Random Early Detection Gateways for Congestion Avoidance

Sally Floyd and Van Jacobson*

Lawrence Berkeley Laboratory
University of California
floyd@ee.lbl.gov
van@ee.lbl.gov

To appear in the August 1993 IEEE/ACM Transactions on Networking

Abstract

This paper presents Random Early Detection (RED) gateways for congestion avoidance in packet-switched networks. The gateway detects incipient congestion by computing the average queue size. The gateway could notify connections of congestion either by dropping packets arriving at the gateway or by setting a bit in packet headers. When the average queue size exceeds a preset threshold, the gateway drops or *marks* each arriving packet with a certain probability, where the exact probability is a function of the average queue size.

RED gateways keep the average queue size low while allowing occasional bursts of packets in the queue. During congestion, the probability that the gateway notifies a particular connection to reduce its window is roughly proportional to that connection's share of the bandwidth through the gateway. RED gateways are designed to accompany a transport-layer congestion control protocol such as TCP. The RED gateway has no bias against bursty traffic and avoids the global synchronization of many connections decreasing their window at the same time. Simulations of a TCP/IP network are used to illustrate the performance of RED gateways.

1 Introduction

In high-speed networks with connections with large delay-bandwidth products, gateways are likely to be designed with correspondingly large maximum queues to accommodate transient congestion. In the current Internet, the TCP transport protocol detects congestion only after a packet has been dropped at the gateway. However, it would clearly be undesirable to have large queues (possibly on the order of a delay-bandwidth product) that were full much of the time; this would significantly increase the average delay in the network. Therefore, with increasingly high-speed networks, it is increasingly important to have mechanisms that keep throughput high but average queue sizes low.

In the absence of explicit feedback from the gateway, there are a number of mechanisms that have been proposed for transport-layer protocols to maintain high throughput and low delay in the network. Some of these proposed mechanisms are designed to work with current gateways [15, 23, 31, 33, 34], while other mechanisms are coupled with gateway scheduling algorithms that require per-connection state in the gateway [20, 22]. In the absence of explicit feedback from the gateway, transport-layer protocols could infer congestion from the estimated bottleneck service time, from changes in throughput, from changes in

*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.

end-to-end delay, as well as from packet drops or other methods. Nevertheless, the view of an individual connection is limited by the timescales of the connection, the traffic pattern of the connection, the lack of knowledge of the number of congested gateways, the possibilities of routing changes, as well as by other difficulties in distinguishing propagation delay from persistent queueing delay.

The most effective detection of congestion can occur in the gateway itself. The gateway can reliably distinguish between propagation delay and persistent queueing delay. Only the gateway has a unified view of the queueing behavior over time; the perspective of individual connections is limited by the packet arrival patterns for those connections. In addition, a gateway is shared by many active connections with a wide range of roundtrip times, tolerances of delay, throughput requirements, etc.; decisions about the duration and magnitude of transient congestion to be allowed at the gateway are best made by the gateway itself.

The method of monitoring the average queue size at the gateway, and of notifying connections of incipient congestion, is based of the assumption that it will continue to be useful to have queues at the gateway where traffic from a number of connections is multiplexed together, with FIFO scheduling. Not only is FIFO scheduling useful for sharing delay among connections, reducing delay for a particular connection during its periods of burstiness [4], but it scales well and is easy to implement efficiently. In an alternate approach, some congestion control mechanisms that use variants of Fair Queueing [20] or hop-by-hop flow control schemes [22] propose that the gateway scheduling algorithm make use of per-connection state for every active connection. We would suggest instead that per-connection gateway mechanisms should be used *only* in those circumstances where gateway scheduling mechanisms without per-connection mechanisms are clearly inadequate.

The DECbit congestion avoidance scheme [18], described later in this paper, is an early example of congestion detection at the gateway; DECbit gateways give explicit feedback when the average queue size exceeds a certain threshold. This paper proposes a different congestion avoidance mechanism at the gateway, RED (Random Early Detection) gateways, with somewhat different methods for detecting congestion and for choosing which connections to notify of this congestion.

While the principles behind RED gateways are fairly general, and RED gateways can be useful in controlling the average queue size even in a network where the transport protocol can not be trusted to be cooperative, RED gateways are intended for a network where the transport protocol responds to congestion indications from the network. The gateway congestion control mechanism in RED gateways simplifies the congestion control job required of the transport protocol, and should be applicable to transport-layer congestion control mechanisms other than the current version of TCP, including protocols with rate-based rather than window-based flow control.

However, some aspects of RED gateways are specifically targeted to TCP/IP networks. The RED gateway is designed for a network where a single marked or dropped packet is sufficient to signal the presence of congestion to the transport-layer protocol. This is different from the DECbit congestion control scheme, where the transport-layer protocol computes the *fraction* of arriving packets that have the congestion indication bit set.

In addition, the emphasis on avoiding the global synchronization that results from many connections reducing their windows at the same time is particularly relevant in a network with 4.3-Tahoe BSD TCP [14], where each connection goes through Slow-Start, reducing the window to one, in response to a dropped packet. In the DECbit congestion control scheme, for example, where each connection's response to congestion is less severe, it is also less critical to avoid this global synchronization.

RED gateways can be useful in gateways with a range of packet-scheduling and packet-dropping algorithms. For example, RED congestion control mechanisms could be implemented in gateways with drop preference, where packets are marked as either "essential" or "optional", and "optional" packets are dropped first when the queue exceeds a certain size. Similarly, for a gateway with separate queues for realtime and non-realtime traffic, for example, RED congestion control mechanisms could be applied to the queue for one of these traffic classes.

The RED congestion control mechanisms monitor the average queue size for each output queue, and, using randomization, choose connections to notify of that congestion. Transient congestion is accommodated by a temporary increase in the queue. Longer-lived congestion is reflected by an increase in the computed average queue size, and results in randomized feedback to some of the connections to decrease their windows. The probability that a connection is notified of congestion is proportional to that connection's share of the throughput through the gateway.

Gateways that detect congestion before the queue overflows are not limited to packet drops as the method for notifying connections of congestion. RED gateways can *mark* a packet by dropping it at the gateway or by setting a bit in the packet header, depending on the transport protocol. When the average queue size exceeds a maximum threshold, the RED gateway marks every packet that arrives at the gateway. If RED gateways mark packets by *dropping* them, rather than by setting a bit in the packet header, when the average queue size exceeds the maximum threshold, then the RED gateway controls the average queue size even in the absence of a cooperating transport protocol.

One advantage of a gateway congestion control mechanism that works with current transport protocols, and that does not require that all gateways in the internet use the same gateway congestion control mechanism, is that it could be deployed gradually in the current Internet. RED gateways are a simple mechanism for congestion avoidance that could be implemented gradually in current TCP/IP networks with no changes to transport protocols.

Section 2 discusses previous research on Early Random Drop gateways and other congestion avoidance gateways. Section 3 outlines design guidelines for RED gateways. Section 4 presents the RED gateway algorithm, and Section 5 describes simple simulations. Section 6 discusses in detail the parameters used in calculating the average queue size, and Section 7 discusses the algorithm used in calculating the packet-marking probability.

Section 8 examines the performance of RED gateways, including the robustness of RED gateways for a range of traffic and for a range of parameter values. Simulations in Section 9 demonstrate, among other things, the RED gateway's lack of bias against bursty traffic. Section 10 describes how RED gateways can be used to identify those users that are using a large fraction of the bandwidth through a congested gateway. Section 11 discusses methods for efficiently implementing RED gateways. Section 12 gives conclusions and describes areas for future work.

2 Previous work on congestion avoidance gateways

2.1 Early Random Drop gateways

Several researchers have studied Early Random Drop gateways as a method for providing congestion avoidance at the gateway.[1]

Hashem [11] discusses some of the shortcomings of Random Drop[2] and Drop Tail gateways, and briefly investigates Early Random Drop gateways. In the implementation of Early Random Drop gateways in [11], if the queue length exceeds a certain *drop level*, then the gateway drops each packet arriving at the gateway with a fixed *drop probability*. This is discussed as a rough initial implementation. Hashem [11] stresses that

[1]Jacobson [14] proposed gateways to monitor the average queue size to detect incipient congestion, and to randomly drop packets when congestion is detected. These proposed gateways are a precursor to the Early Random Drop gateways that have been studied by several authors [11] [36]. We refer to the gateways in this paper as Random Early Detection or RED gateways. RED gateways differ from the earlier Early Random Drop gateways in several respects: the *average* queue size is measured; the gateway is not limited to *dropping* packets; and the packet-marking probability is a function of the average queue size.

[2]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.

in future implementations the drop level and the drop probability should be adjusted dynamically, depending on network traffic.

Hashem [11] points out that with Drop Tail gateways each congestion period introduces global synchronization in the network. When the queue overflows, packets are often dropped from several connections, and these connections decrease their windows at the same time. This results in a loss of throughput at the gateway. The paper shows that Early Random Drop gateways have a broader view of traffic distribution than do Drop Tail or Random Drop gateways and reduce global synchronization. The paper suggests that because of this broader view of traffic distribution, Early Random Drop gateways have a better chance than Drop Tail gateways of targeting aggressive users. The conclusions in [11] are that Early Random Drop gateways deserve further investigation.

For the version of Early Random Drop gateways used in the simulations in [36], if the queue is more than half full then the gateway drops each arriving packet with probability 0.02. Zhang [36] shows that this version of Early Random Drop gateways was not successful in controlling misbehaving users. In these simulations, with both Random Drop and Early Random Drop gateways, the misbehaving users received roughly 75% higher throughput than the users implementing standard 4.3 BSD TCP.

The Gateway Congestion Control Survey [21] considers the versions of Early Random Drop described above. The survey cites the results in which the Early Random Drop gateway is unsuccessful in controlling misbehaving users [36]. As mentioned in [32], Early Random Drop gateways are not expected to solve all of the problems of unequal throughput given connections with different roundtrip times and multiple congested gateways. In [21], the goals of Early Random Drop gateways for congestion avoidance are described as "uniform, dynamic treatment of users (streams/flows), of low overhead, and of good scaling characteristics in large and loaded networks". It is left as an open question whether or not these goals can be achieved.

2.2 Other approaches to gateway mechanisms for congestion avoidance

Early descriptions of IP Source Quench messages suggest that gateways could send Source Quench messages to source hosts before the buffer space at the gateway reaches capacity [26], and before packets have to be dropped at the gateway. One proposal [27] suggests that the gateway send Source Quench messages when the queue size exceeds a certain threshold, and outlines a possible method for flow control at the source hosts in response to these messages. The proposal also suggests that when the gateway queue size approaches the maximum level the gateway could discard arriving packets other than ICMP packets.

The DECbit congestion avoidance scheme, a binary feedback scheme for congestion avoidance, is described in [29]. In the DECbit 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 exceeds 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.

