Router Mechanisms to Support End-to-End Congestion Control

Sally Floyd and Kevin Fall[*]

Network Research Group
Lawrence Berkeley National Laboratory, Berkeley CA
{*floyd,kfall*}*@ee.lbl.gov*

February 15, 1997

Abstract

This paper considers the potential negative impacts from an increasing deployment of non-congestion-controlled best-effort traffic on the Internet. These negative impacts range from extreme unfairness against competing TCP traffic to the potential for congestion collapse. To promote the inclusion of end-to-end congestion control for best-effort traffic, we propose lightweight router mechanisms for identifying and restricting the bandwidth of high-bandwidth best-effort flows that are using a disproportionate share of the bandwidth in times of congestion. Our method does not require per-flow state based on packet arrivals, but instead relies on the history of packet drops from a queue with RED (Random Early Detection) queue management.

Starting with high-bandwidth flows identified from the RED drop history, we describe a sequence of tests capable of suggesting flows for bandwidth regulation. These tests additionally identify a high-bandwidth flow in times of congestion as unresponsive, "not TCP-friendly", or simply very-high-bandwidth. An *unresponsive flow* is one failing to reduce its offered load at a router in response to an increased packet drop rate. A flow that is *not TCP-friendly* is one whose long-term arrival rate exceeds that of any conformant TCP in the same circumstances. A *very-high-bandwidth flow* uses a disproportionate share of the bandwidth relative to other flows. Simulations show the results of regulating the bandwidth of these unresponsive, TCP-unfriendly, or very-high-bandwidth flows in times of congestion. We end with a comparison between this approach and others using per-flow scheduling for all best-effort traffic.

1 Introduction

The use of the end-to-end congestion control mechanisms of TCP has been a critical factor in the robustness of the Internet today. However, the Internet is no longer a small, closely knit user community, and it is no longer possible to rely on all end-nodes to use end-to-end congestion control for best-effort traffic. Similarly, it is no longer possible to rely on all developers to incorporate end-to-end congestion control in their Internet applications. The network itself must now participate in controlling its own resource utilization.

This leads to several possible approaches for best-effort traffic sharing scarce bandwidth in the Internet. One approach is to propose, as the primary mechanism for sharing scarce bandwidth, that routers isolate each flow, as much as possible, from the effects of other flows [She94]. This approach suggests the deployment of per-flow scheduling mechanisms that separately regulate the bandwidth used by each best-effort flow.

A second approach outlined in this paper is for routers to support the continued use of end-to-end congestion control as the primary mechanism for best-effort traffic to share scarce bandwidth, but to deploy additional mechanisms to restrict the bandwidth of best-effort flows using a disproportionate share of the bandwidth in times of congestion. These mechanisms would give a concrete incentive to end-users, application developers, and protocol designers to use end-to-end congestion control for best-effort traffic.

A third approach would be to rely on pricing mechanisms to control sharing, and to gamble that the network providers are able to provision additional bandwidth and deploy appropriate pricing structures fast enough to keep up with the growth in unresponsive best-effort traffic in the Internet.

These three approaches to sharing, of isolating flows at the router, deploying concrete incentives for best-effort traffic to use end-to-end congestion control, and relying on pricing mechanisms, are not necessarily mutually exclusive. Furthermore, given the fundamental heterogeneity of the Internet, there is no requirement that all routers or all service providers follow the same approach.

However, these three approaches can lead to different conclusions about the role of end-to-end congestion control for best-effort traffic, and different consequences in terms of the

[*] This work was supported by the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098, and by ARPA grant DABT63-96-C-0105.

increasing deployment of such traffic in the Internet. The Internet as now at a cross-roads in terms of the use of end-to-end congestion control for best-effort traffic, and is in a position to actively welcome the widespread deployment of non-congestion-controlled best-effort traffic, to actively discourage such a widespread deployment, or, by taking no action, to allow such a widespread deployment to become a simple fact of life. We argue of this paper that reiterating the essential role of end-to-end feedback for best-effort traffic and strengthening incentives for best-effort flows to use end-to-end congestion control are critical issues as the Internet expands to a larger community.

As we show in Section 2, an increasing deployment of traffic lacking end-to-end feedback could lead to congestion collapse in the Internet. This form of congestion collapse would result from congested links sending packets that would only be dropped later in the network. This form of congestion collapse cannot be avoided by router scheduling mechanisms alone, whether they are per-flow scheduling or the mechanisms described in this paper; the essential factor behind this form of congestion collapse is the absence of end-to-end feedback. Per-flow scheduling algorithms supply fairness with a cost of increased state, but provide no inherent incentive structure for best-effort flows to use end-to-end congestion control. Our approach, however, gives a low-overhead mechanism that also provides an incentive structure for flows to use end-to-end feedback.

The mechanisms described in this paper are suggested to help manage best-effort traffic only. We expect other traffic to use one of the "premium services" being added to the Internet. Examples of such premium services are the guaranteed and controlled-load services currently under development in the IETF. These services are primarily for real-time or other traffic with particular quality-of-service requirements, and require explicit admission control and preferential scheduling in the network. However, it seems likely that these premium services will apply only to a small fraction of future Internet traffic, and that the Internet will continue to be dominated by best-effort traffic.[1]

In this paper we propose router-based mechanisms for identifying high-bandwidth unresponsive flows in times of congestion, and for restricting their bandwidth usage. Section 2 discusses the problems of extreme unfairness and potential congestion collapse resulting from increasing levels of best-effort traffic not using end-to-end congestion control. Starting with queues managed by the RED queue management algorithm [FJ93], Section 3 describes the light-weight mechanism we use to identify high-bandwidth flows based on the history of packets dropped by RED. This mechanism does not require per-flow arrival state for every flow, and can run periodically in the background rather than the packet-forwarding path.

Next, Section 4 describes a range of mechanisms for determining which high-bandwidth flows should be *regulated* by having their bandwidth use restricted at the router. The most conservative such mechanism identifies high-bandwidth flows that are not "TCP-friendly" (i.e. using more bandwidth than would any conformant TCP implementation in the same circumstances). The second mechanism identifies high-bandwidth flows as "unresponsive" when they do not seem to be reducing their arrival rate at the router in response to increased packet drops. The third mechanism identifies high-bandwidth flows that may be both responsive and TCP-friendly, but nevertheless are using excessive bandwidth in a time of high congestion.

Section 5 discusses mechanisms in the router capable of restricting the bandwidth of those flows identified as requiring regulation. These mechanisms include either round-robin or priority scheduling. Simulations in Section 6 show how reclassifying unresponsive or high-bandwidth flows into a separate scheduling partition can effectively provide isolation from other best-effort traffic.

As mentioned above, a different approach would be the use of per-flow scheduling mechanisms such as variants of round-robin or fair queueing to isolate all best-effort flows at routers. Most of these per-flow scheduling mechanisms prevent a best-effort flow from using a disproportionate amount of bandwidth in times of congestion, and therefore require no further mechanisms to identify and restrict the bandwidth of particular best-effort flows. Section 7 compares the two approaches, and discusses some advantages of aggregating best-effort traffic in queues using simple FIFO scheduling and RED queue management along with the mechanisms described in this paper. Section 8 gives conclusions and discusses some of the open questions.

For all of the simulations presented, the ns simulator is available at [MF95], and the scripts to run these simulations will be made available from the Network Research Group web page [Gro97].

2 The problem of unresponsive flows

Unresponsive flows do not use end-to-end congestion control, and in particular do not reduce their load on the network when subjected to packet drops. Such behavior can result in two separate problems for the Internet. The first is a form of bandwidth starvation that unresponsive flows can inflict on well-behaved responsive traffic. The second is congestion collapse from a network busy transmitting packets that will simply be discarded before reaching their final destinations. We discuss these two dangers separately below.

[1] Another example of premium services includes mechanisms under investigation to use pricing along with mechanisms to indicate flows that should be differentially treated within the network.

2.1 Problems of unfairness

A first problem caused by the absence of end-to-end congestion control is drastic unfairness resulting from TCP flows competing with unresponsive UDP flows for scarce bandwidth. The TCP flows reduce their sending rates in response to congestion, leaving the uncooperative UDP flows to use the available bandwidth.

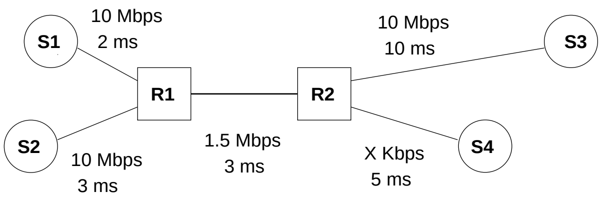

Figure 1: Simulation network.

