roundtrip time changes the phase of node 2 packet arrivals at gateway 5. This changes the throughput for node 3 as well as for nodes 1 and 2. The network in Figure 14 exhibits significant traffic phase effects for all three connections.

2.5 Adding telnet traffic

In this section, we explore the extent to which patterns of bias persist in TCP/IP networks in the presence of (randomly-timed) telnet traffic. For the simulation network in Figure 16, telnet nodes send fixed-size packets at random intervals drawn from an exponential distribution. In this section we show that the bias due to traffic phase effects is strongest when all of the packets in the congested gateway queue are of the same size. The simulations in this section show that significant bias remains when roughly 15% of the packets are 1000-byte telnet packets, and also when roughly 3% of the packets are 40-byte telnet packets. However, when 15% of the packets are 40-byte telnet packets, the bias due to traffic phase effects is largely eliminated. This means that traffic phase effects are unlikely to be observed in networks or simulations with a strong mix of packets sizes through each congested gateway.

The second half of this secion shows that traffic phase effects are unlikely to be observed in a network where there is a random component of the roundtrip time (other than queueing delay) that is often as large as the bottleneck service time.

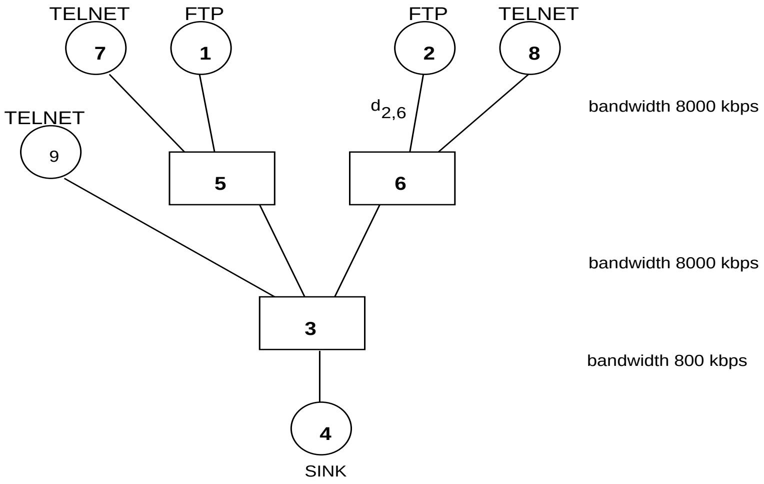

Figure 16: Simulation network with telnet and FTP nodes.

Figure 16 shows a 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 17 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, roughly 15% of the packets are from the telnet nodes. Because the results are not deterministic, 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.

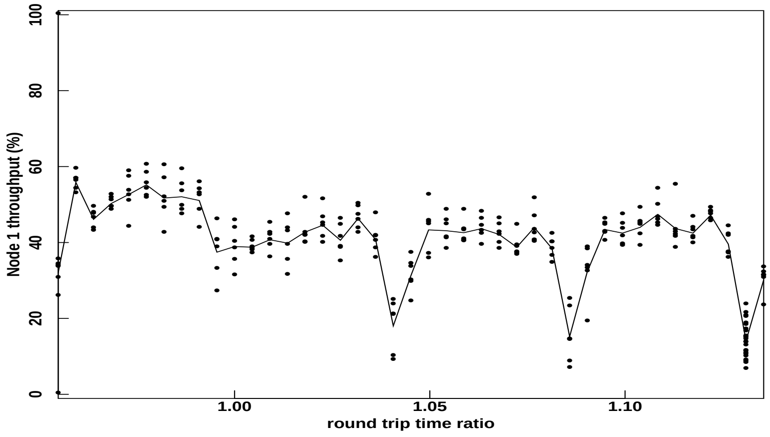

Figure 17: Node 1's throughput, with 1000-byte telnet packets as 15% of packets.

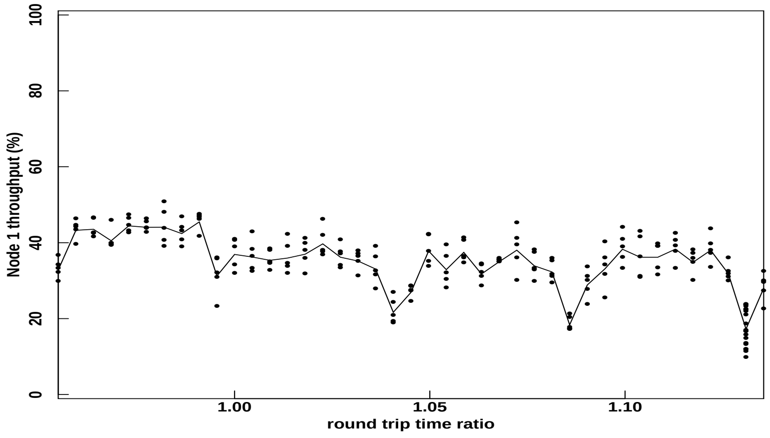

Figure 18: Node 1's throughput, with 1000-byte telnet packets as 30% of packets.

The solid line gives the average throughput for node 1, averaged over all of the simulations. Figure 18 shows results from simulations where roughly 30% of the packets are from the telnet nodes. As Figure 18 shows, there is still discrimination for some roundtrip ratios even from simulations where roughly 30% of the packets through the gateway are 1000-byte telnet packets.

The results are different when each telnet node sends 40-byte packets instead of 1000-byte packets. When roughly 3% of the packets at the gateway are 40-byte telnet packets, the pattern of discrimination still holds. However in simulations in Figure 19 roughly 15% of the packets at the gateway are 40-byte telnet packets, and the pattern of bias is largely broken.

If 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 $r_i \bmod b$ after the start of some service interval, for $i \in \{1, 2\}$. These fixed phase relationships

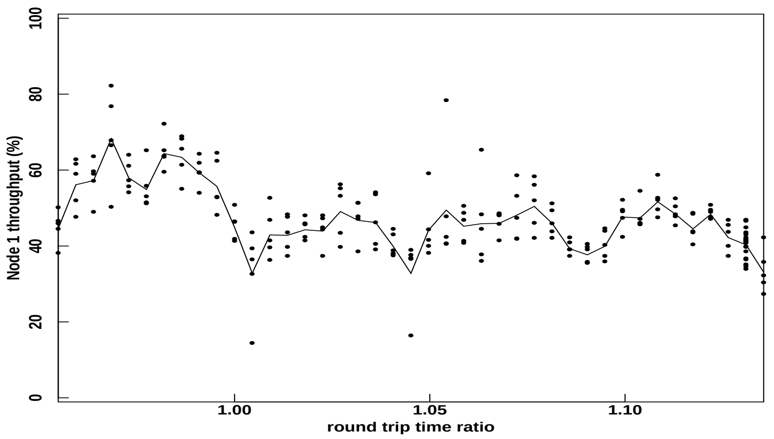

Figure 19: Node 1's throughput, with 40-byte telnet packets as 15% of packets.

no longer hold when the gateway queue contains packets of different sizes. This section suggests that phase effects are unlikely to be found in a nondeterministic network with a strong mix of packet sizes.

The node processing times in the simulations described so far have 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 helps to break up the fixed pattern of packet arrivals at the gateway.

In simulations where each source node uses a time uniformly chosen between zero and half the bottleneck service time to prepare each FTP packet after an ACK packet is received, the pattern of phase effects is changed somewhat, but is still present. However, in simulations where each source node uses a time uniformly chosen between zero and the bottleneck service time to prepare each FTP packet after an ACK packet is received, the pattern of phase effects is largely eliminated. In general, when the goal of network simulations is to explore properties of network behavior unmasked by the specific details of traffic phase effects, a useful technique is to add a random packet-processing time in the source nodes that ranges from zero to the bottleneck service time.