There are several significant differences between DECbit gateways and the RED gateways described in this paper. The first difference concerns the method of computing the average queue size. Because the DECbit scheme chooses the last (busy + idle) cycle plus the current busy period for averaging the queue size, the queue size can sometimes be averaged over a fairly short period of time. In high-speed networks with large buffers at the gateway, it would be desirable to explicitly control the time constant for the computed average queue size; this is done in RED gateways using time-based exponential decay. In [29] the authors report that they rejected the idea of a weighted exponential running average of the queue length because when the time interval was far from the roundtrip time, there was bias in the network. This problem of bias does not arise with RED gateways because RED gateways use a randomized algorithm for marking packets, and assume that the sources use a different algorithm for responding to marked packets. In a DECbit network, the source looks at the fraction of packets that have been marked in the last roundtrip time. For a network with RED gateways, the source should reduce its window even if there is only one marked packet.

A second difference between DECbit gateways and RED gateways concerns the method for choosing connections to notify of congestion. In the DECbit scheme there is no conceptual separation between the algorithm to detect congestion and the algorithm to set the congestion indication bit. When a packet arrives at the gateway and the computed average queue size is too high, the congestion indication bit is set in the header of that packet. Because of this method for marking packets, DECbit networks can exhibit a bias against bursty traffic [see Section 9]; this is avoided in RED gateways by using randomization in the method for marking packets. For congestion avoidance gateways designed to work with TCP, an additional motivation for using randomization in the method for marking packets is to avoid the global synchronization that results from many TCP connections reducing their window at the same time. This is less of a concern in networks with the DECbit congestion avoidance scheme, where each source decreases its window fairly moderately in response to congestion.

Another proposal for adaptive window schemes where the source nodes increase or decrease their windows according to feedback concerning the queue lengths at the gateways is presented in [25]. Each gateway has an upper threshold UT indicating congestion, and a lower threshold LT indicating light load conditions. Information about the queue sizes at the gateways is added to each packet. A source node increases its window only if all the gateway queue lengths in the path are below the lower thresholds. If the queue length is above the upper threshold for any queue along the path, then the source node decreases its window. One disadvantage of this proposal is that the network responds to the instantaneous queue lengths, not to the average queue lengths. We believe that this scheme would be vulnerable to traffic phase effects and to biases against bursty traffic, and would not accommodate transient increases in the queue size.

3 Design guidelines

This section summarizes some of the design goals and guidelines for RED gateways. The main goal is to provide congestion avoidance by controlling the average queue size. Additional goals include the avoidance of global synchronization and of a bias against bursty traffic and the ability to maintain an upper bound on the average queue size even in the absence of cooperation from transport-layer protocols.

The first job of a congestion avoidance mechanism at the gateway is to detect incipient congestion. As defined in [18], a *congestion avoidance* scheme maintains the network in a region of low delay and high throughput. The average queue size should be kept low, while fluctuations in the actual queue size should be allowed to accommodate bursty traffic and transient congestion. Because the gateway can monitor the size of the queue over time, the gateway is the appropriate agent to detect incipient congestion. Because the gateway has a unified view of the various sources contributing to this congestion, the gateway is also the appropriate agent to decide which sources to notify of this congestion.

In a network with connections with a range of roundtrip times, throughput requirements, and delay sensitivities, the gateway is the most appropriate agent to determine the size and duration of short-lived bursts in queue size to be accommodated by the gateway. The gateway can do this by controlling the time constants used by the low-pass filter for computing the average queue size. The goal of the gateway is to detect incipient congestion that has persisted for a "long time" (several roundtrip times).

The second job of a congestion avoidance gateway is to decide which connections to notify of congestion at the gateway. If congestion is detected before the gateway buffer is full, it is not necessary for the gateway to drop packets to notify sources of congestion. In this paper, we say that the gateway *marks* a packet, and *notifies* the source to reduce the window for that connection. This marking and notification can consist of dropping a packet, setting a bit in a packet header, or some other method understood by the transport protocol. The current feedback mechanism in TCP/IP networks is for the gateway to drop packets, and the simulations of RED gateways in this paper use this approach.

One goal is to avoid a bias against bursty traffic. Networks contain connections with a range of burstiness, and gateways such as Drop Tail and Random Drop gateways have a bias against bursty traffic. With Drop Tail gateways, the more bursty the traffic from a particular connection, the more likely it is that the gateway queue will overflow when packets from that connection arrive at the gateway [7].

Another goal in deciding which connections to notify of congestion is to avoid the global synchronization that results from notifying all connections to reduce their windows at the same time. Global synchronization has been studied in networks with Drop Tail gateways [37], and results in loss of throughput in the network. Synchronization as a general network phenomena has been explored in [8].

In order to avoid problems such as biases against bursty traffic and global synchronization, congestion avoidance gateways can use distinct algorithms for congestion detection and for deciding which connections to notify of this congestion. The RED gateway uses randomization in choosing which arriving packets to mark; with this method, the probability of marking a packet from a particular connection is roughly proportional to that connection's share of the bandwidth through the gateway. This method can be efficiently implemented without maintaining per-connection state at the gateway.

One goal for a congestion avoidance gateway is the ability to control the average queue size even in the absence of cooperating sources. This can be done if the gateway *drops* arriving packets when the average queue size exceeds some maximum threshold (rather than setting a bit in the packet header). This method could be used to control the average queue size even if most connections last less than a roundtrip time (as could occur with modified transport protocols in increasingly high-speed networks), and even if connections fail to reduce their throughput in response to marked or dropped packets.

4 The RED algorithm

This section describes the algorithm for RED gateways. The RED gateway calculates the average queue size, using a low-pass filter with an exponential weighted moving average. The average queue size is compared to two thresholds, a *minimum* threshold and a *maximum* threshold. When the average queue size is less than the minimum threshold, no packets are marked. When the average queue size is greater than the maximum threshold, every arriving packet is marked. If marked packets are in fact dropped, or if all source nodes are cooperative, this ensures that the average queue size does not significantly exceed the maximum threshold.

When the average queue size is between the minimum and the maximum threshold, each arriving packet is marked with probability p_a, where p_a is a function of the average queue size avg. Each time that a packet is marked, the probability that a packet is marked from a particular connection is roughly proportional to that connection's share of the bandwidth at the gateway. The general RED gateway algorithm is given in Figure 1.

Thus the RED gateway has two separate algorithms. The algorithm for computing the average queue size determines the degree of burstiness that will be allowed in the gateway queue. The algorithm for calculating the packet-marking probability determines how frequently the gateway marks packets, given the current level of congestion. The goal is for the gateway to mark packets at fairly evenly-spaced intervals, in order to avoid biases and to avoid global synchronization, and to mark packets sufficiently frequently to control the average queue size.

The detailed algorithm for the RED gateway is given in Figure 2. Section 11 discusses efficient implementations of these algorithms.

The gateway's calculations of the average queue size take into account the period when the queue is

```
for each packet arrival
   calculate the average queue size avg
   if min_th ≤ avg < max_th
      calculate probability p_a
      with probability p_a:
         mark the arriving packet
   else if max_th ≤ avg
      mark the arriving packet
```

Figure 1: General algorithm for RED gateways.

empty (the idle period) by estimating the number m of small packets that *could* have been transmitted by the gateway during the idle period. After the idle period the gateway computes the average queue size as if m packets had arrived to an empty queue during that period.

As avg varies from min_{th} to max_{th}, the packet-marking probability p_b varies linearly from 0 to max_p:

$$p_b \leftarrow max_p(avg - min_{th})/(max_{th} - min_{th}).$$

The final packet-marking probability p_a increases slowly as the count increases since the last marked packet:

$$p_a \leftarrow p_b/(1 - count \cdot p_b)$$

As discussed in Section 7, this ensures that the gateway does not wait too long before marking a packet.

The gateway marks each packet that arrives at the gateway when the average queue size avg exceeds max_{th}.

One option for the RED gateway is to measure the queue in bytes rather than in packets. With this option, the average queue size accurately reflects the average delay at the gateway. When this option is used, the algorithm would be modified to ensure that the probability that a packet is marked is proportional to the packet size in bytes:

$$\begin{aligned} p_b &\leftarrow max_p(avg - min_{th})/(max_{th} - min_{th}) \\ p_b &\leftarrow p_b\,\mathrm{PacketSize}/\mathrm{MaximumPacketSize} \\ p_a &\leftarrow p_b/(1 - count \cdot p_b) \end{aligned}$$

In this case a large FTP packet is more likely to be marked than is a small TELNET packet.

Sections 6 and 7 discuss in detail the setting of the various parameters for RED gateways. Section 6 discusses the calculation of the average queue size. The queue weight w_q is determined by the size and duration of bursts in queue size that are allowed at the gateway. The minimum and maximum thresholds min_{th} and max_{th} are determined by the desired average queue size. The average queue size which makes the desired tradeoffs (such as the tradeoff between maximizing throughput and minimizing delay) depends on network characteristics, and is left as a question for further research. Section 7 discusses the calculation of the packet-marking probability.

In this paper our primary interest is in the functional operation of the RED gateways. Specific questions about the most efficient implementation of the RED algorithm are discussed in Section 11.

5 A simple simulation

This section describes our simulator and presents a simple simulation with RED gateways. Our simulator is a version of the REAL simulator [19] built on Columbia's Nest simulation package [1], with extensive

```
Initialization:
   avg ← 0
   count ← −1
for each packet arrival
   calculate the new average queue size avg:
      if the queue is nonempty
         avg ← (1 − w_q)avg + w_q q
      else
         m ← f(time − q_time)
         avg ← (1 − w_q)^m avg
   if min_th ≤ avg < max_th
      increment count
      calculate probability p_a:
         p_b ← max_p(avg − min_th)/(max_th − min_th)
         p_a ← p_b/(1 − count · p_b)
      with probability p_a:
         mark the arriving packet
         count ← 0
   else if max_th ≤ avg
      mark the arriving packet
      count ← 0
   else count ← −1
when queue becomes empty
   q_time ← time
```

Saved Variables:

- avg: average queue size
- q_time: start of the queue idle time
- $count$: packets since last marked packet

Fixed parameters:

- w_q: queue weight
- min_{th}: minimum threshold for queue
- max_{th}: maximum threshold for queue
- max_p: maximum value for p_b

Other:

- p_a: current packet-marking probability
- q: current queue size
- $time$: current time
- $f(t)$: a linear function of the time t

Figure 2: Detailed algorithm for RED gateways.

modifications and bug fixes made by Steven McCanne at LBL. In the simulator, FTP sources always have a packet to send and always send a maximal-sized (1000-byte) packet as soon as the congestion control window allows them to do so. A sink immediately sends an ACK packet when it receives a data packet. The gateways use FIFO queueing.

