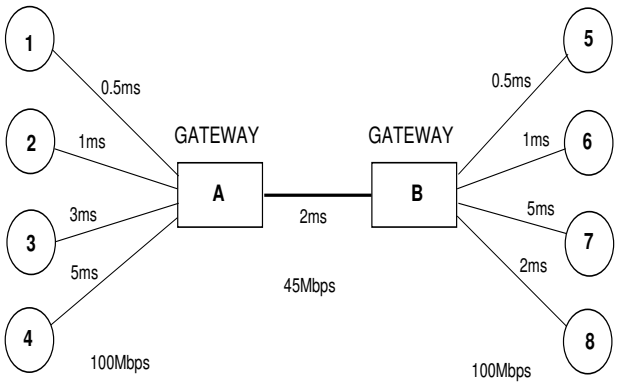

Figure 10: A network with many short connections.

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.

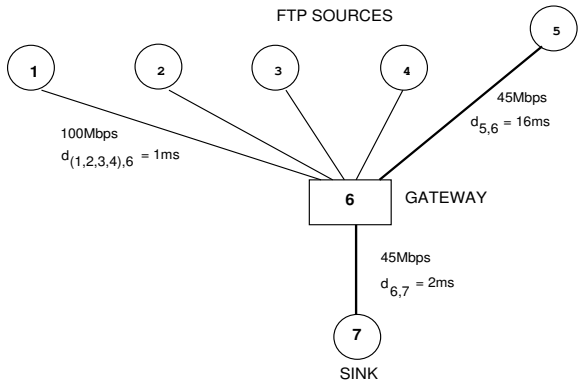

Figure 11: A simulation network with five FTP connections.

[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.

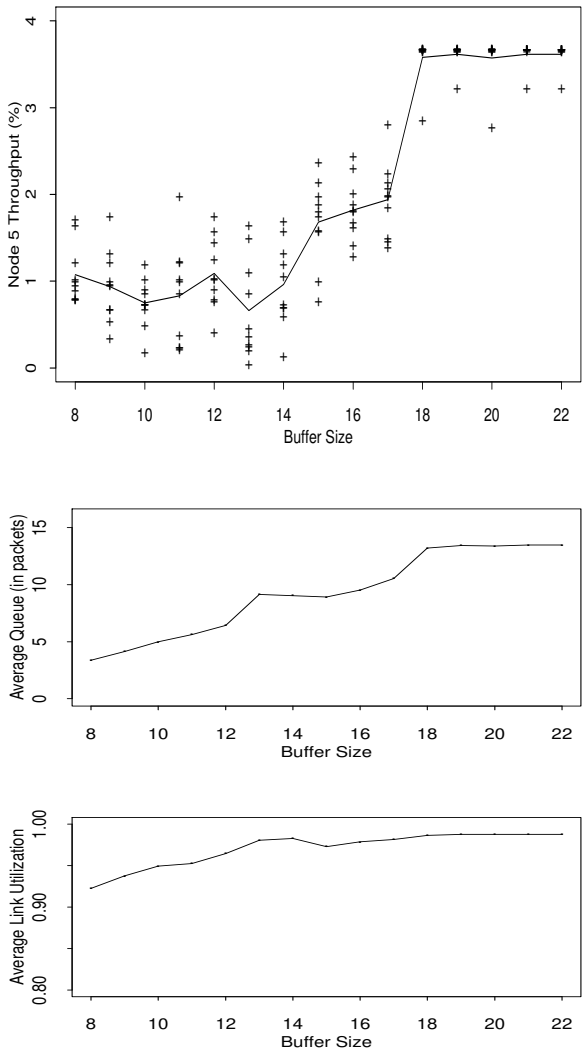

Figure 12: Simulations with Drop Tail gateways.