As Figure 20 shows, the pattern of discrimination is still present when 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. Figure 21 shows the results of simulations when each source node requires a time uniformly chosen between 0 and 10 ms., one bottleneck service time, to prepare each FTP packet. Each packet now arrives at the gateway at a random time with respect to the start of the current service interval. As Figure 21 shows, the pattern of segregation is muted. However, in simulations where the two roundtrip times differ by less than the bottleneck service time the segregation is not completely eliminated. For these

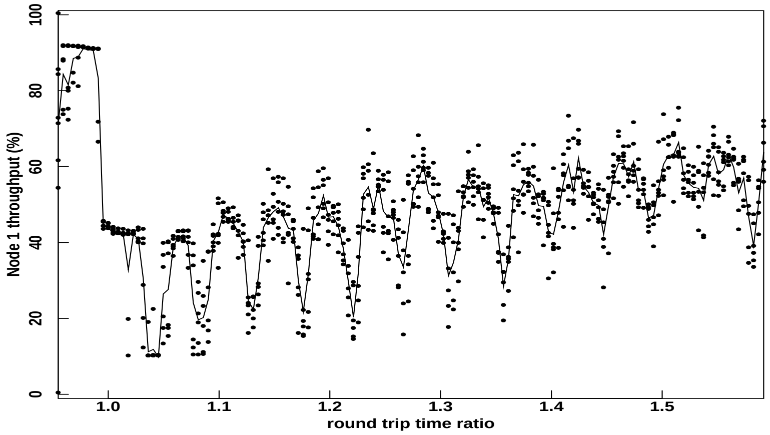

Figure 20: Node 1's throughput, with random processing time from 0 to 5 ms.

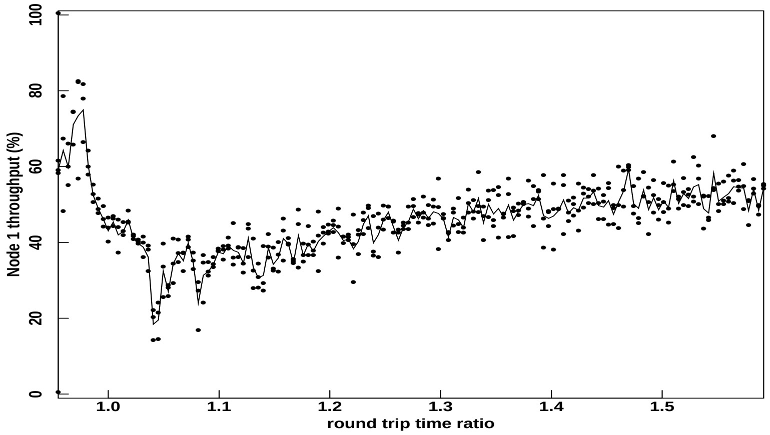

Figure 21: Node 1's throughput, with random processing time from 0 to 10 ms.

parameters there is little interleaving of node 1 and node 2 packets at the gateway.

2.6 Phase effects and Random Drop gateways

This section shows that Random Drop gateways eliminate the network bias due to traffic phase effects. With Random Drop gateways, when a packet arrives at the gateway and the queue is full, the gateway randomly chooses a packet from the gateway queue to drop. 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 max. 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 $max + 1$. (The pseudo-random numbers could be chosen in advance.) The gateway drops the nth packet in the gateway queue. Consider a queue that overflows when a node 1 packet arrives at the gateway immediately following 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. (A variant not investigated in this paper is a Random Drop gateway that measures the queue in bytes rather than in packets. With this variant a packet's probability of being dropped is proportional to that packet's size in bytes.)

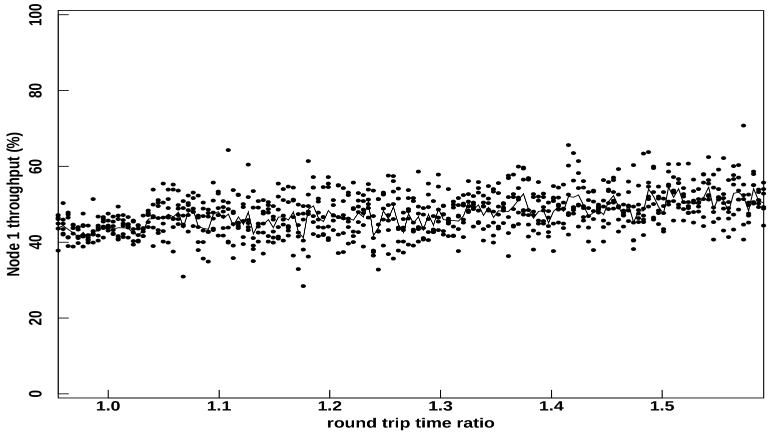

Figure 22: Node 1's throughput with Random Drop gateways.

Figure 22 shows the results from simulations using a Random Drop gateway in the network shown in Figure 3. These simulations differ from the simulations in Figure 4 only in that the network uses a Random-Drop instead of the Drop-Tail gateway. In Figure 22, each dot represents the throughput for node 1 in one 50-second interval of simulation. For each node 2 roundtrip time, six 50-second simulation intervals are shown. The solid line shows the average throughput for node 1 for each roundtrip time ratio. As Figure 22 shows, Random Drop eliminates the bias due to traffic phase effects.

For the simulations in Figure 22 there are roughly 30 packet drops in each 50-second interval of simulation. If the queue contains an equal numbers of packets from node 1 and node 2 each time it overflows, the probability that one node receives all 30 packet drops is 2^{-29} (roughly one in a billion). In this case, 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. This pattern of discrimination could be controlled with Fair Queueing gateways (Demers *et al.*, 1990), for example, where the gateway maintains separate queues for each connection. However, the use of randomization allows Random Drop gateways to break up the bias caused by traffic phase effects with a stateless, low-overhead algorithm that is easily implemented and that scales well to networks with many connections.

The simulations in Figure 22 work well because, for 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. In simulations with two connections with the same roundtrip time and with maximum windows less than the pipe size, the gateway always transmits a window of node 1 packets followed by a window of node 2 packets (Shenker *et al.*, 1990). 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.

2.7 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 possible in networks with Drop Tail gateways. This discrimination can be affected by small changes in either the phase of the FTP connections or the maximum queue size at the bottleneck. We show that the use of Random Drop gateways eliminates discrimination against telnet traffic.

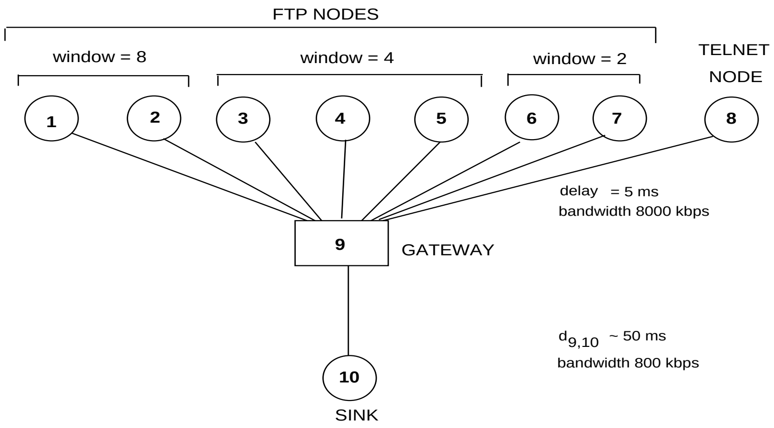

Figure 23: Simulation network with FTP and telnet nodes.