Source and sink nodes implement a congestion control algorithm equivalent to that in 4.3-Tahoe BSD TCP.[3] Briefly, there are two phases to the window-adjustment algorithm. A threshold is set initially to half the receiver's advertised window. In the slow-start phase, 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 this paper refers to as the "maximum window size". In 4.3-Tahoe BSD TCP, packet loss (a dropped packet) is treated as a "congestion experienced" signal. The source reacts to a packet loss by setting the threshold to half the current window, decreasing the current window to one packet, and entering the slow-start phase.

Figure 3 shows a simple simulation with RED gateways. The network is shown in Figure 4. The simulation contains four FTP connections, each with a maximum window roughly equal to the delay-bandwidth product, which ranges from 33 to 112 packets. The RED gateway parameters are set as follows: $w_q = 0.002$, $min_{th} = 5$ packets, $max_{th} = 15$ packets, and $max_p = 1/50$. The buffer size is sufficiently large that packets are never dropped at the gateway due to buffer overflow; in this simulation the RED gateway controls the average queue size, and the actual queue size never exceeds forty packets.

For the charts in Figure 3, the x-axis shows the time in seconds. The bottom chart shows the packets from nodes 1-4. Each of the four main rows shows the packets from one of the four connections; the bottom row shows node 1 packets, and the top row shows node 4 packets. There is a mark for each data packet as it arrives at the gateway and as it departs from the gateway; at this time scale, the two marks are often indistinguishable. The y-axis is a function of the packet sequence number; for packet number n from node i, the y-axis shows $n \bmod 90 + (i-1)100$. Thus, each vertical 'line' represents 90 consecutively-numbered packets from one connection arriving at the gateway. Each 'X' shows a packet dropped by the gateway, and each 'X' is followed by a mark showing the retransmitted packet. Node 1 starts sending packets at time 0, node 2 starts after 0.2 seconds, node 3 starts after 0.4 seconds, and node 4 starts after 0.6 seconds.

The top chart of Figure 3 shows the instantaneous queue size q and the calculated average queue size avg. The dotted lines show min_{th} and max_{th}, the minimum and maximum thresholds for the average queue size. Note that the calculated average queue size avg changes fairly slowly compared to q. The bottom row of X's on the bottom chart shows again the time of each dropped packet.

This simulation shows the success of the RED gateway in controlling the average queue size at the gateway in response to a dynamically changing load. As the number of connections increases, the frequency with which the gateway drops packets also increases. There is no global synchronization. The higher throughput for the connections with shorter roundtrip times is due to the bias of TCP's window increase algorithm in favor of connections with shorter roundtrip times (as discussed in [6, 7]). For the simulation in Figure 3 the average link utilization is 76%. For the following second of the simulation, when all four sources are active, the average link utilization is 82%. (This is not shown in Figure 3.)

Because RED gateways can control the average queue size while accommodating transient congestion, RED gateways are well-suited to provide high throughput and low *average* delay in high-speed networks with TCP connections that have large windows. The RED gateway can accommodate the short burst in the queue required by TCP's slow-start phase; thus RED gateways control the *average* queue size while still allowing TCP connections to smoothly open their windows. Figure 5 shows the results of simulations of the network in Figure 6 with two TCP connections, each with a maximum window of 240 packets, roughly equal to the delay-bandwidth product. The two connections are started at slightly different times. The simulations compare the performance of Drop Tail and of RED gateways.

In Figure 5 the x-axis shows the total throughput as a fraction of the maximum possible throughput on the congested link. The y-axis shows the average queue size in packets (as seen by arriving packets).

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

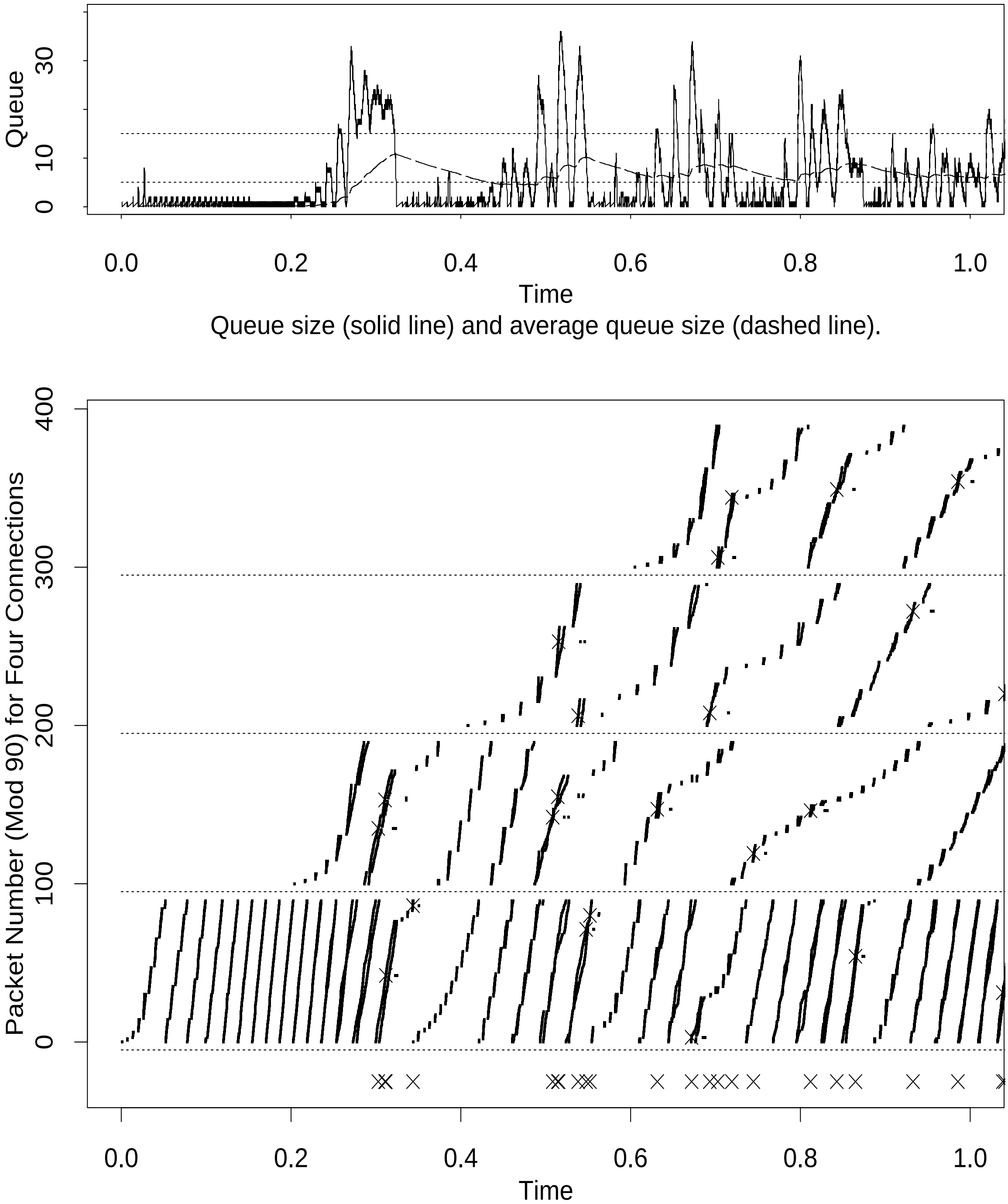

Figure 3: A simulation with four FTP connections with staggered start times.

Five 5-second simulations were run for each of 11 sets of parameters for Drop Tail gateways, and for 11 sets of parameters for RED gateways; each mark in Figure 5 shows the results of one of these five-second simulations. The simulations with Drop Tail gateways were run with the buffer size ranging from 15 to 140 packets; as the buffer size is increased, the throughput and the average queue size increase correspondingly. In order to avoid phase effects in the simulations with Drop Tail gateways, the source node takes a random

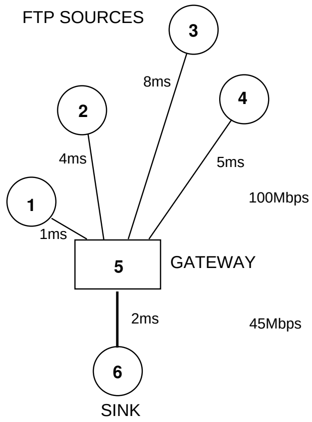

Figure 4: Simulation network.

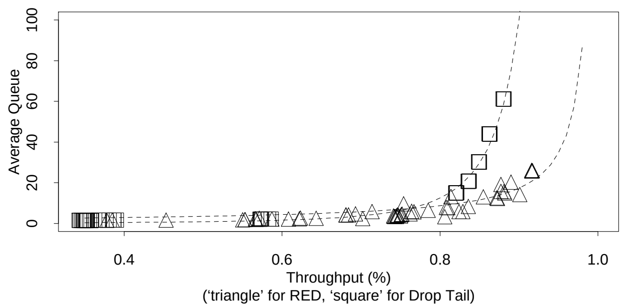

Figure 5: Comparing Drop Tail and RED gateways.

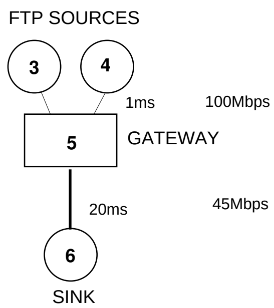

Figure 6: Simulation network.

time drawn from the uniform distribution on [0, t] seconds to prepare an FTP packet for transmission, where t is the bottleneck service time of 0.17 ms. [7].

The simulations with RED gateways were all run with a buffer size of 100 packets, with min_{th} ranging from 3 to 50 packets. For the RED gateways, max_{th} is set to $3\,min_{th}$, with $w_q = 0.002$ and $max_p = 1/50$. The dashed lines show the average delay (as a function of throughput) approximated by $1.73/(1-x)$ for the simulations with RED gateways, and approximated by $0.1/(1-x)^3$ for the simulations with Drop Tail gateways. For this simple network with TCP connections with large windows, the network power (the ratio of throughput to delay) is higher with RED gateways than with Drop Tail gateways. There are several reasons for this difference. With Drop Tail gateways with a small maximum queue, the queue drops packets while the TCP connection is in the slow-start phase of rapidly increasing its window, reducing throughput. On the other hand, with Drop Tail gateways with a large maximum queue the average delay is unacceptably large. In addition, Drop Tail gateways are more likely to drop packets from both connections at the same time, resulting in global synchronization and a further loss of throughput.

Later in the paper, we discuss simulation results from networks with a more diverse range of connections. The RED gateway is not specifically designed for a network dominated by bulk data transfer; this is simply an easy way to simulate increasingly-heavy congestion at a gateway.

6 Calculating the average queue length

The RED gateway uses a low-pass filter to calculate the average queue size. Thus, the short-term increases in the queue size that result from bursty traffic or from transient congestion do not result in a significant increase in the average queue size.

The low-pass filter is an exponential weighted moving average (EWMA):

$$avg \leftarrow (1-w_q)avg + w_q\, q. \tag{1}$$

The weight w_q determines the time constant of the low-pass filter. The following sections discuss upper and lower bounds for setting w_q. The calculation of the average queue size can be implemented particularly efficiently when w_q is a (negative) power of two, as shown in Section 11.