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

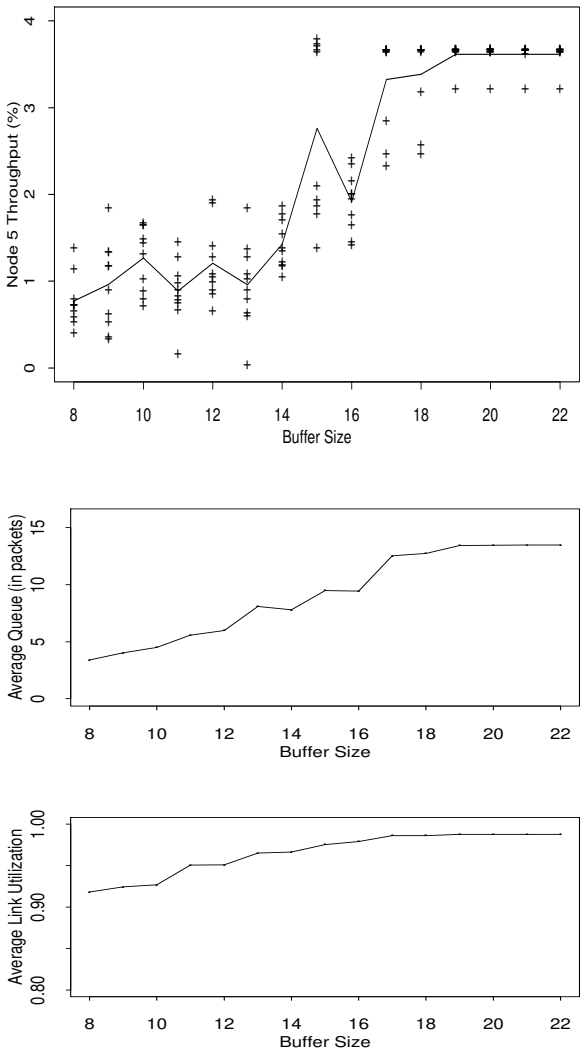

Figure 13: Simulations with Random Drop gateways.

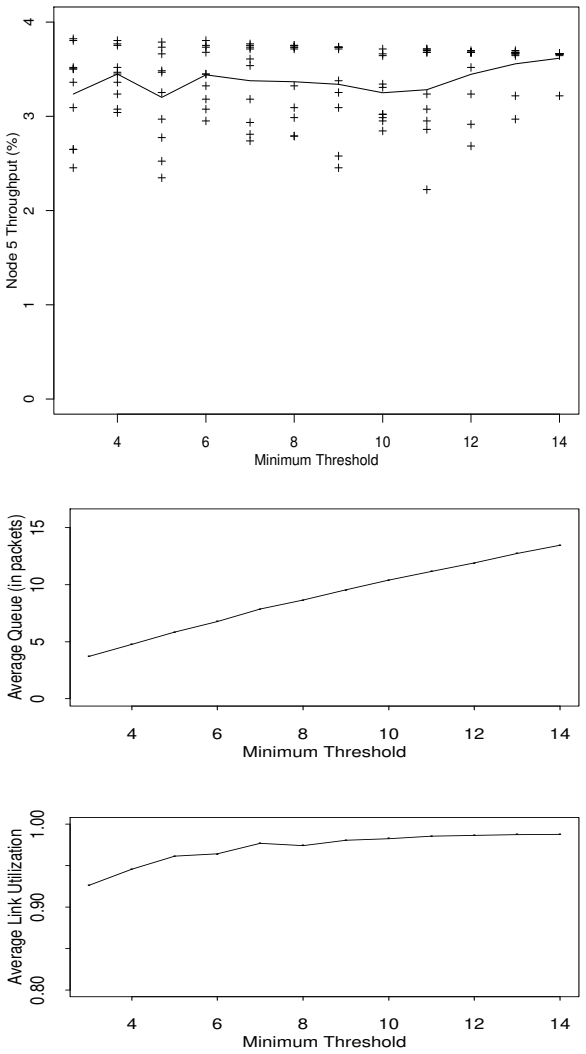

Figure 14: Simulations with RED gateways

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, 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.

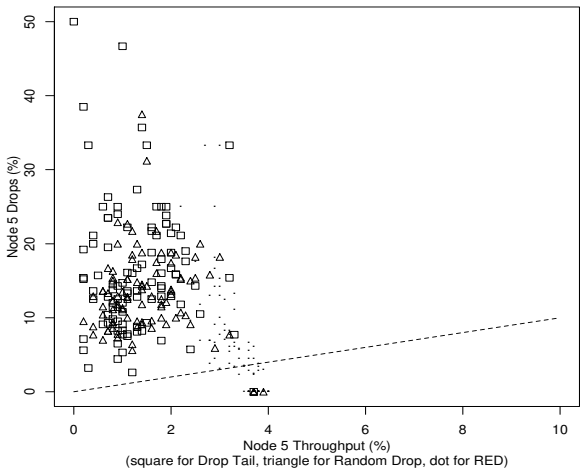

Figure 15: Scatter plot, packet drops vs. throughput

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.

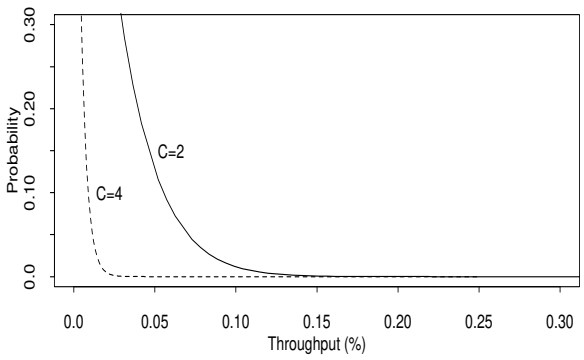

Figure 16: Upper bound on probability that a connection's fraction of marked packets is more than C times the expected number, given 100 total marked packets.

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.

```
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.

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 In-

tegrated 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*, Proceedings of SIGCOMM '88, August 1988.

[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.