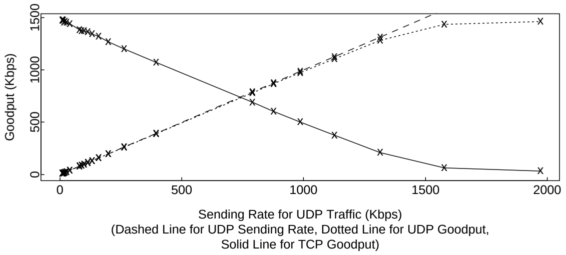

Figure 2: Simulation showing extreme unfairness.

The simulations in Figure 2 graphically illustrate what happens when UDP and TCP flows compete for bandwidth, given routers with FIFO scheduling. The simulation in Figure 2 uses the scenario in Figure 1, with the bandwidth of the R2-S4 link set to 10 Mbps. The traffic consists of several TCP connections from node S1 to node S3, each with unlimited data to send, and a single constant-rate UDP flow from node S2 to S4. The routers have a single output queue for each attached link, and use FIFO scheduling. The sending rate for the UDP flow ranges up to 2 Mbps.

Definition: *goodput*. We define the "goodput" of a flow as the bandwidth delivered to the receiver, excluding duplicate packets.

Each simulation is represented in Figure 3 by three marks, one for the UDP sending rate for that simulation, another for UDP goodput, and a third for TCP goodput. The dashed line shows the UDP sending rate for the entire simulation set, the dotted line shows the UDP goodput, and the solid line shows the TCP goodput, all as a function of the UDP sending rate.

As Figure 2 shows, when the sending rate of the UDP flow is small, the TCP flows have high goodput, and use almost all of the bandwidth on the R1-R2 link. When the sending rate of the UDP flow is larger, the UDP flow receives a correspondingly large fraction of the bandwidth on the R1-R2 link, while the TCP flows back off in response to packet drops. This unfairness results from responsive and unresponsive flows competing for bandwidth under FIFO scheduling. The UDP flow effectively "shuts out" the responsive TCP traffic.

Of course, even if all of the flows were using the exact same TCP congestion control mechanisms, with FIFO scheduling the bandwidth would not necessarily be distributed equally among those TCP flows with sufficient demand. As an example, [FJ92] discusses the relative distribution of bandwidth between two competing TCP connections with different roundtrip times. [Flo91] analyzes this difference, and goes on to discuss the relative distribution of bandwidth between two competing TCP connections on paths with different numbers of congested gateways. For example, [Flo91] shows how, as a result of TCP's congestion control algorithms, a connection's throughput varies as the inverse of the connection's roundtrip time. For paths with multiple congested gateways, [Flo91] further shows how a connection's throughput varies as the inverse of the square root of the number of congested gateways.

2.2 The danger of congestion collapse

This section discusses various forms of congestion collapse, and shows how unresponsive flows could contribute to congestion collapse in the Internet.

Informally, congestion collapse occurs when the network is at full or near full utilization but little useful work is being accomplished. Congestion collapse was first reported in the mid 1980s [Nag84], and was largely due to TCP connections unnecessarily retransmitting packets that were either in transit or had already been received at the receiver. Problems with the unnecessary retransmission of packets have generally been corrected by the timer improvements and congestion control mechanisms in modern implementations of TCP [Jac88].

A second form of congestion collapse has been discussed in [KM87] and [RF95], and consists of the network transmitting fragments or cells of packets that will be discarded at the receiver because they cannot be reassembled into a valid packet. This occurs when any of the cells or fragments of a network-layer packet are discarded (e.g. at the link layer), while the rest are delivered to the receiver, thus wasting bandwidth on a congested path. This danger of congestion collapse comes from a mismatch between link-level transmission units (e.g., cells or fragments) and higher-layer retransmission units (datagrams or packets), and can be prevented by mechanisms aimed at providing network-layer knowledge to the link-layer or vice-versa. One such mechanism is Early Packet Discard [RF95], which arranges that when an ATM switch drops cells, it will drop complete packets worth of cells. Another mechanism is Path MTU discovery [KMMP88], which helps to minimize packet fragmentation.

A variant of the second form of congestion collapse concerns the network transmitting packets received correctly by the transport-level at the end node, but subsequently discarded by the end-node before they can be of use of the end user [Var96]. This can occur when web users abort half-completed TCP connections because of delays in the network and then re-request the same data, or when TCP connections are automatically reset because the TCP data sender has failed to successfully retransmit a packet in a specified time. This form of congestion collapse could result from a persistent high packet drop rate in the network, and could be ameliorated by arranging for end-nodes to not discard data from half-completed connections.

A third form of congestion collapse, and the form of primary interest to us in this paper, arises when bandwidth is wasted by delivering packets through the network that are dropped before reaching their ultimate destination. We believe this is the largest unresolved danger with respect to congestion collapse in the Internet today. It is due primarily to the increasing deployment of open-loop applications not using end-to-end congestion control. Even more destructive would be best-effort applications that *increased* their sending rate in response to an increased packet drop rate (e.g., using an increased level of FEC).

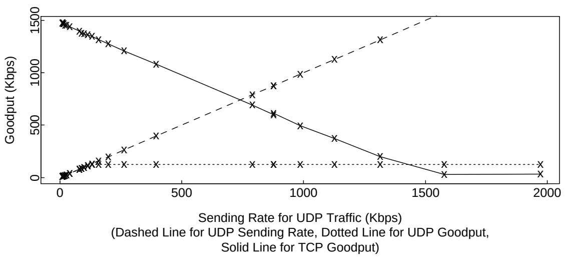

Figure 3: Simulation showing congestion collapse.

Figure 3 illustrates the third form of congestion collapse, caused when bandwidth is wasted by packets that never reach their destination. The simulation in Figure 3 uses the scenario in Figure 1, with the bandwidth of the R2-S4 link set to 128 Kbps. Because the final link in the path for the UDP traffic (R2-S4) is of smaller bandwidth compared to the others, most of the UDP packets will be dropped at R2, at the output port to the R2-S4 link, when the UDP source rate exceeds 128 Kbps.

As illustrated in Figure 3, as the UDP source rate increases linearly, the TCP goodput *decreases* linearly, and the UDP goodput is nearly constant. Thus, as the UDP flow increases its offered load, its only effect is to hurt the TCP (and aggregate) goodput. On the R1-R2 link, the UDP flow ultimately “wastes” the bandwidth that could have been used by the TCP flow, and reduces the goodput in the network as a whole down to a small fraction of the bandwidth of the R1-R2 link.

A different traffic mix could be used to illustrate congestion collapse for a network with routers with per-flow Fair Queueing or Round Robin scheduling. For this traffic mix, let the sending rate for a single UDP flow be fixed. As the *number* of UDP flows is increased, the fraction of the bandwidth in the R1-R2 link used by the UDP traffic would also increase, while the aggregate goodput of the UDP flows would still be limited at 128 Kbps. The limiting case of a very large number of very small bandwidth non-congestion-controlled flows in a highly-congested Internet would threaten congestion collapse regardless of the scheduling discipline at the router. It is difficult to envision incentive structures at the routers capable of discouraging such a situation.

The essential factor behind this form of congestion collapse is not the scheduling algorithm at the router, or the bandwidth used by a single UDP flow, but the absence of end-to-end congestion control for the UDP traffic. The congestion collapse would be essentially the same if the UDP traffic somewhat stupidly reserved and paid for more than 128 Kbps of bandwidth on the R1-R2 link in spite of the bandwidth limitations of the R2-S4 link. In a datagram network, end-to-end congestion control is needed to prevent flows from continuing to send when a large fraction of their packets are dropped in the network before reaching their destination. We note that it would not be possible to duplicate this scenario in a circuit-switched network where a sender is only allowed to send if there is an end-to-end path with the appropriate bandwidth.

3 Identifying high-bandwidth flows from the RED packet drop history

This section describes an efficient mechanism for a router to identify high-bandwidth flows in times of congestion, using the RED packet drop history. This mechanism does not require the router to keep per-flow state for each active flow. Keeping per-flow counters for packet arrivals for all active flows would be an unnecessary overhead for a router handling packets from a large number of very low bandwidth flows, many of which might never have a packet dropped. Instead, our identification mechanism detects high-bandwidth flows with a periodic pass in the background over information about the packets dropped at the router by the RED queue management.