The simulation network in Figure 23 has 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.

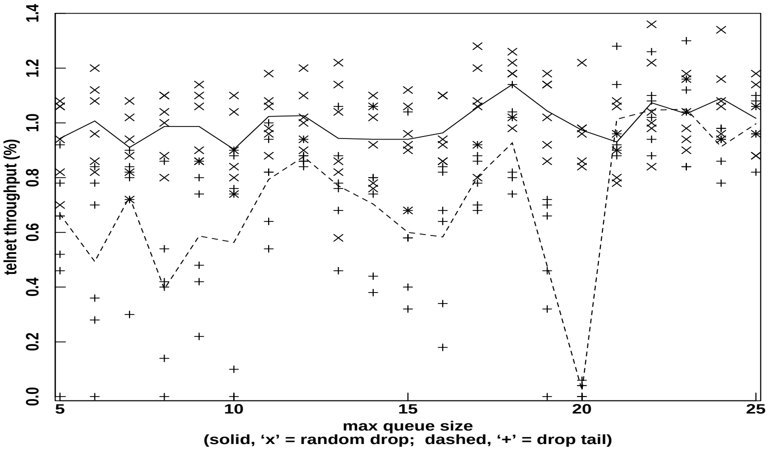

Figure 24: Telnet throughput in Set A.

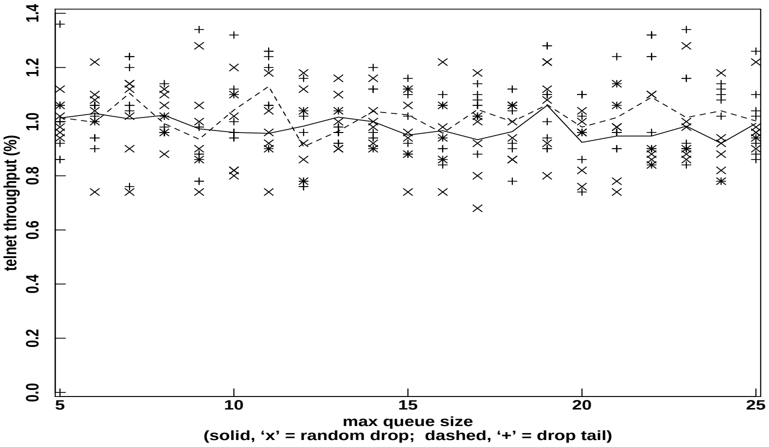

Figure 25: Telnet throughput in Set B.

We compare simulations with Random Drop and with Drop Tail gateways. For the simulations in Set A in Figure 24, $d_{9,10} = 50$ ms., for a roundtrip time in the absence of queues of 121.44 ms. For the simulations in Set B in Figure 25, $d_{9,10}$ =53.7 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 "×" 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 the results with Drop Tail gateways.

For the simulations in Set A, 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 and 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.

For the simulations in Set B, 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.

These simulations 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 showing two telnet connections shut out by six FTP connections (Demers *et al.*, 1990), for example, should be interpreted with this sensitivity to the exact network parameters in mind.

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 Shared biases of Random Drop and Drop Tail gateways

The first half of this paper showed that networks with Drop Tail gateways can be sensitive to traffic phase effects, and that these traffic phase effects can be largely eliminated with Random Drop gateways. The second half of the paper discusses the bias against bursty traffic and the bias against connections with longer roundtrip times that are shared by networks with Drop Tail and with Random Drop gateways. The bias against bursty traffic can be eliminated by Random Early Detection gateways, where the probability that a packet is dropped from a connection is proportional to that connection's share of the throughput. The bias against connections with longer roundtrip times can be eliminated by a modified TCP window-increase algorithm where each connection increases its throughput rate (in pkts/sec) by a constant amount each second.

3.1 Previous research on Random Drop gateways

The reported benefits of Random Drop gateways over Drop Tail gateways (Hashem, 1989) include fairness to late-starting connections and slightly improved throughput for connections with longer roundtrip times. In simulations of a network with two connections, one local and one long-distance, with large maximum windows and a shared gateway, 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 reported shortcomings of the Random Drop algorithm (Hashem, 1989) 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.

Early Random Drop gateways have been investigated as a mechanism for congestion avoidance as well as for congestion control (Hashem, 1989). In that implementation of Early Random Drop gateways, the gateway drops each packet with a fixed probability 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, Hashem suggests that they have a better chance than Random Drop gateways of targeting aggressive users. Hashem 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. Hashem recommends additional work on Early Random Drop gateways. The conclusions on Random Drop gateways are 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 (Hashem, 1989, p.103)." We examine these performance biases in more detail in the next two sections.

Zhang uses simulations to evaluate Random Drop gateways (Zhang 1989). Zhang 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. Zhang remarks that in the simulations, 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 (Zhang 1989, p.99)." We examine this problem in Section 3.3.

The Random Drop and the Drop Tail gateway algorithms are compared in a measurement study of a network with local and long distance traffic, with several congested gateways (Mankin, 1990). 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 topologies, the throughput for the longer connection was better with Random Drop gateways, and for other topologies the throughput was better with Drop Tail gateways. In both cases, connections with longer roundtrip times and small windows received a disproportionate number of dropped packets. As Section 3.2 explains, these results should be interpreted keeping traffic phase effects in mind. Mankin reports that "Random Drop Congestion Recovery improves the fairness of homogeneous connections that have the same bottleneck, but beyond that, it has limited value (Mankin, 1990, p.6)."

The Gateway Congestion Control Survey by the IETF Performance and Congestion Control Working Group (Mankin and Ramakrishnan, 1991) discusses the research results on Random Drop gateways. 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 (Mankin and Ramakrishnan, 1991, p.8)." In this paper, we suggest that, in comparison to Drop Tail gateways, Random Drop gateways offer significant advantages and no significant disadvantages.

Demers et al. briefly compare Fair Queueing gateways with Random Drop gateways (Demers *et al.*, 1990). They report that Random Drop gateways "greatly alleviate" the problem of segregation with Drop Tail gateways, but that Random Drop gateways 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 has been that Random Drop gateways are biased against connections with longer roundtrip times. As some of the papers mention, this bias is shared by Drop Tail gateways. This section examines the bias of Random Drop and of Drop Tail gateways against bursty traffic. [5] The following section examines the bias of the TCP window increase algorithm

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

against connections with long roundtrip times and large maximum windows.

One reason to examine the bias of Drop Tail gateways against bursty traffic is that, due in part to traffic phase effects, the bias of Drop Tail gateways can change significantly with small changes to network parameters. 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.

The main reason to compare the bias of Drop Tail and of Random Drop gateways against bursty traffic is that the poor performance of Random Drop gateways for connections with long roundtrip times has been cited as one reason to avoid the use of Random Drop gateways with mixed traffic. This section shows that the bias against bursty traffic is more severe with Drop Tail gateways for some parameters, and more severe with Random Drop gateways for other parameters. In general, the bias of Random Drop gateways against bursty traffic is no worse than the bias of Drop Tail gateways. In either case, this bias occurs because the contents of the gateway queue when the queue overflows are not necessarily representative of the average traffic through the queue.