6.1 An upper bound for w_q

If w_q is too large, then the averaging procedure will not filter out transient congestion at the gateway.

Assume that the queue is initially empty, with an average queue size of zero, and then the queue increases from 0 to L packets over L packet arrivals. After the Lth packet arrives at the gateway, the average queue size avg_L is

$$\begin{aligned} avg_L &= \sum_{i=1}^{L} i\, w_q (1-w_q)^{L-i} \\ &= w_q(1-w_q)^L \sum_{i=1}^{L} i \left(\frac{1}{1-w_q}\right)^i \\ &= L+1+\frac{(1-w_q)^{L+1}-1}{w_q}. \end{aligned} \tag{2}$$

This derivation uses the following identity [9, p. 65]:

$$\sum_{i=1}^{L} ix^i = \frac{x+(Lx-L-1)x^{L+1}}{(1-x)^2}.$$

Figure 7 shows the average queue size avg_L for a range of values for w_q and L. The x-axis shows w_q from 0.001 to 0.005, and the y-axis shows L from 10 to 100. For example, for $w_q = 0.001$, after a queue increase from 0 to 100 packets, the average queue size avg_{100} is 4.88 packets.

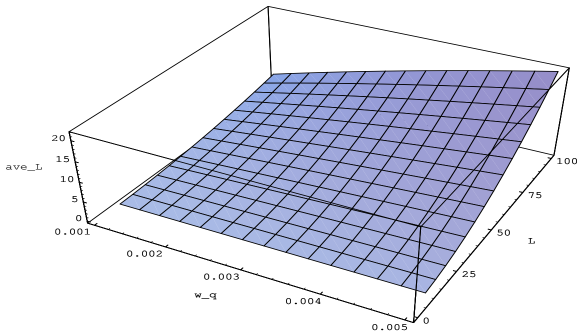

Figure 7: avg_L as a function of w_q and L.

Given a minimum threshold min_{th}, and given that we wish to allow bursts of L packets arriving at the gateway, then w_q should be chosen to satisfy the following equation for $avg_L < min_{th}$:

$$L+1+\frac{(1-w_q)^{L+1}-1}{w_q} < min_{th}. \tag{3}$$

Given $min_{th} = 5$, and $L = 50$, for example, it is necessary to choose $w_q \leq 0.0042$.

6.2 A lower bound for w_q

RED gateways are designed to keep the calculated average queue size avg below a certain threshold. However, this serves little purpose if the calculated average avg is not a reasonable reflection of the current average queue size. If w_q is set too low, then avg responds too slowly to changes in the actual queue size. In this case, the gateway is unable to detect the initial stages of congestion.

Assume that the queue changes from empty to one packet, and that, as packets arrive and depart at the same rate, the queue remains at one packet. Further assume that initially the average queue size was zero. In this case it takes $-1/\ln(1-w_q)$ packet arrivals (with the queue size remaining at one) until the average queue size avg reachs $0.63 = 1-1/e$ [35]. For $w_q = 0.001$, this takes 1000 packet arrivals; for $w_q = 0.002$, this takes 500 packet arrivals; for $w_q = 0.003$, this takes 333 packet arrivals. In most of our simulations we use $w_q = 0.002$.

6.3 Setting min_{th} and max_{th}

The optimal values for min_{th} and max_{th} depend on the desired average queue size. If the typical traffic is fairly bursty, then min_{th} must be correspondingly large to allow the link utilization to be maintained at an acceptably high level. For the typical traffic in our simulations, for connections with reasonably large delay-bandwidth products, a minimum threshold of one packet would result in unacceptably low link utilization. The discussion of the optimal average queue size for a particular traffic mix is left as a question for future research.

The optimal value for max_{th} depends in part on the maximum average delay that can be allowed by the gateway.

The RED gateway functions most effectively when $max_{th} - min_{th}$ is larger than the typical increase in the calculated average queue size in one roundtrip time. A useful rule-of-thumb is to set max_{th} to at least twice min_{th}.

7 Calculating the packet-marking probability

The initial packet-marking probability p_b is calculated as a linear function of the average queue size. In this section we compare two methods for calculating the final packet-marking probability, and demonstrate the advantages of the second method. In the first method, when the average queue size is constant the number of arriving packets between marked packets is a geometric random variable; in the second method the number of arriving packets between marked packets is a uniform random variable.

The initial packet-marking probability is computed as follows:

$$p_b \leftarrow max_p(avg - min_{th})/(max_{th} - min_{th}).$$

The parameter max_p gives the maximum value for the packet-marking probability p_b, achieved when the average queue size reaches the maximum threshold.

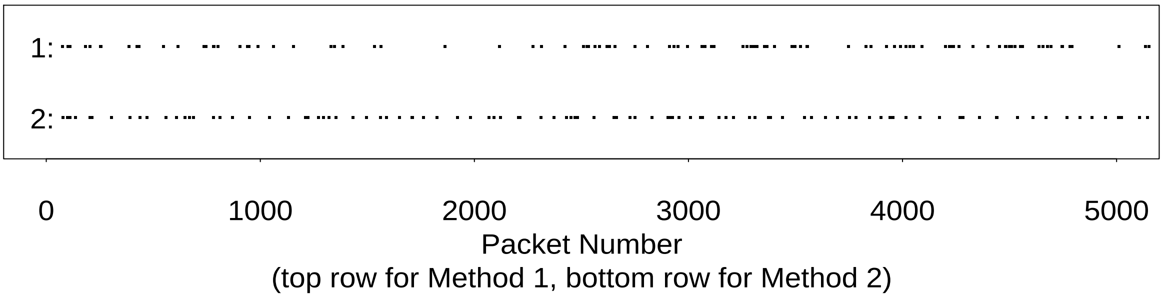

Figure 8: Randomly-marked packets, comparing two packet-marking methods.

Method 1: Geometric random variables. In Method 1, let each packet be marked with probability p_b. Let the intermarking time X be the number of packets that arrive, after a marked packet, until the next packet is marked. Because each packet is marked with probability p_b,

$$Prob[X = n] = (1 - p_b)^{n-1} p_b.$$

Thus with Method 1, X is a *geometric* random variable with parameter p_b, and $E[X] = 1/p_b$.

With a constant average queue size, the goal is to mark packets at fairly regular intervals. It is undesirable to have too many marked packets close together, and it is also undesirable to have too long an interval between marked packets. Both of these events can result in global synchronization, with several connections reducing their windows at the same time, and both of these events can occur when X is a geometric random variable. □

Method 2: Uniform random variables. A more desirable alternative is for X to be a *uniform* random variable from $\{1, 2, ..., 1/p_b\}$ (assuming for simplicity that $1/p_b$ is an integer). This is achieved if the marking probability for each arriving packet is $p_b/(1 - count \cdot p_b)$, where *count* is the number of unmarked packets that have arrived since the last marked packet. Call this Method 2. In this case,

$$\begin{aligned} Prob[X = n] &= \frac{p_b}{1-(n-1)p_b} \prod_{i=0}^{n-2} \left(1 - \frac{p_b}{1 - i\, p_b}\right) \\ &= p_b \text{ for } 1 \le n \le 1/p_b, \end{aligned}$$

and

$$Prob[X = n] = 0 \text{ for } n > 1/p_b.$$

For Method 2, $E[X] = 1/(2p_b) + 1/2$. □

Figure 8 shows an experiment comparing the two methods for marking packets. The top line shows Method 1, where each packet is marked with probability p, for $p = 0.02$. The bottom line shows Method 2, where each packet is marked with probability $p/(1 + i\,p)$, for $p = 0.01$, and for i the number of unmarked packets since the last marked packet. Both methods marked roughly 100 out of the 5000 arriving packets. The x-axis shows the packet number. For each method, there is a dot for each marked packet. As expected, the marked packets are more clustered with Method 1 than with Method 2.

For the simulations in this paper, we set max_p to 1/50. When the average queue size is halfway between min_{th} and max_{th}, the gateway drops, on the average, roughly one out of 50 (or one out of $1/max_p$) of the arriving packets. RED gateways perform best when the packet-marking probability changes fairly slowly as the average queue size changes; this helps to discourage oscillations in the average queue size and in the packet-marking probability. There should never be a reason to set max_p greater than 0.1, for example. When $max_p = 0.1$, then the RED gateway marks close to 1/5th of the arriving packets when the average queue size is close to the maximum threshold (using Method 2 to calculate the packet-marking probability). If congestion is sufficiently heavy that the average queue size cannot be controlled by marking close to 1/5th of the arriving packets, then after the average queue size exceeds the maximum threshold, the gateway will mark every arriving packet.

8 Evaluation of RED gateways

In addition to the design goals discussed in Section 3, several general goals have been outlined for congestion avoidance schemes [14, 16]. In this section we describe how our goals have been met by RED gateways.

• **Congestion avoidance.** If the RED gateway in fact *drops* packets arriving at the gateway when the average queue size reaches the maximum threshold, then the RED gateway guarantees that the *calculated* average queue size does not exceed the maximum threshold. If the weight w_q for the EWMA procedure has been set appropriately [see Section 6.2], then the RED gateway in fact controls the *actual* average queue size. If the RED gateway *sets a bit* in packet headers when the average queue size exceeds the maximum threshold, rather than dropping packets, then the RED gateway relies on the cooperation of the sources to control the average queue size.

• **Appropriate time scales.** After notifying a connection of congestion by marking a packet, it takes at least a roundtrip time for the gateway to see a reduction in the arrival rate. In RED gateways the time scale for the detection of congestion roughly matches the time scale required for connections to respond to congestion. RED gateways don't notify connections to reduce their windows as a result of transient congestion at the gateway.

• **No global synchronization.** The rate at which RED gateways mark packets depends on the level of congestion. During low congestion, the gateway has a low probability of marking each arriving packet, and as congestion increases, the probability of marking each packet increases. RED gateways avoid global synchronization by marking packets at as low a rate as possible.

• **Simplicity.** The RED gateway algorithm could be implemented with moderate overhead in current networks, as discussed further in Section 11.

• **Maximizing global power**[4]**.** The RED gateway explicitly controls the average queue size. Figure 5 shows that for simulations with high link utilization, global power is higher with RED gateways than with Drop Tail gateways. Future research is needed to determine the optimum average queue size for different network and traffic conditions.

• **Fairness.** One goal for a congestion avoidance mechanism is fairness. This goal of fairness is not well-defined, so we simply describe the performance of the RED gateway in this regard. The RED gateway

[4] *Power is defined as the ratio of throughput to delay.*

does not discriminate against particular connections or classes of connections. (This is in contrast to Drop Tail or Random Drop gateways, as described in [7]). For the RED gateway, the fraction of marked packets for each connection is roughly proportional to that connection's share of the bandwidth. However, RED gateways do not attempt to ensure that each connection receives the same fraction of the total throughput, and do not explicitly control misbehaving users. RED gateways provide a mechanism to identify the level of congestion, and RED gateways could also be used to identify connections using a large share of the total bandwidth. If desired, additional mechanisms could be added to RED gateways to control the throughput of such connections during periods of congestion.