The mechanism is independent of the granularity used to define a flow. One possibility would be for a router to define a flow by source and destination IP addresses. This would have the advantage of not being “fooled” by an application that breaks a single TCP connection into multiple connections to increase throughput.[2] Another possibility for defining the granularity of a flow would be to use source and destination IP addresses and port numbers to distinguish flows. The packet format for IPv6 provides a flow ID field routers could use to define some flows. Routers attached to high speed links in the interior of the Internet might use a coarser granularity to define a flow, rather than have each TCP connection belong to a separate flow.

The identification mechanism in this section assumes a router with RED queue management, and draws on the discussion in [Nai96] for identifying high-bandwidth flows from the RED packet drop history. RED queue management gives an efficient sampling mechanism and provides exactly the information needed for identifying high-bandwidth flows in times of congestion. For the remainder of this section, we distinguish between *forced* and *random* packet drops, and define both a *packet* and *byte drop metric*. We show the mechanism for identifying high-bandwidth flows should use the packet drop metric for random packet drops, and the byte drop metric for forced packet drops. The appendix shows that for queues with Drop-Tail queue management, the history of packet drops does not give sufficiently reliable information for identifying high-bandwidth flows.

Definitions: *forced* and *random packet drops*. We say a packet drop is *forced* if a packet is dropped because either the FIFO buffer overflowed, or the average queue size maintained by RED exceeded the RED maximum threshold parameter *maxthresh*. Otherwise a packet drop is called *random*. Random packet drops are expected to represent the majority of all packet drops for a properly-configured RED gateway, and result from RED's probabilistic sampling of the arriving packet stream.

When the average queue size exceeds some minimum threshold, indicating incipient congestion, RED queue management uses a random sampling method to choose which arriving packets to drop. [FJ93] describes two variants of the RED algorithm. In *packet mode*, for a given average queue size, each arriving packet has the same probability of being dropped regardless of the packet size in bytes. In *byte mode*, the probability a packet is dropped takes into account its size in bytes. For the remainder of this paper, we assume RED queue management operates in byte mode. RED in *packet mode* is preferable for routers limited by the number of *packets* arriving from each flow, rather than the number of *bytes*; RED in *packet mode* would give flows an incentive to use larger packets.

RED in byte mode is designed so that a flow's fraction of the aggregate random packet drops roughly equals its fraction of the aggregate arrival rate in bytes per second.[3] The reasoning for the design of byte-mode RED comes from the operation of TCP congestion avoidance. TCP assumes a single packet drop indicates congestion to the end nodes, regardless of the *number* of bytes lost in any dropped packet. Thus for random packet drops, the goal of RED queue management in byte-mode is to have each flow's fraction of the congestion indications correspond to its fraction of the arriving traffic in bytes per second. Section X of [FJ93] gives a statistical result showing that given a fixed average queue size, n packet drops, and a flow with a fraction b of the arriving bandwidth in bytes per second, the probability a flow receives more than twice its share of packet drops is at most $1/(e^{2nb^2})$. This is illustrated quantitatively in [FJ93] for $n = 100$. Informally, the result implies a high-bandwidth flow is very unlikely to receive more than twice its "share" of packet drops, and therefore the RED packet drop history should give an effective aid in estimating the arrival rate of high-bandwidth flows.

Definition: the *packet drop metric*. We define the *packet drop metric* for a flow as the ratio of the number of packets dropped from that flow over the total number of dropped packets. For random packet drops for RED in byte mode, the packet drop metric estimates a flow's fraction of the aggregate arrival rate in bytes per second (Bps). For random packet drops for RED in packet mode, the packet drop metric instead estimates a flow's fraction of the aggregate arrival rate in packets per second.

We used extensive simulations to verify that the per-flow packet drop metric for random packet drops is a good estimator of a flow's arrival rate in times of congestion. Figure 4 shows the results of a simple simulation in a topology similar to that in Figure 1, but with the link R1-R2 assigned a propagation delay of 20 ms, and link R2-S3 assigned a propagation delay of 4 ms, and the bandwidth of the R2-S4 link set to 10 Mbps. For all simulations in this paper, the RED queue management is configured with a minimum threshold of five packets, a maximum threshold of 20 packets, and a packet drop rate approaching 10% as the average queue size approaches the maximum threshold.[4] The FIFO buffer size in router R1 for the queue for the congested link R1-R2 is set to 100 packets; packets are rarely dropped due to buffer

[2] Breaking a single TCP connection into multiple connections at the application level to increase throughput would be one example of a possible spiral of increasingly-aggressive TCP congestion control. For a TCP connection that has been separated into N different TCP subconnections, a single packet drop results in one of the N subconnections, receiving $1/N$-th of the aggregate bandwidth, having its throughput cut in half. Thus, a single packet drop causes the aggregate bandwidth to be dropped to a fraction $(2N-1)/(2N)$ of its previous value. Then, because each TCP subconnection continues to increase its congestion window by one packet per RTT for those TCP subconnections that have not yet reached the receiver's advertised window, the aggregate TCP connections together increase their bandwidth by up to N packets per RTT. This is a much more aggressive congestion control algorithm that would lead to a correspondingly-larger steady-state packet drop rate in the Internet.

[3] This assumes the packet drop rate is sufficiently low relative to the packet size that, in byte mode, it is unlikely that two "bytes" in the same packet will be marked to be dropped.

[4] This is a change from the upper bound on the packet drop rate used for simulations in [FJ93]. This change is better suited for routers that typically have high levels of congestion.

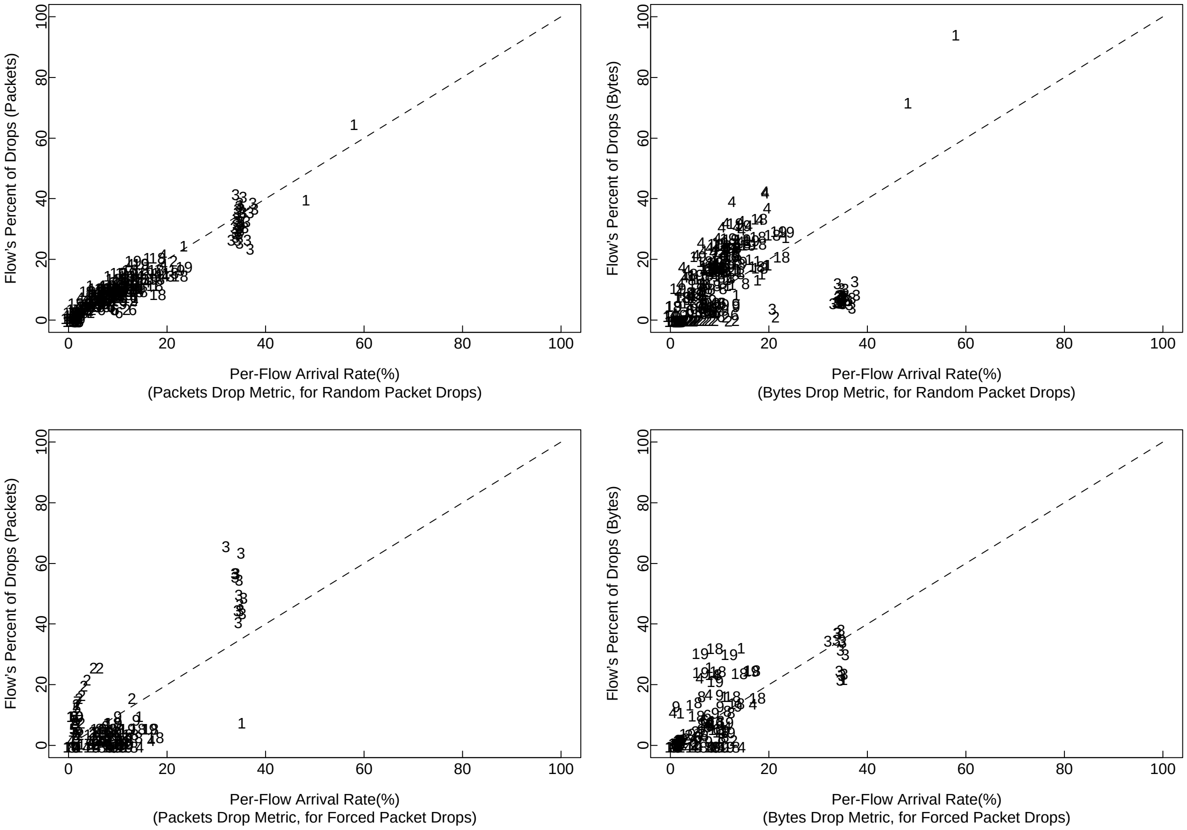

Figure 4: Comparing drop metrics for forced and random packet drops.

overflow.