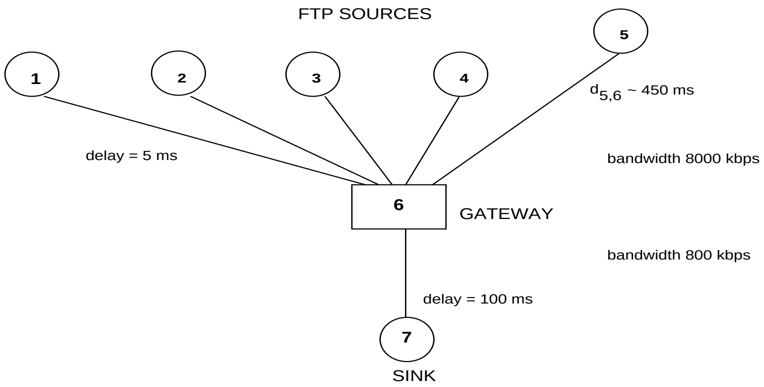

Figure 26: Simulation network with five FTP connections.

We consider simulations of the network in Figure 26, with a maximum window of 8 packets for each connection. 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.

Figures 27 and Figure 28 show the results of simulations for the network with four short FTP connections and one long FTP connection. The simulations were

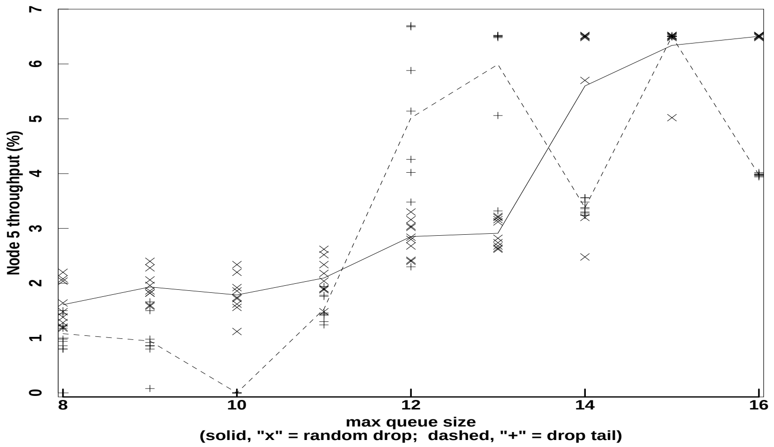

Figure 27: Node 5's throughput with Set A.

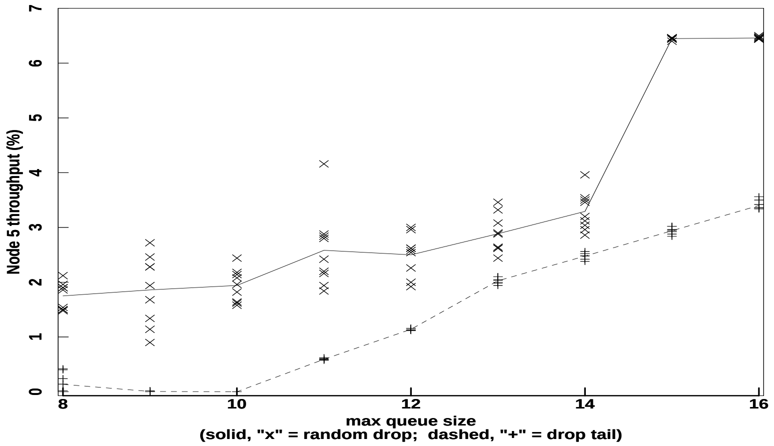

Figure 28: Node 5's throughput with Set B.

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 the simulations in Set A, $d_{5,6} = 449.4$ ms., and node 5 packets arrive at the gateway at the start of a service interval. For the simulations in Set B, $d_{5,6} = 453$ ms., and node 5 packets arrive at the gateway towards the end of a service interval. With Drop Tail gateways the throughput for node 5 is affected by small changes in phase for node 5 packets; this is not the case with Random Drop gateways. Node 5's roundtrip time differs by less than one bottleneck service time in the two sets of simulations. In both cases, node 5's roundtrip time is more than five times larger than the other roundtrip times.

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 shows the average throughput with Drop Tail gateways.

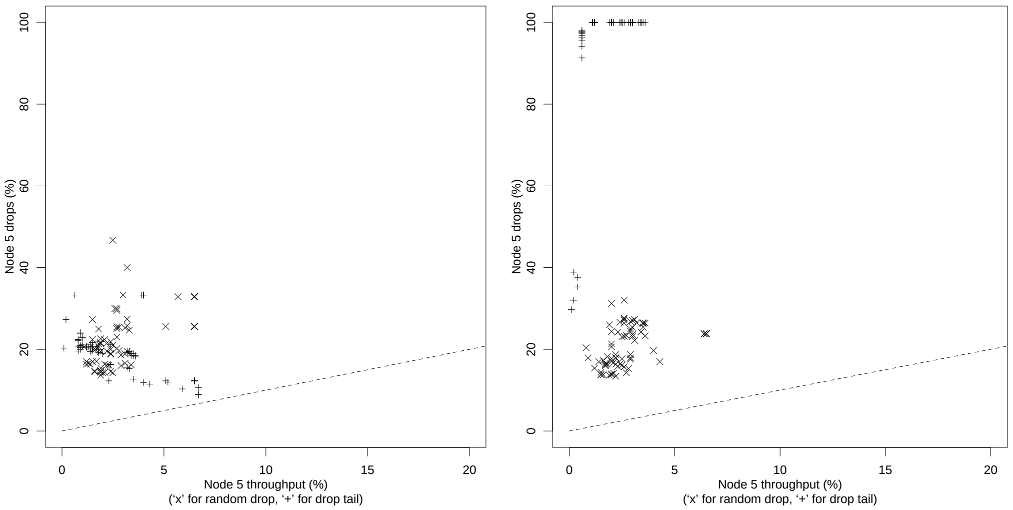

Figure 29: Packet drops vs. throughput for node 5, for Sets A and B.

Because the node 5 packets are transmitted in a loose cluster, the queue is more likely to overflow when it contains packets from node 5. 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.

With Drop Tail gateways and 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 in Set A, because node 5 packets arrive at the gateway at the start of a service interval, these packets are unlikely to arrive at a full queue. For simulations in Set B node 5 packets arrive towards the end of the service interval and are more likely to be dropped. Thus for Drop Tail gateways node 5 receives better throughput for the simulations in Set A.

Note that with Random Drop gateways, node 5 is never completely shut out. However, in simulations with Drop Tail gateways and a maximum queue of 10, node 5 is completely shut out. With this queue size, the gateway queue is full but not overflowing before packets from node 5 arrive. For Drop Tail simulations in Set B node 5 packets are always dropped when they arrive at the gateway. For Drop Tail simulations in Set A the explanation is slightly more complicated, but the results are similar. In this case, node 5 packets are likely to be dropped when 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 earlier results 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 (Mankin, 1990).

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. The performance problems for nodes with long roundtrip times and small windows are neither cured, nor significantly worsened, by Random Drop gateways. Figure 3.2 shows that with both Drop Tail and Random Drop gateways, node 5 receives a disproportionate share of packet drops. The chart on the left shows the results from the simulations in Set A, and the chart on the right shows the simulations in Set B. Each mark represents the result from one 50-second simulation. The x-axis shows node 5's average throughput (as a percentage of the total throughput through that gateway) and the y-axis shows node 5's average number of packet drops (as a percentage of the total number of packet drops). The dashed line shows the position of points where node 5's share of the drops equals node 5's share of the throughput. These figures only show those simulations with at least 20 packet drops in a 50-second simulation. Tor the simulations in Set B with Drop Tail gateways node 5 gets from 1% to 4% of the throughput and up to 100% of the packet drops. This is unfair behavior by any definition of unfairness.