• **Appropriate for a wide range of environments.** The randomized mechanism for marking packets is appropriate for networks with connections with a range of roundtrip times and throughput, and for a large range in the number of active connections at one time. Changes in the load are detected through changes in the average queue size, and the rate at which packets are marked is adjusted correspondingly. The RED gateway's performance is discussed further in the following section.

Even in a network where RED gateways signals congestion by dropping marked packets, there are many occasions in a TCP/IP network when a dropped packet does not result in any decrease in load at the gateway. If the gateway drops a data packet for a TCP connection, this packet drop will be detected by the source, possibly after a retransmission timer expires. If the gateway drops an ACK packet for a TCP connection, or a packet from a non-TCP connection, this packet drop could go unnoticed by the source. However, even for a congested network with a traffic mix dominated by short TCP connections or by non-TCP connections, the RED gateway still controls the average queue size by dropping all arriving packets when the average queue size exceeds a maximum threshold.

8.1 Parameter sensitivity

This section discusses the parameter sensitivity of RED gateways. Unlike Drop Tail gateways, where the only free parameter is the buffer size, RED gateways have additional parameters that determine the upper bound on the average queue size, the time interval over which the average queue size is computed, and the maximum rate for marking packets. The congestion avoidance mechanism should have low parameter sensitivity, and the parameters should be applicable to networks with widely varying bandwidths.

The RED gateway parameters w_q, min_{th}, and max_{th} are necessary so that the network designer can make conscious decisions about the desired average queue size, and about the size and duration in queue bursts to be allowed at the gateway. The parameter max_p can be chosen from a fairly wide range, because it is only an upper bound on the actual marking probability p_b. If congestion is sufficiently heavy that the gateway cannot control the average queue size by marking at most a fraction max_p of the packets, then the average queue size will exceed the maximum threshold, and the gateway will mark every packet until congestion is controlled.

We give a few rules that give adequate performance of the RED gateway under a wide range of traffic conditions and gateway parameters.

1: Ensure adequate calculation of the average queue size: set $w_q \geq 0.001$. The average queue size at the gateway is limited by max_{th}, as long as the calculated average queue size *avg* is a fairly accurate reflection of the actual average queue size. The weight w_q should not be set too low, so that the calculated average queue length does not delay too long in reflecting increases in the actual queue length [See Section 6]. Equation 3 describes the upper bound on w_q required to allow the queue to accommodate bursts of L packets without marking packets.

2: Set min_{th} sufficiently high to maximize network power. The thresholds min_{th} and max_{th} should be set sufficiently high to maximize network power. As we stated earlier, more research is needed on determining the optimal average queue size for various network conditions. Because network traffic is often bursty, the actual queue size can also be quite bursty; if the average queue size is kept too low, then the output link will be underutilized.

3: Make $max_{th} - min_{th}$ sufficiently large to avoid global synchronization. Make $max_{th} - min_{th}$ larger than the typical increase in the average queue size during a roundtrip time, to avoid the global synchronization that results when the gateway marks many packets at one time. One rule of thumb would be to set max_{th} to at least twice min_{th}. If $max_{th} - min_{th}$ is too small, then the computed average queue size can regularly oscillate up to max_{th}; this behavior is similar to the oscillations of the queue up to the maximum queue size with Drop Tail gateways.

To investigate the performance of RED gateways in a range of traffic conditions, this section discusses a simulation with two-way traffic, where there is heavy congestion resulting from many FTP and TELNET connections, each with a small window and limited data to send. The RED gateway parameters are the same as in the simple simulation in Figure 3, but the network traffic is quite different.

Figure 9 shows the simulation, which uses the network in Figure 10. Roughly half of the 41 connections go from one of the left-hand nodes 1-4 to one of the right-hand nodes 5-8; the other connections go in the opposite direction. The roundtrip times for the connections vary by a factor of 4 to 1. Most of the connections are FTP connections, but there are a few TELNET connections. (One of the reasons to keep the average queue size small is to ensure low average delay for the TELNET connections.) Unlike the previous simulations, in this simulation all of the connections have a maximum window of either 8 or 16 packets. The total number of packets for a connection ranges from 20 to 400 packets. The starting times and the total number of packets for each connection were chosen rather arbitrarily; we are not claiming to represent realistic traffic models. The intention is simply to show RED gateways in a range of environments.

Because of the effects of ack-compression with two-way traffic, the packets arriving at the gateway from each connection are somewhat bursty. When ack-packets are 'compressed' in a queue, the ack packets arrive at the source node in a burst. In response, the source sends a burst of data packets [38].

The top chart in Figure 9 shows the queue for gateway A, and the next chart shows the queue for gateway B. For each chart, each 'X' indicates a packet dropped at that gateway. The bottom chart shows the packets for each connection arriving and departing from gateway A (and heading towards gateway B). For each connection, there is a mark for each packet arriving and departing from gateway A, though at this time scale the two marks are indistinguishable. Unlike the chart in Figures 3, in Figure 9 the packets for the different connections are displayed overlapped, rather than displayed on separate rows. The x-axis shows time, and the y-axis shows the packet number for that connection, where each connection starts at packet number 0. For example, the leftmost 'strand' shows a connection that starts at time 0, and that sends 220 packets in all. Each 'X' shows a packet dropped by one of the two gateways. The queue is measured in packets rather in bytes; short packets are just as likely to be dropped as are longer packets. The bottom line of the bottom chart shows again an 'X' for each packet dropped by one of the two gateways.

Because Figure 9 shows many overlapping connections, it is not possible to trace the behavior of each of the connections. As Figure 9 shows, the RED gateway is effective in controlling the average queue size. When congestion is low at one of the gateways, the average queue size and the rate of marking packets is also low at that gateway. As congestion increases at the gateway, the average queue size and the rate of marking packets both increase. Because this simulation consists of heavy congestion caused by many connections, each with a small maximum window, the RED gateways have to drop a fairly large number of packets in order to control congestion. The average link utilization over the one-second period is 61% for the congested link in one direction, and 59% for the other direction. As the figure shows, there are periods at the beginning and the end of the simulation when the arrival rate at the gateways is low.

Note that the traffic in Figures 3 and 9 in quite varied, and in each case the RED gateway adjusts its rate of marking packets to maintain an acceptable average queue size. For the simulations in Figure 9 with many short connections, there are occasional periods of heavy congestion, and a higher rate of packet drops is needed to control congestion. In contrast, with the simulations in Figure 3 with a small number of connections with large maximum windows, the congestion can be controlled with a small number of dropped packets. For the simulations in Figure 9, the burstiness of the queue is dominated by short-term burstiness as packet bursts arrive at the gateway from individual connections. For the simulations in Figure 3, the burstiness of the queue is dominated by the window increase/decrease cycles of the individual connections. Note that the RED gateway parameters are unchanged in these two simulations.

The performance of a slightly different version of RED gateways with connections with different roundtrip times and with connections with multiple congested gateways has been analyzed and explored elsewhere [5].

9 Bursty traffic

This section shows that unlike Drop Tail or Random Drop gateways, RED gateways do not have a bias against bursty traffic.[5] Bursty traffic at the gateway can result from an FTP connection with a long delay-bandwidth product but a small window; a window of traffic will be sent, and then there will be a delay until the ack packets return and another window of data can be sent. Variable-bit-rate video traffic and some forms of interactive traffic are other examples of bursty traffic seen by the gateway.

In this section we use FTP connections with infinite data, small windows, and small roundtrip times to model the less-bursty traffic, and we use FTP connections with smaller windows and longer roundtrip times to model the more-bursty traffic.

We consider simulations of the network in Figure 11. Node 5 packets have a roundtrip time that is six times that of the other packets. Connections 1-4 have a maximum window of 12 packets, while connection 5 has a maximum window of 8 packets. Because node 5 has a large roundtrip time and a small window, node 5 packets 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.

Figures 12 through 14 show the results of simulations of the network in Figure 11 with Drop Tail, Random Drop, and RED gateways respectively. The simulations in Figures 12 and 13 were run with the buffer size ranging from 8 packets to 22 packets. The simulations in Figure 14 were run many times with a minimum threshold ranging from 3 to 14 packets, and a buffer size ranging from 12 to 56 packets.

Each simulation was run for ten seconds, and each mark represents one one-second period of that simulation. For Figures 12 and 13, the x-axis shows the buffer size, and the y-axis shows node 5's throughput as a percentage of the total throughput through the gateway. In order to avoid traffic phase effects (effects caused by the precise timing of packet arrivals at the gateway), in the simulations with Drop Tail gateways the source takes a random time drawn from the uniform distribution on [0, t] seconds to prepare an FTP packet for transmission, where t is the bottleneck service time of 0.17 ms. [7]. In these simulations our concern is to examine the gateway's bias against bursty traffic.

For each set of simulations there is a second figure showing the average queue size (in packets) seen by arriving packets at the bottleneck gateway, and a third figure showing the average link utilization on the congested link. Because RED gateways are quite different from Drop Tail or Random Drop gateways, the gateways cannot be compared simply by comparing the maximum queue size; the most appropriate comparison is between a Drop Tail gateway and a RED gateway that maintain the same average queue size.

With Drop Tail or Random Drop gateways, the queue is more likely to overflow when the queue contains some packets from node 5. In this case, with either Random Drop or Drop Tail gateways, node 5 packets have a disproportionate probability of being dropped; the queue contents when the queue overflows are not representative of the average queue contents.

Figure 14 shows the result of simulations with RED gateways. The x-axis shows min_{th} and the y-axis shows node 5's throughput. The throughput for node 5 is close to the maximum possible throughput,

[5]By bursty traffic we mean traffic from a connection where the amount of data transmitted in one roundtrip time is small compared to the delay-bandwidth product, but where multiple packets from that connection arrive at the gateway in a short period of time.

given node 5's roundtrip time and maximum window. The parameters for the RED gateway are as follows: $w_q = 0.002$ and $max_p = 1/50$. The maximum threshold is twice the minimum threshold and the buffer size, which ranges from 12 to 56 packets, is four times the minimum threshold.

Figure 15 shows that with the simulations with Drop Tail or with Random Drop gateways, node 5 receives a disproportionate share of the packet drops. Each mark in Figure 15 shows the results from a one-second period of simulation. The boxes show the simulations with Drop Tail gateways from Figure 12, the triangles show the simulations with Random Drop gateways from Figure 13, and the dots show the simulations with RED gateways from Figure 14. For each one-second period of simulation, the x-axis shows node 5's throughput (as a percentage of the total throughput) and the y-axis shows node 5's packet drops (as a percentage of the total packet drops). The number of packets dropped in one one-second simulation period ranges from zero to 61; the chart excludes those one-second simulation periods with less than three dropped packets.