The simulation includes a range of two-way traffic, including both constant-bit-rate (CBR) UDP flows and TCP connections. The TCP connections have a range of start times, packet sizes (from 512 to 2000 bytes), receiver's advertised windows, and round-trip times. Of particular interest are the high-bandwidth flows. Flow 3 is a CBR flow with 190 byte packets and an arrival rate of 64 KBps, about one-third of the link bandwidth. Flow 4 is a TCP flow whose high bandwidth is due to its larger packet size of 2000 bytes; most of the TCP flows in the simulation use 512-byte packets. To include a significant number of forced packet drops in the simulation, TCP connections “18” and “19” start up at roughly the same time, with large packets (1500 bytes) and large receiver's advertised windows, after 50 seconds of the 300-second simulation. More details of the simulation scenario will be given in the scripts that will be made available shortly.

The graph in the upper left corner of Figure 4 shows the packet drop metric for the random packet drops in the simulation. For every 100 random packet drops, there is a mark in the graph for every flow experiencing at least one packet drop. For each flow i, the number i is plotted on the graph, with the x-axis giving i's fraction of the aggregate arrival rate in Bps over the 100-drop time interval, and the y-axis giving i's fraction of the 100 packet drops.

If a flow's packet drop metric for random packet drops was an exact indication of the flow's arrival rate in Bps, then all of the marks in the left-hand graph would lie precisely on the diagonal line. A mark in the upper left quadrant of the graph indicates a flow with a large fraction of the dropped packets, but only a small fraction of the arriving packets. As the graph shows, the packet drop metric for the random packet drops does indeed give a reliable identification of the high-bandwidth flows.

Unlike random packet drops, with forced packet drops the RED algorithm does not get to “choose” whether or not to drop a packet. When the buffer is full, or when the average queue size exceeds the maximum threshold, RED drops *all* arriving packets until conditions change (the buffer is not longer full, or the average queue size no longer exceeds the threshold). Thus, a flow with one large packet arriving during a forced-drop time interval will have its packet dropped, and a flow with several small packets arriving during this interval will instead have all of its small packets dropped.

The graph in the bottom left corner of Figure 4 shows the packet drop metric for forced packet drops. As Figure 4 shows, the packet drop metric with forced packet drops has a systematic bias overestimating the arrival rate for flows with small packets such as Flow 3 and underestimating the arrival rate for flows with larger packets such as Flow 4.

Definition: the *byte drop metric*. The byte drop metric is defined as the ratio of the number of bytes dropped from a flow to the total number of bytes dropped. For forced packet drops, this metric gives the best estimate of a flow's arrival rate, as shown in the lower right graph of Figure 4. The upper right graph of Figure 4 shows that the byte drop metric is not adequate for random packet drops, because it overestimates the arrival rate of flows with larger packets and underestimates the arrival rate of flows with smaller packets.

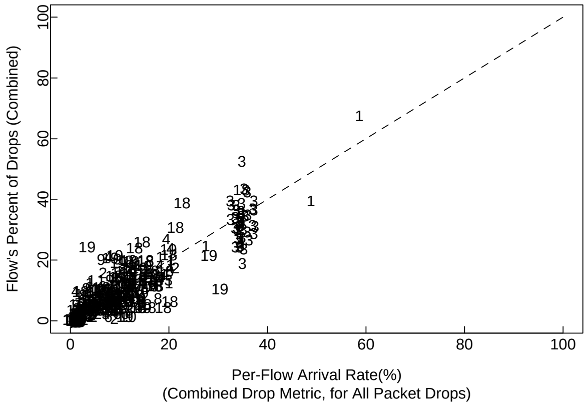

Figure 5: The combined drop metric for all packet drops.

Definition: the *combined drop metric*. By weighting a flow's byte and packet drop metrics by the ratio of forced and random packet drops, we can better estimate a flow's behavior than by using either metric alone. We define the *combined drop metric* for forced and random packet drops as follows:

$$\mathrm{M}_{\mathrm{Forced}} * \mathrm{f}_{\mathrm{Forced}} + \mathrm{M}_{\mathrm{Random}} * \mathrm{f}_{\mathrm{Random}},$$

where $\mathrm{M}_{\mathrm{Forced}}$ is the flow's byte drop metric for the forced packet drops, $\mathrm{M}_{\mathrm{Random}}$ is the flow's packet drop metric for the random packet drops, and $\mathrm{f}_{\mathrm{Forced}}$ and $\mathrm{f}_{\mathrm{Random}}$ are the fraction of the total packet drops from that queue that are forced and random, respectively.

Figure 5 shows the combined drop metric for each flow for the simulation in Figure 4, calculated for every 100 packet drops. As Figure 5 shows, the combined drop metric is a reasonably accurate indicator of the arrival rate for high-bandwidth flows.

A router particularly concerned about accurately estimating the arrival rate of high-bandwidth flows could monitor the arrival rate and drop rate for each flow identified by the packet drop history as likely to be high-bandwidth. Instead of estimating a flow's arrival rate from drop metrics, the router could make accurate measurements of the number of packet arrivals and drops for each candidate high-bandwidth flow over that time interval. Our simulations to date do not show any significant advantages to collecting this optional arrival information. Collecting this additional arrival information would not imply the router keeps per-flow statistics for all active flows; it would only collect per-flow arrival statistics for the particular high-bandwidth flows it is monitoring.

For a queue in units of packets, the appendix shows that packets dropped because of buffer overflow should be counted as forced packet drops. In contrast, for a queue in units of bytes, packets dropped because of buffer overflow should be counted as if they were random packet drops. The appendix also shows that routers with Drop-Tail queue management cannot use the packet drop history to reliably identify high bandwidth flows.

4 Identifying flows to regulate

The previous section shows how a router can examine the history of packet drops to help identify high-bandwidth flows. In this section, we discuss the range of policies a router might use to decide which flows to regulate. The router only needs to consider regulating those best-effort flows using significantly more than their “share” of the bandwidth in the presense of suppressed demand (as evidenced by packet drops) from other best-effort flows. A router can “regulate” a flow's bandwidth by using available scheduling mechanisms. When congestion is mild (as represented by a low packet drop rate), a router does not need to take any steps to identify high-bandwidth flows or further check if those flows need to be regulated.

The policies outlined in this section for identifying which high-bandwidth flows should be regulated range in the degree of caution. The most conservative policy is to only regulate high-bandwidth flows in times of congestion when they are determined to be either unresponsive to congestion or exceeding the bandwidth used by any conformant TCP flow under the same circumstances. A less cautious policy would include regulating any high-bandwidth flow using significantly more than its “share” of the bandwidth in a time of high congestion.

The tests in this section assume a “flow” is defined on the granularity of source and destination IP addresses and port numbers, so each TCP connection is a single flow. For a router in the interior of the network where a different granularity is used to define a flow, it will also be necessary to use different policies to identify a “flow” whose bandwidth should be regulated.

In our simulations the RED packet drop history is examined after 100 packet drops or five seconds, whichever is longer. From the set of packet drops, a flow is identified as the high-bandwidth flow if it has the highest combined drop metric. The router applies a set of tests to determines if the selected flow is unresponsive, not TCP-friendly, or "very-high-bandwidth". If the flow meets the criteria for any of these tests, the bandwidth of the flow is regulated by the router. Section 5 discusses the scheduling or packet drop mechanisms a router could use to restrict the bandwidth of regulated flows.

4.1 Identifying flows that are not TCP-friendly

Definition: *TCP-friendly flows.* We say that a flow is *TCP-friendly* if its arrival rate does not exceed that of any TCP connection in the same circumstances. The most conservative and clearly-defined test to deploy is to restrict the bandwidth of a high-bandwidth flow only when the router has high confidence it is not TCP-friendly. The test of whether or not a flow is TCP-friendly assumes TCP can be characterized by a congestion response of reducing its congestion window at least by half upon indications of congestion (i.e., packet drops), and of increasing its congestion window by a constant rate of at most one packet per roundtrip time otherwise. This response to congestion leads to a maximum overall sending rate for a TCP connection with a given packet loss rate, packet size, and roundtrip time. Given a packet drop rate of p, the maximum sending rate for a TCP connection is T Bps, for

$$T \leq \frac{1.5\sqrt{2/3} * B}{R * \sqrt{p}}, \tag{1}$$

for a TCP connection sending packets of B bytes, with a fairly constant roundtrip time, including queueing delays, of R seconds. This equation is derived in the appendix.