The throughput for bursty traffic can be improved with gateways such as Random Early Detection gateways, which detect incipient congestion early and which do not have a bias against connections with bursty traffic. With our implementation of Random Early Detection gateways the gateway computes the average size for the output queue using an exponential weighted moving average. When the average queue size is less than the minimum threshold, no packets are dropped. When the average queue size is greater than the maximum threshold, every arriving packet is dropped, ensuring that the average queue size does not exceed the maximum threshold. When the average queue size is between the minimum threshold and the maximum threshold, each arriving packet packet is dropped with probability p_a, where p_a is a linear function of the average queue size a. (When the average queue size a equals the minimum threshold, p_a is set to zero, and when the average queue size equals the maximum threshold, p_a is set to 0.1.) In order to avoid dropping two packets in quick succession, after the gateway drops a packet the gateway waits for a maximum of $1/p_a$ and 100 packets before allowing another packet to be dropped. With a Random Early Detection gateway, a node that transmits packets in a cluster does not have a disproportionate probability of having a dropped packet.

The result of simulations with Random Early Detection gateways are shown in Figure 30, with $d_{5,6} = 450$ ms. The x-axis shows the minimum threshold (in packets) for the Random Early Detection gateway, and the y-axis shows the average throughput for node 5. The maximum threshold is 5 greater than the minimum threshold. 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 15 packets, as in the simulations with Drop Tail and with Random Drop gateways. Figure 31 shows node 5's percentage of the total packet drops,

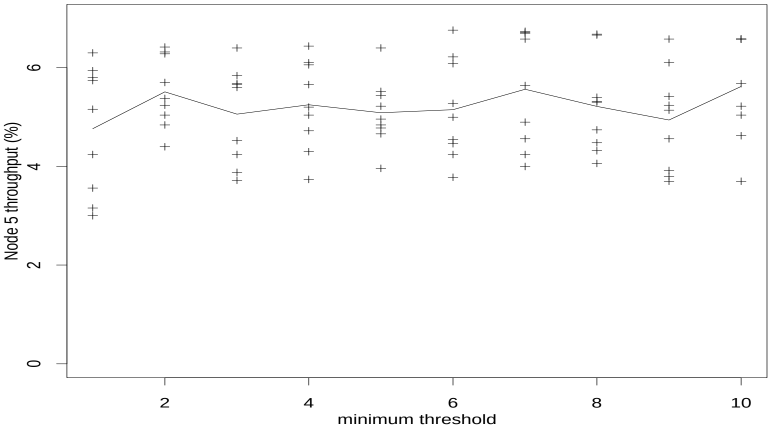

Figure 30: Node 5 throughput with Random Early Detection gateways

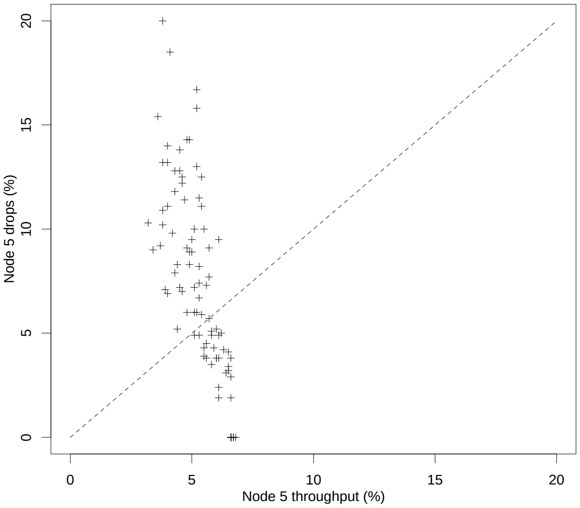

Figure 31: Packet drops vs. throughput for node 5 with Random Early Detection gateways.

plotted against node 5's percentage of the total throughput. Node 5 gets from 3% to 7% of the throughput and from zero to 20% of the packet drops. These simulations suggest 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

The bias against connections with longer roundtrip times and large maximum windows in networks with TCP congestion control is similar for Drop Tail or for Random Drop gateways. This bias results from the TCP window increase algorithm, not from the gateway packet-dropping algorithm. With the window modification algorithm in 4.3 BSD TCP, in the absence of 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. This section examines this bias and discusses possible alternatives to the current window increase algorithm.

This section shows 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 bandwidth-delay 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 roundtrip time. Thus node 2's maximum window ranges from 22 packets to more than 180 packets. The simulations with Drop Tail gateways are shown in Figure 32, and the simulations with Random Drop gateways are shown in Figure 33. 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.

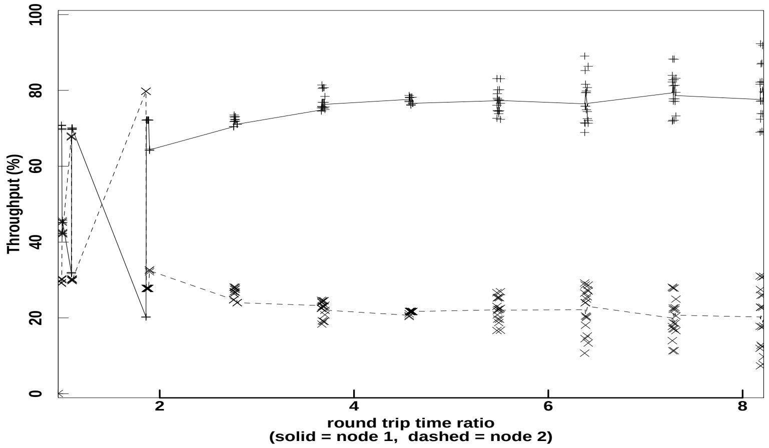

Figure 32: Node 1 and node 2 throughput with Drop Tail gateways.

For each cluster of simulations with Drop Tail gateways in Figure 32, 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. For simulations with both Drop Tail and Random Drop gateways, 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.

As Figure 34 shows, the performance is not significantly improved by the use of Random Early Detection gateways. For the simulations with Random Early Detection gateways, the source and sink nodes use the Fast Recovery algorithm in

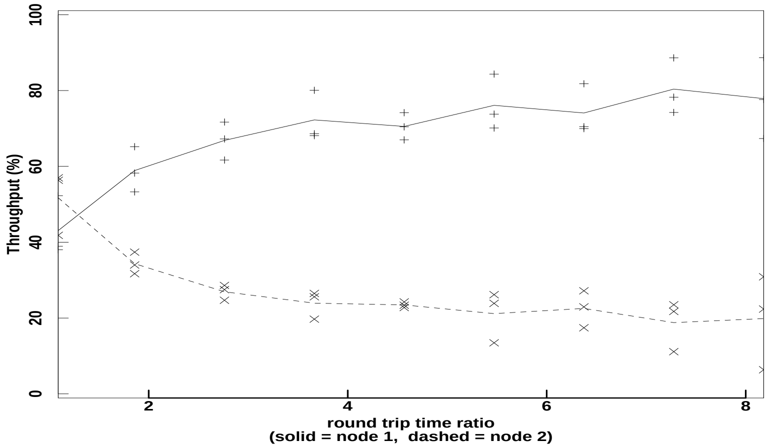

Figure 33: Node 1 and node 2 throughput with Random Drop gateways.

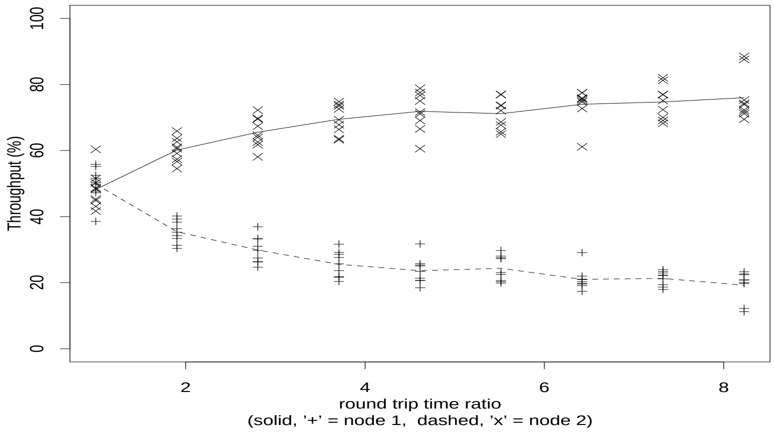

Figure 34: Node 1 and node 2 throughput with Random Early Detection gateways.