The dashed line in Figure 15 shows the position where node 5's share of packet drops exactly equals node 5's share of the throughput. The cluster of dots is roughly centered on the dashed line, indicating that for the RED gateways, node 5's share of dropped packets reflects node 5's share of the throughput. In contrast, for simulations with Random Drop (or with Drop Tail) gateways node 5 receives a small fraction of the throughput but a large fraction of the packet drops. This shows the bias of Drop Tail and Random Drop gateways against the bursty traffic from node 5.

Our simulations with an ISO TP4 network using the DECbit congestion avoidance scheme also show a bias against bursty traffic. With the DECbit congestion avoidance scheme node 5 packets have a disproportionate chance of having their congestion indication bits set. The DECbit congestion avoidance scheme's bias against bursty traffic would be corrected by DECbit congestion avoidance with selective feedback [28], which has been proposed with a fairness goal of dividing each resource equally among all of the users sharing it. This modification uses a selective feedback algorithm at the gateway. The gateway determines which users are using more than their "fair share" of the bandwidth, and only sets the congestion-indication bit in packets belonging to those users. We have not run simulations with this algorithm.

10 Identifying misbehaving users

In this section we show that RED gateways provide an efficient mechanism for identifying connections that use a large share of the bandwidth in times of congestion. Because RED gateways randomly choose packets to be marked during congestion, RED gateways could easily identify which connections have received a significant fraction of the recently-marked packets. When the number of marked packets is sufficiently large, a connection that has received a large share of the marked packets is also likely to be a connection that has received a large share of the bandwidth. This information could be used by higher policy layers to restrict the bandwidth of those connections during congestion.

The RED gateway notifies connections of congestion at the gateway by marking packets. With RED gateways, when a packet is marked, the probability of marking a packet from a particular connection is roughly proportional to that connection's current share of the bandwidth through the gateway. Note that this property does not hold for Drop-Tail gateways, as demonstrated in Section 9.

For the rest of this section, we assume that each time the gateway marks a packet, the probability that a packet from a particular connection is marked *exactly* equals that connection's fraction of the bandwidth through the gateway. Assume that connection i has a fixed fraction p_i of the bandwidth through the gateway. Let $S_{i,n}$ be the number of the n most-recently-marked packets that are from connection i. From the assumptions above, the expected value for $S_{i,n}$ is np_i.

From standard statistical results given in the appendix, $S_{i,n}$ is unlikely to be much larger than its expected value for sufficiently large n:

$$Prob(S_{i,n} \geq cp_i n) \leq e^{-2n(c-1)^2 p_i^2}$$

for $1 \leq c \leq 1/p_i$. The two lines in Figure 16 show the upper bound on the probability that a connection receives more than C times the expected number of marked packets, for $C = 2, 4$, and for $n = 100$; the x-axis shows p_i.

The RED gateway could easily keep a list of the n most recently-marked packets. If some connection has a large fraction of the marked packets, it is likely that the connection also had a large fraction of the average bandwidth. If some TCP connection is receiving a large fraction of the bandwidth, that connection could be a misbehaving host that is not following current TCP protocols, or simply a connection with either a shorter roundtrip time or a larger window than other active connections. In either case, if desired, the RED gateway could be modified to give lower priority to those connections that receive a large fraction of the bandwidth during times of congestion.

11 Implementation

This section considers efficient implementations of RED gateways. We show that the RED gateway algorithm can be implemented efficiently, with only a small number of add and shift instructions for each packet arrival. In addition, the RED gateway algorithm is not tightly coupled to packet forwarding and its computations do not have to be made in the time-critical packet forwarding path. Much of the work of the RED gateway algorithm, such as the computation of the average queue size and of the packet-marking probability p_b, could be performed in parallel with packet forwarding, or could be computed by the gateway as a lower-priority task as time permits. This means that the RED gateway algorithm need not impair the gateway's ability to process packets, and the RED gateway algorithm can be adapted to increasingly-high-speed output lines.

If the RED gateway's method of marking packets is to set a congestion indication bit in the packet header, rather than dropping the arriving packet, then setting the congestion indication bit itself adds overhead to the gateway algorithm. However, because RED gateways are designed to mark as few packets as possible, the overhead of setting the congestion indication bit is kept to a minimum. This is unlike DECbit gateways, for example, which set the congestion indication bit in every packet that arrives at the gateway when the average queue size exceeds the threshold.

For every packet arrival at the gateway queue, the RED gateway calculates the average queue size. This can be implemented as follows:

$$avg \leftarrow avg + w_q\,(q - avg)$$

As long as w_q is chosen as a (negative) power of two, this can be implemented with one shift and two additions (given scaled versions of the parameters) [14].

Because the RED gateway computes the average queue size at packet arrivals, rather than at fixed time intervals, the calculation of the average queue size is modified when a packet arrives at the gateway to an empty queue. After the packet arrives at the gateway to an empty queue the gateway calculates m, the number of packets that might have been transmitted by the gateway during the time that the line was free. The gateway calculates the average queue size *as if* m packets had arrived at the gateway with a queue size of zero. The calculation is as follows:

$$m \leftarrow (time - q\_time)/s$$
$$avg \leftarrow (1 - w_q)^m\, avg,$$

where *q_time* is the start of the queue idle time, and s is a typical transmission time for a small packet. This entire calculation is an approximation, as it is based on the number of packets that *might* have arrived at the gateway during a certain period of time. After the idle time $(time - q\_time)$ has been computed to a rough level of accuracy, a table lookup could be used to get the term $(1-w_q)^{(time-q\_time)/s}$, which could itself be an approximation by a power of two.

When a packet arrives at the gateway and the average queue size *avg* exceeds the threshold max_{th}, the arriving packet is marked. There is no recalculation of the packet-marking probability. However, when a packet arrives at the gateway and the average queue size *avg* is between the two thresholds min_{th} and max_{th}, the initial packet-marking probability p_b is calculated as follows:

$$p_b \leftarrow C_1\, avg - C_2$$

for

$$C_1 = \frac{max_p}{max_{th} - min_{th}},$$

$$C_2 = \frac{max_p\, min_{th}}{max_{th} - min_{th}}.$$

The parameters max_p, max_{th}, and min_{th} are fixed parameters that are determined in advance. The values for max_{th} and min_{th} are determined by the desired bounds on the average queue size, and might have limited flexibility. The fixed parameter max_p, however, could easily be set to a range of values. In particular, max_p could be chosen so that C_1 is a power of two. Thus, the calculation of p_b can be accomplished with one shift and one add instruction.

In the algorithm described in Section 4, when $min_{th} \leq avg < max_{th}$ a new pseudo-random number R is computed for each arriving packet, where $R = Random[0,1]$ is from the uniform distribution on [0,1]. These random numbers could be gotten from a table of random numbers stored in memory or could be computed fairly efficiently on a 32-bit computer [3]. In the algorithm described in Section 4, the arriving packet is marked if

$$R < p_b/(1 - count \cdot p_b).$$

If p_b is approximated by a negative power of two, then this can be efficiently computed.

It is possible to implement the RED gateway algorithm to use a new random number only once for every marked packet, instead of using a new random number for every packet that arrives at the gateway when $min_{th} \leq avg < max_{th}$. As Section 7 explains, when the average queue size is constant the number of packet arrivals after a marked packet until the next packet is marked is a uniform random variable from $\{1, 2, ..., \lfloor 1/p_b \rfloor\}$. Thus, if the average queue size was constant, then after each packet is marked the gateway could simply choose a value for the uniform random variable $R = Random[0,1]$, and mark the n-th arriving packet if $n \geq R/p_b$. Because the average queue size changes over time, we recompute R/p_b each time that p_b is recomputed. If p_b is approximated by a negative power of two, then this can be computed using a shift instruction instead of a divide instruction.

Figure 17 gives the pseudocode for an efficient version of the RED gateway algorithm. This is just one suggestion for an efficient version of the RED gateway algorithm. The *most* efficient way to implement this algorithm depends, of course, on the gateway in question.

The memory requirements of the RED gateway algorithm are modest. Instead of keeping state for each active connection, the RED gateway requires a small number of fixed and variable parameters for each output line. This is not a burden on gateway memory.

12 Further work and conclusions

Random Early Detection gateways are an effective mechanism for congestion avoidance at the gateway, in cooperation with network transport protocols. If RED gateways *drop* packets when the average queue size exceeds the maximum threshold, rather than simply setting a bit in packet headers, then RED gateways control the calculated average queue size. This action provides an upper bound on the average delay at the gateway.

The probability that the RED gateway chooses a particular connection to notify during congestion is roughly proportional to that connection's share of the bandwidth at the gateway. This approach avoids a bias against bursty traffic at the gateway. For RED gateways, the rate at which the gateway marks packets depends on the level of congestion, avoiding the global synchronization that results from many connections decreasing their windows at the same time. The RED gateway is a relatively simple gateway algorithm that could be implemented in current networks or in high-speed networks of the future. The RED gateway allows conscious design decisions to be made about the average queue size and the maximum queue size allowed at the gateway.

There are many areas for further research on RED gateways. The foremost open question involves determining the optimum average queue size for maximizing throughput and minimizing delay for various network configurations. This question is heavily dependent of the characterization of the network traffic as well as on the physical characteristics of the network. Some work has been done in this area for other congestion avoidance algorithms [23], but there are still many open questions.

One area for further research concerns traffic dynamics with a mix of Drop Tail and RED gateways, as would result from partial deployment of RED gateways in the current internet. Another area for further research concerns the behavior of the RED gateway machinery with transport protocols other than TCP, including open- or closed-loop rate-based protocols.

As mentioned in Section 10, the list of packets marked by the RED gateway could be used by the gateway to identify connections that are receiving a large fraction of the bandwidth through the gateway. The gateway could use this information to give such connections lower priority at the gateway. We leave this as an area for further research.

We do not specify in this paper whether the queue size should be measured in bytes or in packets. For networks with a range of packet sizes at the congested gateway the difference can be significant. This includes networks with two-way traffic where the queue at the congested gateway contains large FTP packets, small TELNET packets, and small control packets. For a network where the time required to transmit a packet is proportional to the size of the packet, and the gateway queue is measured in bytes, the queue size reflects the delay in seconds for a packet arriving at the gateway.

The RED gateway is not constrained to provide strict FIFO service. For example, we have experimented with a version of RED gateways that provides priority service for short control packets, reducing problems with compressed ACKs.

By controlling the average queue size *before* the gateway queue overflows, RED gateways could be particularly useful in networks where it is undesirable to drop packets at the gateway. This would be the case, for example, in running TCP transport protocols over cell-based networks such as ATM. There are serious performance penalties for cell-based networks if a large number of cells are dropped at the gateway; in this case it is possible that many of the cells successfully transmitted belong to a packet in which *some* cell was dropped at a gateway [30]. By providing advance warning of incipient congestion, RED gateways can be useful in avoiding unnecessary packet or cell drops at the gateway.