To apply this test, the router must be configured with a minimum roundtrip time R for flows and a maximum packet size B in bytes. In the most conservative case, R should be set to twice the one-way propagation delay of the attached link. The router can use its measurement of the aggregate packet drop rate for that queue over the recent time interval to estimate p, the packet drop rate experienced by a particular flow. Given the packet drop rate p, the minimum roundtrip time R, and the maximum packet size B, a router can use equation (1) to easily calculate the maximum arrival rate from a conformant TCP connection. TCP connections will generally use less than this maximum bandwidth, because they have limited demand, a longer roundtrip time, a window size limitation, a smaller packet size, a less-aggressive TCP implementation, a receiver that sends delayed ACKs, or additional packet drops from elsewhere in the network.

Given R and B, equation (1) reduces to a simple table at the router: if the steady-state packet drop rate is "*x*", then the arrival rate of an individual flow should be at most "*y*". The arrival rate of a high-bandwidth flow can be estimated from the flow's combined drop metric. If a flow's drop rate (the ratio of a flow's dropped packets to its arriving packets) is lower than the aggregate drop rate for the queue, the router will overestimate the flow's actual drop rate, but at the same time will underestimate the flow's arrival rate in Bps. These effects tend to cancel, implying the estimates should not lead to problems with incorrect identification of unresponsive or unfriendly flows. This is confirmed by our simulations to date.

The test of TCP-friendliness does not attempt to verify that a flow responds to each and every packet drop exactly as would a conformant TCP flow. It does however assume a flow should not use more bandwidth than would the most aggressive conformant TCP implementation in the same circumstances. The TCP protocol itself is subject to change, and the congestion control mechanisms used to derive equation 1 could at some point be changed by the IETF (Internet Engineering Task Force), the responsible standards body. Nevertheless, the two limitations on TCP's window increase and decrease algorithms have been followed by all conformant TCP implementations since 1988 [Jac88], and have an installed base in the end-systems of the Internet that will persist for some time, even if at some point in the future changes might be proposed to the TCP standards to allow more aggressive responses to congestion. As long as best-effort traffic is dominated by such an installed base of TCP traffic, it would be reasonable for routers to restrict the bandwidth of any flow with an arrival rate higher than that of any conformant TCP implementation in the same circumstances.

Care should be taken to only apply the TCP-friendly test to measurements taken over a sufficiently large time interval. The time period should not correspond to only one or two flow round-trip times. If the interval represents a small number of roundtrip times, then the flow might not have time to respond to the packet drops during that cycle until one roundtrip time later (i.e. in the subsequent cycle). If a very long round-trip time flow is incorrectly identified as not TCP-friendly because of a short measurement interval relative to its roundtrip time, then the router will notice the flow's delayed response to congestion a short time later, and can remove the bandwidth restrictions then.

Another consideration in applying equation (1) is the prevalence of forced packet drops. If the router is experiencing a large number of forced packet drops, a flow identified by the equation could be experiencing clusters of packet drops, with each cluster of packet drops representing a single indication of congestion to that flow. This is likely only to be a problem if the level of congestion is high; otherwise, RED gateways are characterized by random packet drops, with few instances of multiple packets dropped from a single window of data.

This test does not attempt to detect all flows which are not TCP-friendly. For example, the router might know a lower bound on the flow's roundtrip time, but the router does not know any flow's actual round-trip time. For routers with attached links with large propagation delays, the TCP-friendly test of equation (1) gives a useful tool for identifying flows which are not TCP-friendly. For routers with attached links of smaller propagation delay, the TCP-friendly test of equation (1) is less likely to identify any unfriendly flows. Such routers cannot exclude the possibility that a conformant TCP flow could receive a disproportionate share of the link bandwidth simply because it has a significantly smaller roundtrip time than competing TCP flows.

Flows whose arrival rates significantly exceed the maximum TCP-friendly arrival rate either are not using TCP-friendly congestion control, or have larger packets or a smaller round-trip time than assumed by the router. (Flows with particularly large packets could be observed at the router, but there is no simple test for a router to determine the end-to-end round-trip time of an active connection.) The router can freely restrict the bandwidth of best-effort flows determined not to be TCP-friendly in times of congestion. Such flows are "stealing" bandwidth from TCP-friendly traffic. Any such flow should only have its bandwidth restriction removed when there is no longer any significant link congestion, or when it has shown to reduce its arrival rate appropriately in response to congestion.

Definition: the *TCP-friendly test*. In our simulations, a high-bandwidth best-effort flow is restricted as not TCP-friendly if its estimated arrival rate is greater than $1.45B/(R\sqrt{p})$, for $B = 1460$ bytes, R twice the propagation delay of the attached link, and p the aggregate packet drop rate for that queue. A flow's restriction is removed if its arrival rate returns to less than $1.22B/(R\sqrt{p})$, for the new packet drop rate p.

4.2 Identifying unresponsive flows

The TCP-friendly test is based on the specific congestion control responses of TCP, and many routers may not want to use such a "TCP-centric" measure. The TCP-friendly test is also of limited usefulness for routers unable to assume strong bounds on TCP packet sizes and round-trip times. A more general test would be simply to verify a high-bandwidth flow was *responsive* (i.e. its arrival rate should decrease in response to an increased packet drop rate).

Equation (1) shows that for a TCP flow with persistent demand, if the long-term packet drop rate of the connection increases by a factor of x, then the arrival rate from the source should decrease by a factor of at least $\sqrt{x}$. For example, if the long term packet drop rate increases by a factor of four, than the arrival rate should decrease at least by a factor of two. This suggests a test for identifying unresponsive flows if the drop rate is changing. If the steady state drop rate increases by a factor x, and the presented load for a high-bandwidth flow does not decrease by a factor reasonably close to $\sqrt{x}$ or more, then the flow can be deemed not to be using congestion control (unresponsive).

Applying this test requires estimates of a flow's arrival rate and packet drop rate over several long time intervals. As in the previous section, the arrival rate can be estimated using the flow's drop metric, and the flow's packet drop rate can be estimated using the aggregate packet drop rate at the queue.

This test does not attempt to detect all flows that are not responding to congestion, but is only applied to the high bandwidth flows. When the packet drop rate remains relatively constant, no flows will be identified as unresponsive. In addition, the router has limited information about the flow's responses to congestion. The primary congestion indications experienced by a flow might be coming from elsewhere in the network. In addition, the arrival rate seen by a router is a result not only of the sending rate, but also of the drop rate experienced by a flow at a congested link earlier on its path.

As discussed in the previous section, care should be taken when applying this test. In particular, a test for unresponsiveness is less straightforward for a flow with a variable demand. In addition to possible end-to-end congestion mechanisms such as senders adjusting their coding rates or receivers subscribing and unsubscribing from layered multicast groups, the original data source itself could be ON/OFF or otherwise have strong rate variations over time. If a high-bandwidth flow is restricted because it has been identified as unresponsive, and it is later determined to be responding to congestion by reducing its arrival rate, then the restriction is removed

An additional refinement of this "responsiveness" test would be to distinguish three separate subcases: flows with an increasing or relatively constant average arrival rate (as indicated by the drop metric) in the face of an increasing packet drop rate at the router; a flow whose average arrival rate generally tracks longer-term changes in the packet drop rate at the router; and a flow whose average arrival rate seems to change independently of changes in the router's packet drop rate.

The router can freely restrict the bandwidth of best-effort flows determined to be unresponsive in times of congestion. Such flows are "stealing" bandwidth from responsive TCP-friendly traffic.

Definition: the *test for unresponsiveness*. In our simulations, a high-bandwidth best-effort flow is restricted as unresponsive if the packet drop rate has increased by a factor of four, but the flow's arrival rate has not decreased to below 90% of its previous value. Restrictions are removed from an unresponsive flow only if, after an increased packet drop rate, its arrival rate returns to at most half of its arrival rate when it was restricted.