4.3-reno BSD TCP (Jacobson, 1990), designed for improved performance over long high-speed links. With the Fast Recovery algorithm, when a packet is dropped the current window is effectively cut in half rather than reduced to one. The simulations in Figure 34 also use Selective Acknowledgement sinks. Each acknowledgement specifies not only the last sequential packet received for that connection but also the highest sequence number received, along with a list of the sequence numbers of the missing packets. For the simulations in Figure 34, the maximum queue is 60 packets, the minimum threshold for the average queue size is 5 packets, and the maximum threshold is 15 packets. As Figure 34 shows, even with Random Early Detection gateways and these improvements to TCP there is a strong network bias in favor of connections with shorter roundtrip times.

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 roughly 1 packet every r_i seconds. Thus, node i's throughput is increased by $1/r_i$

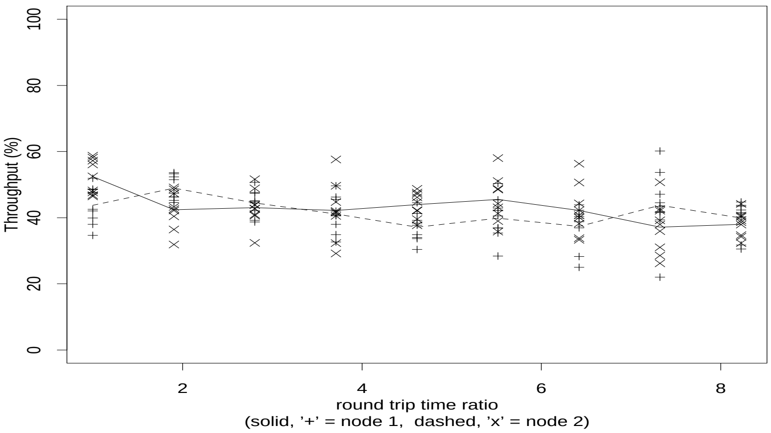

Figure 35: Node 1 and node 2 throughput with Random Early Detection gateways and a modified window increase algorithm.

pkts/sec every r_i seconds, or by $1/(r_i)^2$ pkts/sec every second. Therefore, after a packet from node 2 is dropped and node 2's window is decreased, it takes node 2 significantly longer than node 1 to recover its former throughput rate. This accounts for the reduced throughput for node 2.

Note that if each node i increases 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. Figure 35 shows the results of simulations where each connection increases its window by $2 * (r_i)^2$ packets each roundtrip time. These simulations differ from the simulations in Figure 34 only in the window increase algorithm; in every other respect the two sets of simulations are the same. These simulations use Fast Recovery TCP, Selective Acknowledgement sinks, and Random Early Detection 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 is analyzed in more detail elsewhere (Floyd, 1991).

These simulation results are in accord with previous analytical results (Chiu and Jain, 1989). Chiu and Jain consider linear algorithms for increasing and decreasing the load, where the load could be considered either as a rate or as a window. They show that a purely additive increase in the load gives the quickest convergence to fairness. For the model investigated by Chiu and Jain, this increase occurs at fixed 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. Thus, different fairness goals would imply different window increase algorithms. 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. Fairness goals for connections with multiple congested gateways are discussed further elsewhere (Floyd, 1991).

There are many open questions concerning alternatives to the TCP window modification algorithms. 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

This paper examines traffic phase effects in networks with highly periodic traffic and deterministic gateways. Because of traffic phase effects, 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. The paper discusses 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). 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 show that phase-related biases can be eliminated with the use of appropriate randomization in the gateways. Section 2.5 recommends that when the goal of network simulations is to explore properties of networks with Drop Tail gateways unmasked by the specific details of traffic phase effects, a useful technique is to add a random packet-processing time in the source nodes that ranges from zero to the bottleneck service time.

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. Nevertheless, there are several areas in which networks with Random Drop or Drop Tail gateways both 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 either in this paper or in another paper (Floyd, 1991). 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 (Mankin and Ramakrishnan, 1991).

We show 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. This is not an argument against Random Drop gateways. Our research suggests that this bias against connections with bursty traffic can be corrected with gateways such as Random Early Detection gateways, which provide for congestion avoidance as well as congestion control.

We suggest in Section 3.3 that the bias against connections with longer roundtrip times and large windows results from the TCP window increase algorithm. We have investigated the implications of the bias against traffic with multiple congested gateways (Floyd, 1991). Additional research is necessary to explore possible modifications of the Random Early Detection gateway to identify misbehaving users. We believe that Random Early Detection gateways in general are a promising area for further research.

There are still many open questions. Additional research is needed 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 questions all require additional research.

This paper has focused on understanding the behavior of existing networks and network simulations rather than on designing high-speed networks for the future. Nevertheless, many of the issues discussed in this paper should still be of concern in future high-speed 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, Lixia Zhang, Sugih Jamin, Srinivasan Keshav, Steven McCanne, Chris Torek, and the anonymous referees for helpful comments on this work. This work would have been impossible without Steven McCanne, who made the extensive modifications to our simulator.

A Proofs for analysis of phase effects

In this appendix we give proofs for the analysis of phase effects in Section 2.2. These proofs use the model of packet arrivals at the bottleneck gateway described in that section.

Assumptions for proofs:

• We assume that the transmission time s from the sources is less than the bottleneck transmission time b. This is always true when two fast lines feed into a slow line.

• For simplicity, we assume that $t_1 + s < b$, where t_1 is the phase for node 1 packets. Thus, when node 1 increases its window, both node 1 packets arrive at the gateway during the same service interval.

• For the proofs, we assume that after the gateway queue first reaches size max but before the gateway queue first overflows, at least one node 1 and one node 2 packet arrive at the gateway queue without being dropped. This assumption will not necessarily be true if either connection is in the slow-start phase of the window increment algorithm when the gateway queue first reaches size max. □

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 both node 1 and node 2 packet arrivals. These are shown in Figure 9. □

Claim 1 and Corollary 2 prove that for r_2 slightly less than r_1, after either node increases its window the gateway drops a packet from node 2. Claim 1 applies to the simulations in Figure 4 for 0.5 $< d_{2,3} <$ 3.78 (for an r_2/r_1 roundtrip time ratio between 0.959 and 0.989). Corollary 2 applies to the same simulations for 3.78 $< d_{2,3} <$ 4.28 (for an r_2/r_1 roundtrip time ratio between 0.989 and 0.993). Figure 36 shows the same data in Figure 4, with a line showing the roundtrip time ratios covered by each claim below.