The simulations in this paper use gateways where there is one output queue for each output line, as in most gateways in current networks. RED gateways could also be used in routers with resource management where different *classes* of traffic are treated differently and each class has its own queue [6]. For example, in a router where interactive (TELNET) traffic and bulk data (FTP) traffic are in separate classes with separate queues (in order to give priority to the interactive traffic), each class could have a separate Random Early Detection queue. The general issue of resource management at gateways will be addressed in future papers.

13 Acknowledgements

We thank Allyn Romanow for discussions on RED gateways in ATM networks, and we thank Vern Paxson and the referees for helpful suggestions. This work could not have been done without Steven McCanne, who developed and maintained our simulator.

References

[1] 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.

[2] Bala, K., Cidon, I., and Sohraby, K., "Congestion Control for High Speed Packet Switched Networks", INFOCOM '90, pp. 520-526, 1990.

[3] Carta, D., "Two Fast Implementations of the 'Minimal Standard' Random Number Generator", Communications of the ACM, V.33 N.1, January 1990, pp. 87-88.

[4] Clark, D.D., Shenker, S., and Zhang, L., "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", SIGCOMM '92, August 1992, p. 14-26.

[5] 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.

[6] Floyd, S., "Issues in Flexible Resource Management for Datagram Networks", Proceedings of the 3rd Workshop on Very High Speed Networks, March, 1992.

[7] Floyd, S., and Jacobson, V., *On Traffic Phase Effects in Packet-Switched Gateways*, Internetworking: Research and Experience, V.3 N.3, September 1992, p.115-156.

[8] Floyd, S., and Jacobson, V., *The Synchronization of Periodic Routing Messages*, to appear in SIGCOMM 93.

[9] Hansen, *A Table of Series and Products*, Prentice Hall, Englewood Cliffs, NJ, 1975.

[10] Harvey, A., Forecasting, structural time series models and the Kalman filter, Cambridge University Press, 1989.

[11] Hashem, E., "Analysis of random drop for gateway congestion control", *Report LCS TR-465*, Laboratory for Computer Science, MIT, Cambridge, MA, 1989, p.103.

[12] Hoeffding, W., *Probability Inequalities for Sums of Bounded Random Variables*, American Statistical Association Journal, Vol. 58, March 1963, p. 13-30.

[13] Hofri, M., Probabilistic Analysis of Algorithms, Springer-Verlag, 1987.

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

[15] Jain, R., "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks", Computer Communication Review, V.19 N.5, October 1989, pp. 56-71.

[16] Jain, R., "Congestion Control in Computer Networks: Issues and Trends", IEEE Network, May 1990, pp. 24-30.

[17] Jain, R., "Myths About Congestion Management in High-Speed Networks", Internetworking: Research and Experience, V.3 N.3, September 1992, pp. 101-114.

[18] Jain, R., and Ramakrishnan, K.K., *Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals, and Methodology*, Proc. IEEE Comp. Networking Symp., Washington, D.C., April 1988, pp.134-143.

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

[20] Keshav, S., "A Control-Theoretic Approach to Flow Control", Proceedings of SIGCOMM '91, September 1991, p.3-16.

[21] Mankin, A. and Ramakrishnan, K. K., editors for the IETF Performance and Congestion Control Working Group, "Gateway congestion control survey", RFC 1254, August 1991, p.21.

[22] Mishra, P., and Kanakia, H,. "A Hop by Hop Rate-based Congestion Control Scheme", Proceedings of SIGCOMM '92, August 1992, p.112-123.

[23] Mitra, D. and Seery, J., *Dynamic Adaptive Windows for High Speed Data Networks: Theory and Simulations*, Proceedings of SIGCOMM '90, September 1990, p.30-40.

[24] Mitra, D. and Seery, J., *Dynamic Adaptive Windows for High Speed Data Networks with Multiple Paths and Propagation Delays*, Proc. IEEE INFOCOM '91, pp. 2B.1.1-2B.1.10.

[25] Pingali, S., Tipper, D., and Hammond, J., "The Performance of Adaptive Window Flow Controls in a Dynamic Load Environment", Proc. of IEEE Infocom '90, June 1990, pp. 55-62.

[26] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.

[27] Prue, W., and Postel, J., "Something a Host Could Do with Source Quench", RFC 1016, July 1987.

[28] Ramakrishnan, K.K., Chiu, D., and Jain, R., "Congestion Avoidance in Computer Networks with a Connectionless Network Layer, Part IV: A Selective Binary Feedback Scheme for General Topologies", DEC-TR-510, November, 1987.

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

[30] Romanow, A., "Some Performance Results for TCP over ATM with Congestion", to appear in Second IEEE Workshop on Architecture and Implementation of High Performance Communications Subsystems (HPCS 93), Williamsburg, VA, Sept. 1-3, 1993.

[31] Sanghi, D., and Agrawala, A., "DTP: An Efficient Transport Protocol", University of Maryland tech report UMIACS-TR-91-133, October 1991.

[32] Shenker, S., "Comments on the IETF performance and congestion control working group draft on gateway congestion control policies", unpublished, 1989.

[33] Wang, Z., and Crowcroft, J., "A New Congestion Control Scheme: Slow Start and Search (Tri-S)", Computer Communication Review, V.21 N.1, January 1991, pp. 32-43.

[34] Wang, Z., and Crowcroft, J., "Eliminating Periodic Packet Losses in the 4.3-Tahoe BSD TCP Congestion Control Algorithm", Computer Communication Review, V.22 N.2, April 1992, pp. 9-16.

[35] Young, P., Recursive Estimation and Time-Series Analysis, Springer-Verlag, 1984, pp. 60-65.

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

[37] Zhang, L., and Clark, D., "Oscillating Behavior of Network Traffic: A Case Study Simulation", Internetworking: Research and Experience, Vol. 1, 1990, pp. 101-112.

[38] Zhang, L., Shenker, S., and Clark, D., "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic", SIGCOMM '91, September 1991, pp. 133-148.

A Appendix

In this section we give the statistical result used in Section 10 on identifying misbehaving users.

Let X_j, $1 \leq j \leq n$, be independent random variables, let S be their sum, and let $\bar{X} = S/n$.

Theorem 1 (Hoeffding, 1963) *[12, p.15] [13, p.104]: Let* $X_1, X_2, \ldots, X_n$ *be independent, and let* $0 \leq X_j \leq 1$ *for all* X_j. *Then for* $0 \leq t \leq 1 - E[\bar{X}]$,

$$\begin{aligned} & Prob[\bar{X} \geq E[\bar{X}] + t] \\ \leq \quad & \left[\left(\frac{\mu}{\mu + t} \right)^{\mu + t} \left(\frac{1 - \mu}{1 - \mu - t} \right)^{1 - \mu - t} \right]^n \\ \leq \quad & e^{-2nt^2}. \end{aligned} \tag{4}$$

□

Let $X_{i,j}$ be an indicator random variable that is 1 if the jth marked packet is from connection i, and 0 otherwise. Then

$$S_{i,n} = \sum_{j=1}^{n} X_{i,j}.$$

From Theorem 1,

$$Prob(S_{i,n} \geq p_i\, n + t\, n) \leq e^{-2nt^2}$$

for $0 \leq t \leq 1 - p_i$. Thus

$$Prob(S_{i,n} \geq c p_i n) \leq e^{-2n(c-1)^2 p_i^2}$$

for $1 \leq c \leq 1/p_i$.

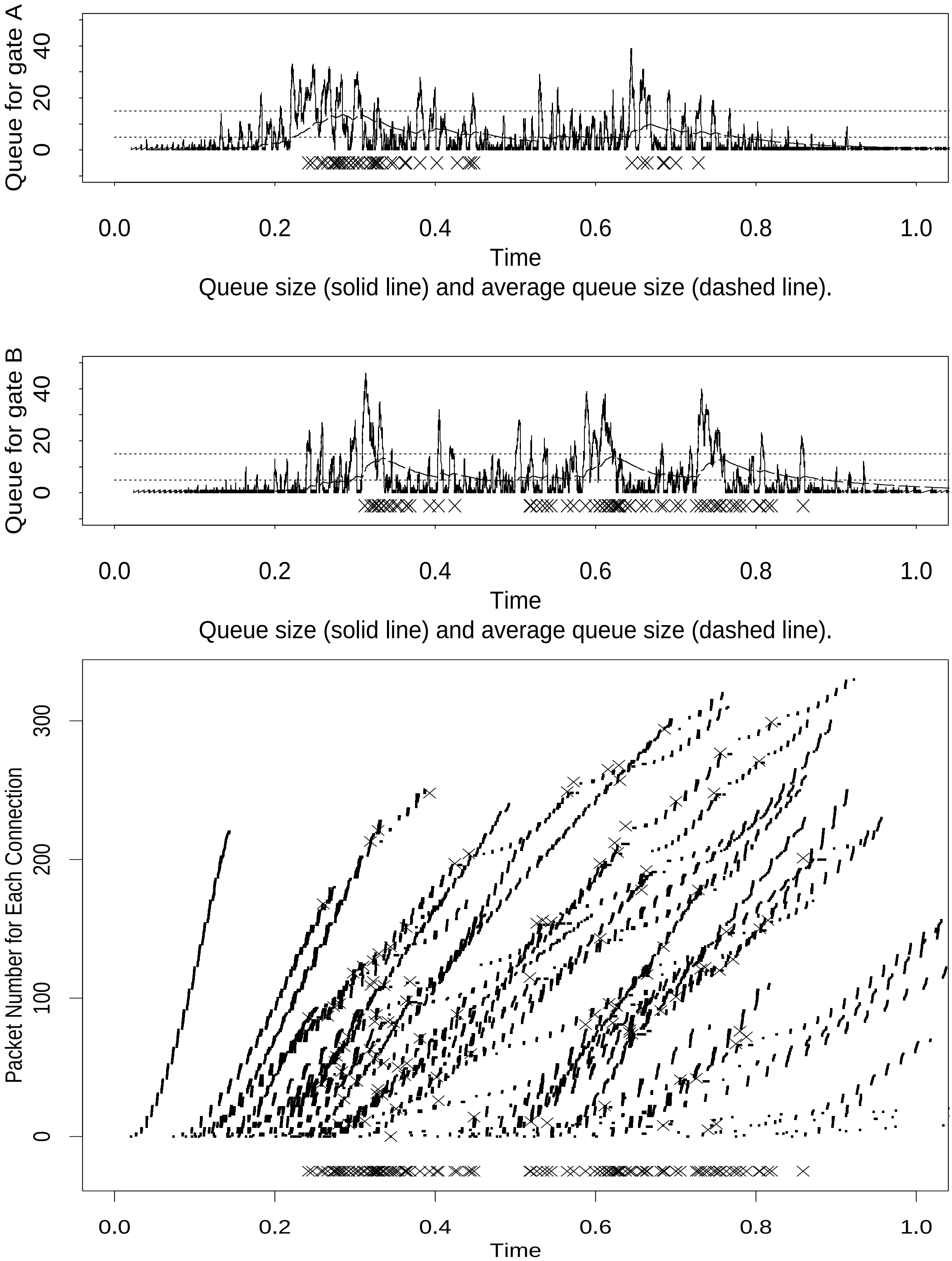

Figure 9: A RED gateway simulation with heavy congestion, two-way traffic, and many short FTP and TELNET connections.