4.3 Identifying very-high-bandwidth flows

A third test some routers might use identifies flows using a disproportionate share of the bandwidth in times of high congestion. There are times when a router might want to restrict the bandwidth of such flows even if they are TCP-friendly. A "disproportionate share" of bandwidth can be consumed by a TCP flow under several circumstances: if there is only one TCP with sustained persistent demand, or only one TCP using large windows, or one TCP with a significantly smaller roundtrip time or larger packet size than other active TCPs.

Let n be the number of flows with packet drops in the recent time interval. The most straightforward test to check if a flow was using a disproportionate share of the bandwidth in times of congestion would be to test if the flow's fraction of the aggregate arrival rate was greater than some preconfigured fraction, or greater than some small constant times $1/n$, when the aggregate packet drop rate was greater than some preconfigured threshold deemed as an unacceptable level of congestion. Our test is a modification of this approach that, instead of relying on a preconfigured threshold for the packet drop rate, allows for greater skewedness in the distribution of best-effort bandwidth when packet drop rates are lower. The goal is to prevent flows from using a highly disproportionate share of the bandwidth only when there is likely to be "sufficient" demand from other best-effort flows.

The first component of the very-high-bandwidth test is to check if a flow is using a disproportionate share of the bandwidth. We define a flow as using a *disproportionate share* of the best-effort bandwidth if its fraction of the aggregate arrival rate is more than $\log(3n)/n$, for $\log$ is the natural logarithm. We chose this fraction because it is close to one (i.e., 0.9) for n equal to two, and grows slowly as a multiple of $1/n$.

The second component of our test takes into account the level of congestion itself, as reflected in the aggregate packet drop rate p. We define a flow as having a high arrival rate relative to the level of congestion if its arrival rate is greater than $c/\sqrt{p}$ Bps for some constant c. This definition is motivated by our characterization in the appendix of the relationship between the arrival rate and the packet drop rate for conformant TCP. For our simulations we set c to 12,000, which is close to $1.5\sqrt{2/3}B/R$ for $B = 512$ bytes and $R = 0.05$ seconds.

Gauging the level of unsatisfied demand using packet drops can be difficult. For a large round-trip time TCP flow with persistent demand, a single packet drop can represent a significant suppressed demand. For a short bursty web transfer, a single packet drop might not mean much in terms of unsatisfied demand. A conservative approach would be to limit the restriction of a high-bandwidth responsive flow so that over the long run, each such flow receives as much bandwidth as the highest-bandwidth unrestricted flow.

In restricting the bandwidth of a high-bandwidth flow that has not been identified as either unresponsive or not TCP-friendly, care should be taken not to "punish" it by restricting its bandwidth too severely. If the regulation is performed by reassigning the flow to a different scheduling partition at the same priority level, this could be done by classifing the regulated high-bandwidth-but-responsive flows in a separate partition, and controlling the bandwidth allocated to the partition. If the regulation is by reassigning the flow to a scheduling partition at a lower priority level, then one possibility is to have separate priority levels for unrestricted best-effort traffic, restricted very-high-bandwidth traffic, and restricted unresponsive or unfriendly traffic. Another possibility is to monitor the bandwidth achieved by a restricted very-high-bandwidth flow and periodically reclassify it back to the unregulated best-effort partition to avoid starvation.

Definition: the *very-high-bandwidth test*. Let p be the aggregate packet drop rate for the unrestricted best-effort traffic, and let n be the number of flows with packet drops in the most recent interval. In our simulations, a best-effort flow is restricted as very-high-bandwidth if the estimated arrival rate is greater than $12,000/\sqrt{p}$ and the arrival rate is also greater than a fraction $\log(3n)/n$ of the best-effort bandwidth. The restriction is removed when one of these conditions is no longer true.

5 Regulating high-bandwidth or unresponsive flows

Regulating the bandwidth of identified flows requires two mechanisms at the router: a classifier to identify arriving packets belonging to a regulated flow, and a scheduling mechanism to restrict the bandwidth of these identified flows. We discuss each of these two components below.

5.1 Classifiers

IP routers must inspect packet headers to "classify" arriving packets and determine their proper output ports. For regulating flows, the classifier must also be able to identify packets belonging to each regulated flow.

The per-flow state required for the identification of high-bandwidth or unresponsive flows comes from a background pass over information from the history of dropped packets, and therefore does not include flows that have not had packets dropped. The computation of the combined drop metric uses the number of random packet drops for a flow, along with the number of bytes of forced packet drops from that flow. In addition, the combined drop metric uses the aggregate measures of packets dropped and bytes dropped for all flows.

Optional per-flow arrival information could be kept for flows that have been identified as high-bandwidth and are pending reclassification. The optional arrival information would consist of the number of packet and byte arrivals for that flow. Our simulations to date do not show any significant advantages to collecting this optional arrival information.

5.2 Scheduling mechanisms

We assume that routers already have available some scheduling mechanism to restrict the bandwidth of a specified flow. In this section we discuss two possible scheduling mechanisms, priority scheduling, and weighted round-robin or weighted fair-queueing scheduling between two partitions at the same priority level.

Restricting the bandwidth of specified flows would be straightforward for routers with class-based queueing (CBQ [FJ95]) or similar scheduling mechanisms that allow the router to combine priority scheduling with bandwidth allocations. In this case, restricted flows could be assigned to a lower-priority class, while the scheduler could at the same time ensure that the lower-priority class always receives some minimum fraction of the link bandwidth. However, the router could also use the priority-scheduling or round-robin scheduling mechanisms that might be already available to it.

For the simulations in this paper the scheduler has two scheduling classes or partitions at the same priority level, and uses weighted round robin scheduling between the two partitions. One partition is the default class for best-effort traffic, and contains unrestricted best-effort traffic. The other partition contains restricted best-effort traffic, and packets for regulated flows are classified to this partition.

For a router with weighted round-robin or weighted fair queueing scheduling, the regulation mechanism also has to dynamically allocate the bandwidth between the two partitions. In our simulations, the restricted partition is never allocated more than 25% of the best-effort bandwidth.

Consider a restricted partition with n active flows. In our simulations, the bandwidth allocated to the restricted partition is changed each time a new flow is added to or removed from that partition. The partition's allocated bandwidth is never more than $n/2$ times the computed TCP-friendly bandwidth for the packet drop rate in the unrestricted best-effort queue, and never more than $n/2$ times the highest per-flow arrival rate for an unrestricted flow. Thus, congestion in the restricted partition should always be significantly greater than the level of congestion in the unrestricted partition. If the drop rate in the restricted partition is not at least twice the drop rate in the unrestricted partition, then the bandwidth allocation to the restricted partition is halved. Without this mechanism, a flow could actually benefit from being placed in the restricted partition by being isolated from the unrestricted best-effort traffic.

For routers with priority scheduling, flows positively identified as either unresponsive or as not TCP-friendly can simply be classified to a lower-priority scheduling partition. This should serve as a strong incentive for best-effort traffic to use end-to-end congestion control. Flows can be reclassified back to the higher-priority partition if it is determined they are decreasing their arrival rate in response to increases in the packet drop rate.

However, care should be taken to avoid starvation of flows that have simply been identified as very-high-bandwidth. One way to do this would be to use three priorities of best-effort traffic, with the highest priority for the unrestricted traffic, the second priority for the restricted very-high-bandwidth traffic, and the lowest priority for the unresponsive or unfriendly traffic. Another possibility would be to only reclassify the very-high-bandwidth traffic for short periods of time, to avoid starvation.

For routers lacking a suitable scheduling mechanism for regulating flows, identified flows could be regulated by having the queue management mechanism preferentially drop packets from those flows. We have not investigated this option [rdm97].

6 Simulations

We have run extensive simulations using our mechanisms for identifying and regulating high-bandwidth flows, and will report on them in more detail in a later paper. This section shows a simple simulation illustrating the mechanisms for identifying and regulating high-bandwidth flows.

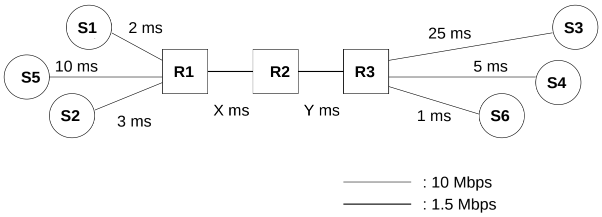

Figure 6: Simulation network.