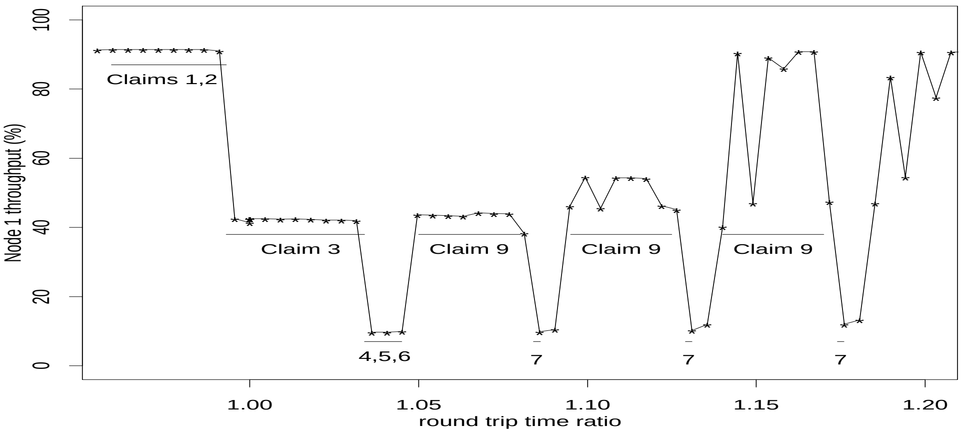

Figure 36: Node 1 throughput as a function of node 2's roundtrip time.

Claim 1 *Let $r_1 - b + s < r_2 < r_1 - t_1 - s$. Then after either node 1 or node 2 increases its window, the gateway drops a packet from node 2.*

Proof: The phase of packet arrivals is illustrated in Figure 9. In this case, $t_1 + s < t_2 < b - s$. Consider the packet arrivals at the gateway before the gateway queue overflows. Each service interval is either a blank interval, a node 1 interval, a node 2 interval, or a double interval. In a double interval, a node 1 packet arrives at time t_1, followed at time $t_2 > t_1 + s$ by a node 2 packet. For each service interval during which the gateway transmits a node 1 packet, another node 1 packet arrives at the gateway $\lfloor r_1/b \rfloor$ service intervals later. For each service interval during which the gateway transmits a node 2 packet, another node 2 packet arrives at the gateway

$$\lfloor r_2/b \rfloor = \lfloor r_1/b \rfloor - 1$$

service intervals later.

Each double or node 2 interval is followed by a node 2 or blank interval. To prove this, assume for contradiction that some interval containing a node 2 packet arrival is followed by some interval containing a node 1 packet arrival. Then one roundtrip earlier, the gateway transmitted a node 1 packet and a node 2 packet at the same time. This is not possible.

Each blank or node 1 interval is followed by a node 1 or a double interval. To prove this, assume for contradiction that some interval with no node 2 packet arrival is followed by some interval with no node 1 packet arrival. Then one roundtrip earlier, there was a service interval in which no node 1 or node 2 packets were transmitted. This violates the assumptions of the model.

Thus, following each blank interval, there is a (possibly empty) sequence of node 1 intervals, followed by a double interval, followed by a (possibly empty) sequence of node 2 intervals, again followed by a blank interval. For a node 1 or node 2 interval the overall queue size does not change. However, during a blank interval the overall queue size decreases, and during a double interval the overall queue size increases. At the end of each double or node 2 interval, the queue is of size max. Therefore, after each node 2 packet arrival the queue is of size max, and after each node 1 packet arrival the queue is of size $max - 1$.

If node 1 increases its window the additional packet increases the queue to size max, because $t_1 + s < t_2$. When a node 2 packet arrives during the double service interval the queue is already full, and the gateway drops the node 2 packet.

If node 2 increases its window the queue full when the additional node 2 packet arrives, because $t_2 < b - s$. The gateway drops the additional node 2 packet. □

Corollary 2 *Let $r_1 - t_1 - s < r_2 < r_1 - t_1$. Then after either node increases its window, the gateway drops a packet from node 2.*

Proof: The pattern of packet arrivals at the gateway is described in Claim 1. Because $b - s < t_2$, if node 2 increases its window the first node 2 packet arrives at the gateway in one service interval, and the additional node 2 packet arrives at the gateway at the start of the following service interval, when the queue is at size $max - 1$. This additional node 2 packet increases the queue to size max. The next packet from node 2, possibly arriving many service intervals later, is dropped at the gateway. □

Claim 3 proves that when r_1 and r_2 are roughly the same, then when some node increases its window a packet from that node is dropped. Claim 3 applies to the simulations in Figure 4 for 4.28 $< d_{2,3} <$ 8.78 (for an r_2/r_1 roundtrip time ratio between 0.993 and 1.034).

Claim 3 *Let $r_1 - t_1 < r_2 < r_1 + (b - t_1 - s)$. Then after either node increases its window, the gateway drops a packet from that node.*

Proof: The phase of packet arrivals is shown in Figure 8. In this case, $t_1, t_2 < b - s$. Each service interval is either a node 1 interval or a node 2 interval. Following the arrival of each packet the queue is at size max, and at the end of the service interval the queue returns to size $max - 1$. When some node increases its window, the additional packet arrives at time s after the original packet, and is dropped. □

Corollary 4 applies to the simulations in Figure 4 for 8.78 $< d_{2,3} <$ 9.28 (for an r_2/r_1 roundtrip time ratio between 1.034 and 1.039). For these parameters Corollary 4 shows that when node 1 increases its window the gateway drops a node 1 packet. When node 2 increases its window the gateway drops the packet following the pair of node 2 packets. If node 2 packets arrive at the gateway in a cluster, then in this case the gateway is likely to drop a node 2 packet.

Corollary 4 *Let $r_1 + (b - t_1 - s) < r_2 < r_1 + (b - t_1)$. When node 1 increases its window, the gateway drops a node 1 packet. When node 2 increases its window,* *the gateway drops the packet that arrives following the additional node 2 packet. This packet could be either from node 1 or from node 2.*

Proof: In this case, $t_1 < b - s < t_2$. When node 2 increases its window, the two packets arrive at the gateway in different service intervals. □

Claim 5 shows that when r_2 is slightly greater than r_1, when either node increases its window the gateway drops a node 1 packet. Claim 5 applies to the simulations in Figure 4 for 9.28 $< d_{2,3} <$ 9.5 (for an r_2/r_1 roundtrip time ratio between 1.039 and 1.041).

Claim 5 *Let $r_1 + (b - t_1) < r_2 < r_1 + b - s$. Then after either node increases its window, the gateway drops a node 1 packet.*

Proof: The proof for Claim 5 is similar to that of Claim 1 and Corollary 2, and is omitted. □

Corollary 6 applies to the simulations in Figure 4 for 9.5 $< d_{2,3} <$ 10.0, (for an r_2/r_1 roundtrip time ratio between 1.041 and 1.045). Corollary 6 states that if node 1 increases its window the gateway drops a node 1 packet. When node 2 increases its window, the gateway drops a node 2 packet only if node 2's window is increased by a node 2 packet that arrives at the gateway during a double interval.

Corollary 6 *Let $r_1 + b - s < r_2 < r_1 + b$. If node 2's window is increased by a node 2 packet that arrives at the gateway during a double interval, then the gateway drops the additional node 2 packet arriving at the gateway. Otherwise, after either node's window is increased, the gateway drops a node 1 packet.*

Proof: In this case, $t_1 - s < t_2 < t_1$. The behavior differs from Claim 5 above only if node 2's window is increased by a node 2 packet that arrives at the gateway during a double interval. In this case, the additional node 2 packet arrives at the gateway after a node 1 packet, when the queue is already at size max, and the additional node 2 packet is dropped. □

Claim 7 applies to the simulations in Figure 4 for 9.28 $+ 5k < d_{2,3} <$ 9.5 $+ 5k$, for any nonnegative integer k. (This corresponds to an r_2/r_1 roundtrip time ratio from 1.039 + 0.045*k to 1.041 + 0.045*k.) For the simulations in Figure 4 for this range, node 1 generally receives all of the packet drops. Claim 7 states that if node 1 increases its window, the gateway drops a node 1 packet. If node 2 increases its window, the gateway could drop either a node 1 or a node 2 packet. Corollary 8 gives a condition that, if satisfied, ensures that the gateway only drops node 1 packets.