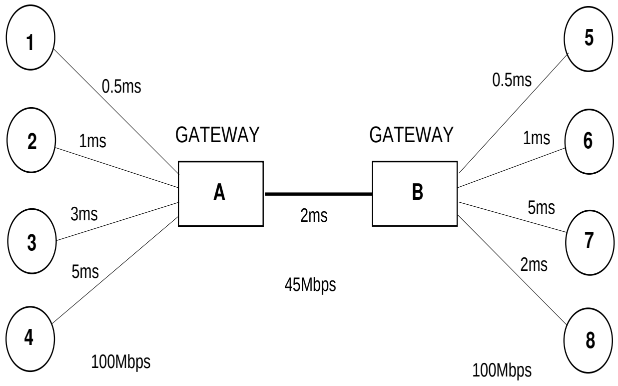

Figure 10: A network with many short connections.

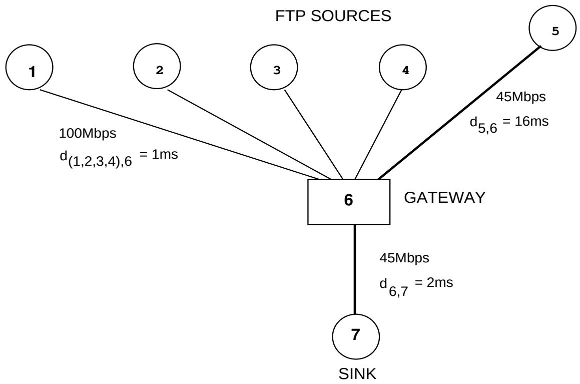

Figure 11: A simulation network with five FTP connections.

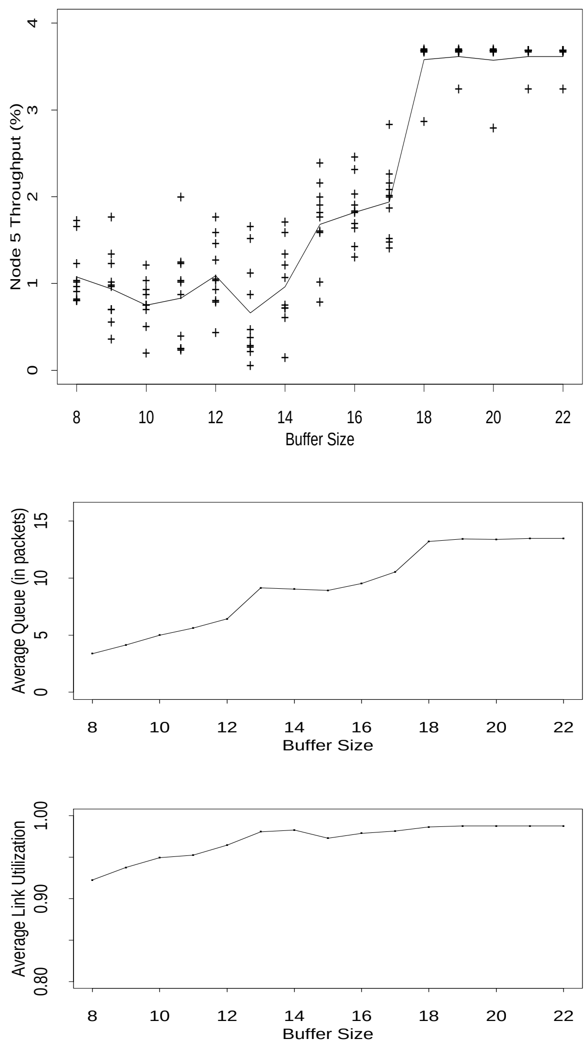

Figure 12: Simulations with Drop Tail gateways.

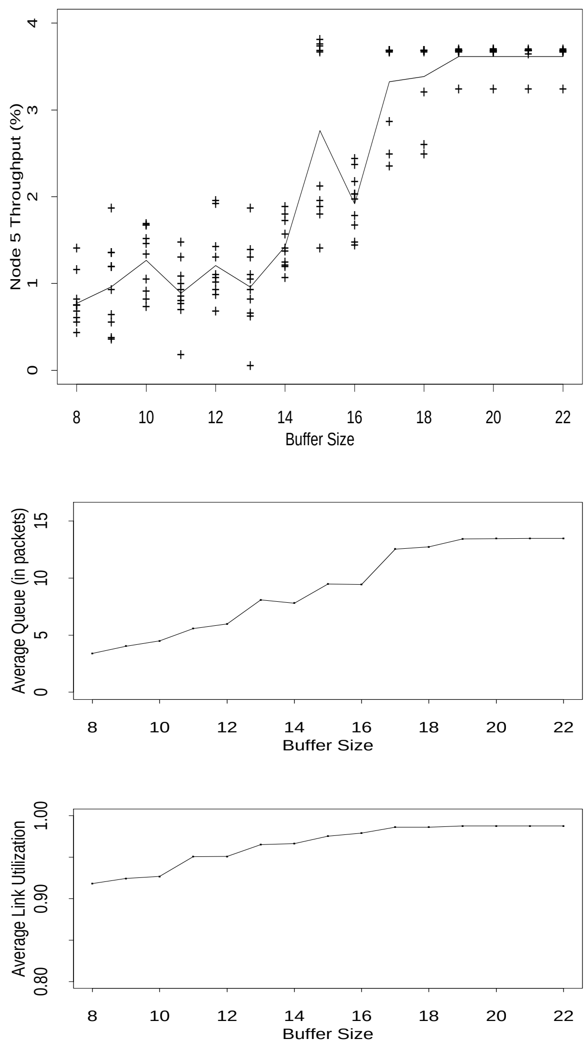

Figure 13: Simulations with Random Drop gateways.

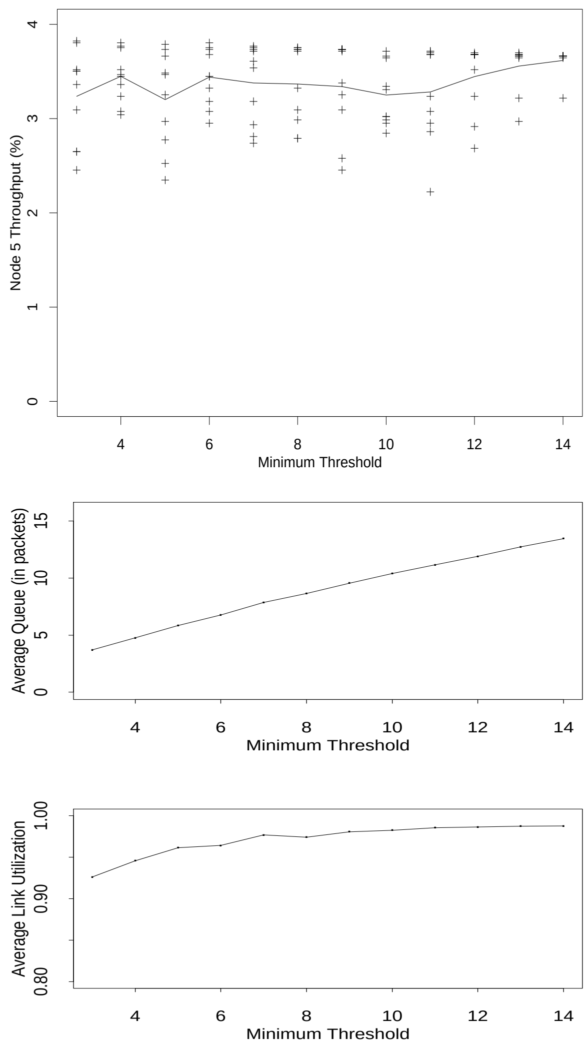

Figure 14: Simulations with RED gateways

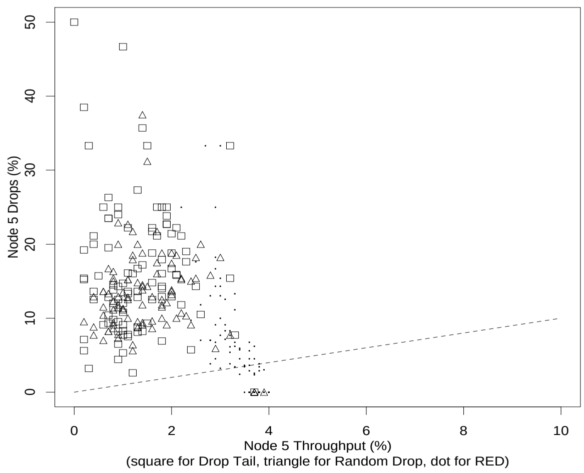

Figure 15: Scatter plot, packet drops vs. throughput

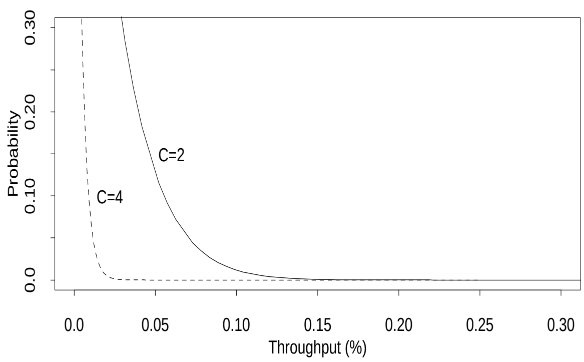

Figure 16: Upper bound on probability that a connection's fraction of marked packets is more than C times the expected number, given 100 total marked packets.

```
Initialization:
   avg ← 0
   count ← −1
for each packet arrival:
   calculate the new average queue size avg:
      if the queue is nonempty
         avg ← avg + w_q (q − avg)
      else using a table lookup:
         avg ← (1 − w_q)^((time−q_time)/s) avg
   if min_th ≤ avg < max_th
      increment count
      p_b ← C_1 · avg − C_2
      if count > 0 and count ≥ Approx[R/p_b]
         mark the arriving packet
         count ← 0
      if count = 0 (choosing random number)
         R ← Random[0,1]
   else if max_th ≤ avg
      mark the arriving packet
      count ← −1
   else count ← −1
when queue becomes empty
   q_time ← time
```

New variables:

R: a random number

New fixed parameters:

s: typical transmission time

Figure 17: Efficient algorithm for RED gateways.