The top graph of Figure 7 shows a simulation not employing the mechanisms for regulating unresponsive or high bandwidth flows. For each flow, there is a line in the graph showing the average bandwidth used by that flow in successive 20-second intervals. The simulation uses the scenario in Figure 6, with $X = 30$ ms and $Y = 0$ ms. The congested links between the routers use RED queue management. There is a range of two-way UDP and TCP traffic; the UDP traffic consists of constant-bit-rate flows, and the TCP connections use a range of TCP variants (Tahoe, Reno, SACK), packet sizes (from 512 bytes to 1500 bytes), start times, and demand (either an infinite or a limited amount of data to send). For the UDP flows, there is also a range of packet sizes and average arrival rates (from 2 to 66 KBps). The simulation is constructed so that the demand peaks at roughly 300 seconds, and then starts to decrease again. While it is not possible to extract too much detail from this graph, the UDP flows are recognizable as straight lines. For further details, the scripts for these simulations will be made available from our web page [Gro97].

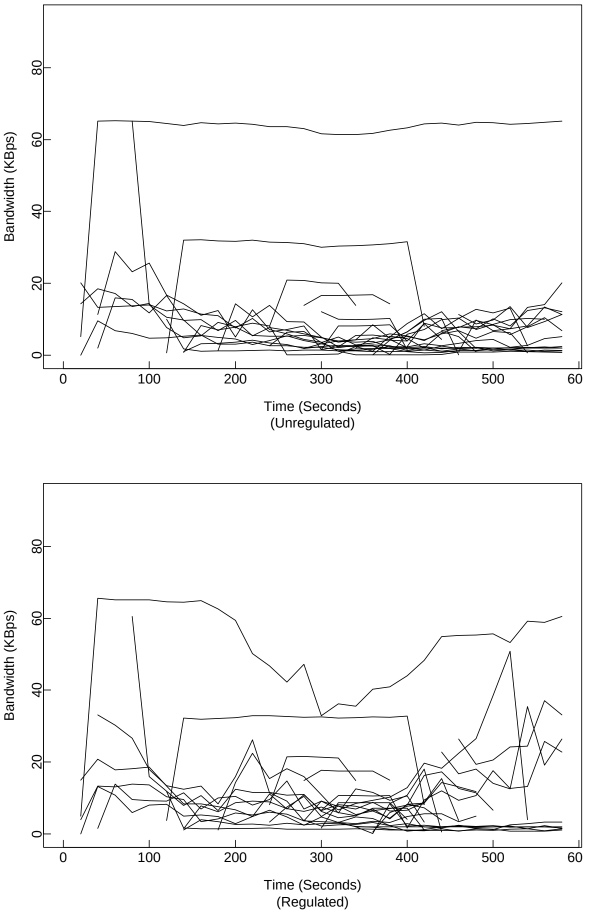

Figure 7: Simulations with and without regulation.

The bottom graph of Figure 7 shows the same simulation scenario with the mechanisms for regulating unresponsive or high bandwidth flows enabled. As the packet drop rate on R1-R2 increases to 2.5%, the high-bandwidth UDP flow is identified as unresponsive, and has its bandwidth restricted.

In our simulations, the router periodically examines the restricted flow that has the smallest combined drop metric to see if its restriction can be removed. In general, the restricted flow is only reclassified to the unrestricted partition if it has shown it is not likely to be immediately reclassified as needing restriction.

If the packet drop rate at the router stays relatively constant, then the router has limited information for distinguishing a responsive from an unresponsive flow. For example, it is possible to construct three different simulations with a steady-state packet drop rate of roughly 2%, where a single flow is receiving 80% of the link bandwidth, and twenty smaller flows share the remaining 20%. In this scenario the high bandwidth flow could be an unresponsive UDP flow, and the twenty smaller flows could be TCPs willing to use a much larger share of the link bandwidth. It is also possible to construct scenarios where the high-bandwidth flow is a TCP flow, with the twenty smaller flows either short TCP flows with limited demand, or low-bandwidth UDP flows. For this scenario, the router can determine if the high-bandwidth flow is truly unresponsive, and if the low-bandwidth flows truly have suppressed demand, by restricting the bandwidth of the high-bandwidth flow, and examining the change (or lack of change) in that flow's arrival rate.

We have not yet run simulations with realistic UDP traffic sources, or with multiple congested gateways.

7 A comparison with per-flow scheduling

One approach for constructing routers would be to use per-flow scheduling mechanisms for best-effort traffic, such as variants of round-robin or fair queueing scheduling. Most forms of these per-flow scheduling algorithms separately schedule the packets from each flow, dividing the available bandwidth among the various flows. With per-flow scheduling at the router, there would be no need for further mechanisms to identify and restrict the bandwidth of best-effort flows using a disproportionate amount of bandwidth in times of congestion. While this might seem like a compelling case for deploying per-flow scheduling for best-effort traffic, in this section we discuss some of the advantages in continuing to aggregate best-effort traffic in FIFO queues using FIFO scheduling.

First, FIFO scheduling is efficient to implement, a particularly important concern as links go to higher speeds while the number of best-effort flows active at one time increases. Many of the best-effort flows will have relatively short lifetimes, and will have little need for per-flow scheduling at the routers.

Second, completely apart from considerations of efficiency, FIFO scheduling is in many ways the optimal scheduling algorithm for a class of traffic where the long-term aggregate arrival rate is restricted by either admission controls or, in the case of best-effort traffic, by compatible end-to-end congestion control procedures. In comparison to Fair Queueing [DKS90] or Round Robin scheduling, FIFO scheduling reduces the tail of the delay distribution [CSZ92]. In particular, FIFO scheduling allows packets arriving in small bursts to be transmitted in a burst, rather than having the packets “spread out” and delayed by the scheduler.

More critical than issues of efficiency or delay distributions are the questions of whether the overall architecture in the routers encourages end-to-end congestion control for best-effort traffic. Recommendations for the ubiquitous deployment in routers of per-flow scheduling for best-effort traffic are based on an assumption that in a heterogeneous world, best-effort flows cannot be relied upon to be responsive to congestion, and therefore best-effort flows should be isolated from each other. Such an architecture might encourage individual best-effort flows to each "greedily" optimize their own use of network resources, but it does not necessarily encourage best-effort flows to use end-to-end congestion control. As we showed in Section 2, the absence of end-to-end congestion control could threaten congestion collapse for best-effort traffic. In contrast, FIFO scheduling for best-effort traffic, when coupled with mechanisms for restricting the bandwidth of high-bandwidth unresponsive flows in times of congestion, encourages and supports the continued use of end-to-end congestion control for best-effort traffic.

We note that routers with per-flow scheduling for best effort traffic still require queue management mechanisms; the queue management mechanism (or lack thereof) is to some extent orthogonal to the scheduling mechanism. Queue management mechanisms such as RED allow the router to control the average queue size and provide indications of incipient congestion to the end-nodes [BDF$^+$96]. A router with per-flow scheduling for best-effort traffic still needs queue management to prevent the unnecessary per-packet delay that can result from a persistent queue and to minimize the total number of packets dropped at the router.

A more speculative issue is whether min-max fairness is the ideal fairness metric to use for best-effort traffic at a specific router. It has the advantage of being simple to define at a router, and is indeed the basis for our approach in this paper for defining flows using a disproportionate share of the link bandwidth. However, instead of considering the network as a whole, the *min-max* definition of fairness restricts attention separately to each isolated component. A fairness metric that would recognize each flow's equal access to the scarce resources of the Internet would have to take into account such global factors as the number of congested links on each flow's path.

FIFO scheduling for an aggregation of best-effort traffic is completely compatible with differential scheduling at the router for link-sharing or for premium services. FIFO scheduling for a single queue can be combined with multiple queues at an output port, with a range of scheduling mechanisms among the multiple queues. One such mechanism is CBQ, where best-effort traffic can be aggregated in one or more queues, with other queues for real-time traffic or traffic with other requirements.

In a network engineered so that the typical case is one of sufficient bandwidth for the demand, distinctions between the various scheduling algorithms would become less important. Similarly, in such a network the possibility of congestion collapse due to congested links carrying packets that would later be dropped in the network would become more remote. It is hard to predict, however, when or if the scenario of sufficient bandwidth for the demand is likely to be achieved.

8 Conclusions and future work

We have demonstrated light-weight router mechanisms for detecting and restricting unresponsive or high-bandwidth best-effort flows in times of congestion. These mechanisms operate in conjunction with RED queue mangement and concretely support end-to-end congestion control for best-effort traffic.