Claim 7 *Let $r_2 > r_1 + b$, and $t_2 < t_1 - s$. If node 1 increases its window, the gateway drops a node 1 packet. If node 2's window is increased by a node 2 packet* *that arrives at the gateway during a node 2 interval with a maximum-size queue, then the gateway drops a node 2 packet. Otherwise, the gateway drops a node 1 packet.*

Proof: For some $i \geq 1$,

$$r_2 = r_1 + i * b + 1 - (t_1 - t_2).$$

By definition $r_1 = k * b + t_1$ for some k, and each node 1 packet arrives at the gateway k service intervals after some previous node 1 packet was transmitted by the gateway. Each node 2 packet arrives at the gateway $k + i + 1$ service intervals after some previous node 2 packet was transmitted by the gateway. Each service interval containing a node 1 packet arrival is followed $i+1$ intervals later by a service interval that does not contain a node 2 packet arrival. Similarly, each service interval that doesn't contain a node 1 packet arrival is followed $i + 1$ intervals later by an interval that contains a node 2 packet arrival.

Start in one of the first $i + 1$ service intervals after the queue first reaches size max, and consider the subsequence that includes that service interval and each succeeding *(i+1)*th service interval. We call this a *mod(i+1)-subsequence*. Within each mod*(i+1)*-subsequence, following each blank interval there is a (possibly empty) sequence of node 2 intervals, followed by a double interval, followed by a (possibly empty) sequence of node 1 intervals. Each double interval contains a node 2 packet arrival followed by a node 1 packet arrival.

For each service interval and for each mod*(i+1)*-subsequence, we say that that service interval is in either the *low part* or the *high part* of that subsequence. We say that a service interval is in the *low part* of a subsequence if it occurs on or after a blank interval in that subsequence and before the following double interval in that subsequence. Otherwise, the service interval is in the *high part* of that subsequence. If some subsequence contains no blank or double intervals, then we will say that all service intervals lie in the high part of that subsequence. Note that a service interval can lie in the high part of some subsequences, and in the low part of others.

Each service interval lies in the high part of k subsequences for some k, for $0 \leq k \leq i + 1$. If one service interval lies in the high part of k subsequences, and the following service interval lies in the high part of $k + 1$ subsequences, then the second service interval is a double interval, and the queue at the end of the second service interval is larger than the queue at the end of the first service interval. On the other hand, if one service interval lies in the high part of k subsequences, and the following service interval lies in the high part of $k - 1$ subsequences, then the second service interval is a blank interval, and the queue at the end of the second service interval is smaller than the queue at the end of the first service interval. If two consecutive service intervals both lie in the high part of k subsequences for some k, then they have the same queue size.

Assume that some service interval lies in the high part of j mod*(i+1)*-subsequences, for $j \leq i + 1$, and that no service interval lies in the high part of $j + 1$ mod*(i+1)*-subsequences. Then the maximum queue size is reached only during those service intervals that lie in the high part of j mod*(i+1)*-subsequences. We call these the *maximum* service intervals. For each consecutive run of maximum service intervals, the first interval is a double interval, and each succeeding interval is either a node 2 or a node 1 interval.

If node 1's window is increased by a node 1 packet that arrives at the gateway during a maximum service interval, then the queue is at size max when the additional node 1 packet arrives at the gateway, and the gateway drops the additional node 1 packet.

If node 1 or node 2's window is increased by a packet that arrives at the gateway during a non-maximum service interval, then the queue size increases by one, and the gateway drops the node 1 packet that arrives in the first succeeding maximum service interval.

If node 2's window is increased by a packet that arrives at the gateway during a maximum service interval that is also a double interval, then the gateway drops the node 1 packet arriving in that service interval. If node 2's is increased by a packet that arrives during a maximum service interval that is a node 2 interval, then the gateway drops the additional node 2 packet arriving in that service interval. □

Corollary 8 *Let $r_2 > r_1 + b$, and $t_2 < t_1 - s$, as in Claim 7. If there exists some service interval that lies in the high part of all $i + 1$ $mod(i + 1)$-subsequences (as defined in the proof of Claim 7), then when either node increases its window, the gateway drops a node 1 packet.*

Proof: A node 2 interval lies in the low part of the subsequence containing that service interval. Therefore no node 2 interval lies in the high part of all $i + 1$ mod*(i+1)*-subsequences. From Claim 7, when either node increases its window, the gateway drops a node 1 packet. □

Claim 9 applies to the simulations in Figure 4 for $10.5 + 5k < d_{2,3} < 13.78 + 5k$, for any nonnegative integer k. (This corresponds to an r_2/r_1 roundtrip time ratio from 1.050 + 0.045*k to 1.08 + 0.045*k.) For the simulations in Figure 4 for this range, the percentage of packet drops for node 1 ranges from 0% to 50%. Claim 9 states that when node 2 increases its window, the gateway drops a node 2 packet. When node 1 increases its window, the gateway drops either a node 1 or a node 2 packet.

Claim 9 *Let $r_2 > r_1 + b$, and $t_1 + s < t_2 < b - s$. When node 2 increases its window the gateway drops a node 2 packet. When node 1 increases its window the gateway could drop either a node 1 or a node 2 packet.*

Proof: We omit this proof, which is similar to that of Claim 7 above. □

We use the results above to explain the simulations presented in Section 2.1. If both roundtrip times are equal and neither node 1 nor node 2 has reached its maximum window, then both nodes increase their windows in each drop period. From Claim 3, when some node increases its window, the gateway drops a packet from that node. In this case the gateway drops a node 1 and a node 2 packet 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 $r_1 - b + s \leq r_2 \leq r_1 - t_1$, as in Case 2, Claim 1, and Corollary 2, the gateway only drops node 2 packets. Even if node 1 and node 2 both 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 from 212.44 ms. to 220 ms., corresponding to roundtrip ratios from 0.960 to 0.993.)

When $r_1 + (b - t_1) < r_2 < r_1 + b - s$, as in Case 3 and Claim 5, the gateway only drops node 1 packets. This is shown in Figure 4 when r_2 ranges roughly from 230 ms. to 230.44 ms. (corresponding to roundtrip ratios from 1.039 to 1.041). 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 as shown 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 the phase relationship between node 1 and node 2 packet arrivals. This is explained in Claim 7 and Claim 9.

References

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.

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.

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

Floyd, S., and Jacobson, V., *Traffic Phase Effects in Packet-Switched Gateways*, Computer Communications Review, V.21 N.2, April 1991, pp. 26-42.

Floyd, S., *Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic*, Computer Communication Review, V.21 N.5, October 1991, pp. 30-47.

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

Jacobson, V., *Congestion Avoidance and Control*, Proceedings of SIGCOMM '88, August 1988, pp. 314-329.

Jacobson, V., "Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno", Proceedings of the British Columbia Internet Engineering Task Force, July 1990.

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

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

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

Mankin, A., *Random Drop Congestion Control*, Proceedings of SIGCOMM 90, Sept. 1990, pp.1-7.

Mankin, A. and Ramakrishnan, K. K., editors for the IETF Performance and Congestion Control Working Group, "Gateway congestion control survey", *RFC 1254*, August 1991.

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.

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

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, pp. 30-39.

Wilder, R., Ramakrishnan, K.K., and Mankin, A., "Dynamics of Congestion Control and Avoidance in Two-Way Traffic in an OSI Testbed", Computer Communication Review, V.21 N.2, April 1991. pp.43-48.

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