Clearly there is more work still to be done in investigating the mechanisms outlined in this paper in a wide range of environments and with a wider range of traffic mixes. We have not yet outlined a specific proposal for implementing these mechanisms in routers with priority scheduling. We also intend to explore these mechanisms in more complex scenarios with multiple congested gateways, more realistic traffic models for UDP traffic, and higher-priority real-time traffic.

We believe the most important issue is not the precise functioning of the mechanisms to restrict the bandwidth of unresponsive best-effort flows, but simply that such mechanisms be deployed. Mechanisms such as these would go a long way to making concrete the essential role played by congestion control for best-effort traffic in the Internet.

9 Acknowledgments

This paper results in part from a long collaboration with Van Jacobson. It also results from a long history of discussions and disagreements in the IETF Transport Area Directorate, the Internet End-to-End Research Group, and elsewhere. We are particularly indebted to Greg Minshall and Lixia Zhang for feedback on this paper, and to Jean Bolot, Bob Braden, Jamshid Mahdavi, Matt Mathis, and Scott Shenker for discussions of some of these matters.

References

[BDF$^+$96] B. Braden, B. Davie, S. Floyd, G. Minshall, and S. Shenker. "Recommendations on Queue Management and Congestion Avoidance in the Internet,". 1996. unpublished manuscript.

[CSZ92] D.D. Clark, S. Shenker, and L. Zhang. "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism,". *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 14–26, 1992.

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

[FF95] K. Fall and S. Floyd. "Comparisons of Tahoe, Reno, and Sack TCP,". Technical report, 1995. URL http://www-nrg.ee.lbl.gov/nrg-papers.html.

[FJ92] S. Floyd and V. Jacobson. "On Traffic Phase Effects in Packet-Switched Gateways,". *Internetworking: Research and Experience*, 3(3):115–156, Sep. 1992.

[FJ93] S. Floyd and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance,". *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug. 1993. URL http://www-nrg.ee.lbl.gov/nrg-papers.html.

[FJ95] S. Floyd and V. Jacobson. "Link-sharing and Resource Management Models for Packet Networks,". *IEEE/ACM Transactions on Networking*, 3(4), 1995.

[Flo91] S. Floyd. "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic,". *ACM Computer Communication Review*, 21(5):30–47, Oct. 1991.

[Flo94] S. Floyd. "TCP and Explicit Congestion Notification,". *ACM Computer Communication Review*, 24(5):10–23, Oct. 1994.

[Gro97] Network Research Group. "LBNL Network Research Group Web Page,". Technical report, 1997. http://www-nrg.ee.lbl.gov/.

[Jac88] V. Jacobson. "Congestion Avoidance and Control,". *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 314–329, 1988. An updated version is available via ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[KM87] C. Kent and J. Mogul. "Fragmentation Considered Harmful,". *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 390–401, Aug. 1987.

[KMMP88] C. Kent, K. McCloghrie, J. Mogul, and C. Partridge. "IP MTU Discovery options,". Request for Comments RFC 1063, Internet Engineering Task Force, July 1988.

[MF95] S. McCanne and S. Floyd. "NS (Network Simulator),", 1995. URL http://www-nrg.ee.lbl.gov/ns.

[Nag84] J. Nagle. "Congestion control in IP/TCP internetworks,". Request for Comments RFC 896, Internet Engineering Task Force, January 1984.

[Nai96] T. Nairne. "Idenfitying High-Bandwidth Users in RED Gateways,". Technical report, Oct. 1996. UCLA Computer Science Department.

[OKM96] T. Ott, J. Kemperman, and M. Mathis. "The Stationary Distribution of Ideal TCP Congestion Avoidance,". Technical report, Aug. 1996.

[rdm97] rdm@tad.micro.umn.edu). "Re: congestion control mechanisms in realaudio,". Technical report, 9 Jan 1997. Message ¡19970109200431.29536.qmail@test.legislate.com¿ to the end2end-interest mailing list, archived at ftp://isi.edu/end2end/.

[RF95] A. Romanow and S. Floyd. "Dynamics of TCP Traffic over ATM Networks,". *IEEE Journal on Selected Areas in Communications*, 13(4), 1995. URL http://www-nrg.ee.lbl.gov/nrg-papers.html.

[She94] S. Shenker. "Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines,". *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 47–57, Aug. 1994.

[Var96] G. Varghese. "On Avoiding Congestion Collapse,". Technical report, Nov. 19 1996. viewgraphs, Washington University Workshop on the Integration of IP and ATM.

A Characterizing TCP-friendly flows

Since congestion control was introduced to TCP in 1988 [Jac88], TCP flows in the Internet have responded to congestion signals from the network (i.e. packet drops) by reducing their offered load by half for each window of data experiencing a packet drop. For a responsive flow with persistent demand, increasing the packet drop rate of a flow at a router should result in a decreased arrival rate from that flow at that router. The maximum arrival rate a router should see from any single conformant TCP connection can be characterized, given the steady-state packet drop rate at the router, an upper bound the TCP packet size, and a lower bound on the TCP connection's roundtrip time. Using this characterization, routers can identify any flows using more bandwidth than would any TCP flow in the same circumstances.

In this section we explore the relationship between throughput and the packet drop rate for a *conformant* TCP connection. By a *conformant* TCP connection, we mean a TCP connection where the TCP sender follows the following two essential components of today's TCP congestion control. First, the TCP data sender interprets any packet drop in a window of data as an indication of congestion, and responds by reducing the congestion window, and therefore the effective sending rate, at least in half. Second, during the congestion avoidance phase in the absence of congestion, the TCP sender increases the congestion window by at most one packet per roundtrip time (or more precisely, by at most one packet per window of data). These two components lead to a simple relationship between the "steady-state" packet drop rate received by a TCP connection, and the "steady-state" average throughput achieved by that connection.

Many conformant TCP implementations respond to congestion less aggressively than allowed by the limits of congestion control described above. TCP implementations have potentially-long delays due to retransmit timeouts; at times, TCP senders invoke slow-start in responding to congestion; they may be limited by maximum bounds on the window size, imposed by buffering or lack of window scaling at either at the sender or receiver; for TCP connections where the receiver only sends an ACK packet for every two data packets, the TCP sender increases the congestion window by less than one packet per roundtrip time.

We assume a *steady-state* model of TCP as introduced in Section 5 of [Flo91]. For the purposes of heuristic analysis, we assume a single packet is dropped from a TCP connection each time the congestion window is increased to W packets (and never when the congestion window is below W packets). The *steady-state* model assumes a non-zero average packet drop rate of p and independent packet drops. The TCP sender responds to a packet drop by cutting the congestion window at least in half. After a packet is dropped, the TCP sender increases its congestion window by at most one packet each roundtrip time, until the congestion window again reaches its old value of W packets (and, in steady state, the TCP connection receives another packet drop). The assumption in this model of a deterministic and repeatable pattern, although admittedly unrealistic, leads to results verified by simulations in this section and by an independently derived more rigorous analysis in [OKM96].

We consider a TCP connection sending packets (or more precisely, segments) of B bytes, with a fairly constant roundtrip time, including queueing delays, of R seconds. Each time a packet is dropped, the TCP sender has a congestion window of W packets.

By decreasing its window by at least half for each packet drop and increasing its window by at most one per round-trip time afterwards, the TCP sender transmits at least

$$\frac{W}{2} + \left(\frac{W}{2} + 1\right) + ... + W \approx \frac{3}{8}W^2. \tag{2}$$

packets for each packet dropped. The fraction of the sender's packets that are dropped is then bounded by the reciprocal of that value:

$$p \leq \frac{8}{3W^2}. \tag{3}$$

From equation (3),

$$W \leq \sqrt{\frac{8}{3p}}. \tag{4}$$

For our steady-state model assuming a link with steady-state packet drop rate p, equation (4) gives the maximum congestion window W of a TCP connection when a packet is dropped. With a steady-state packet drop rate of p in the steady-state model, the TCP connection sends $\frac{3}{8}W^2$ packets between packet drops. Because the congestion window is decreased by at least half, and increased by at most one packet per roundtrip time, there are at least $W/2$ roundtrip times between packet drops in the steady-state model. The maximum sending rate for a TCP connection over a single cycle of the steady-state model is thus T Bps, for

$$T \leq \frac{0.75 * W * B}{R}.$$

Substituting for W from equation (4), we get

$$T \leq \frac{1.5\sqrt{2/3} * B}{R * \sqrt{p}}. \tag{5}$$

This upper bound on a TCP's sending rate applies for any conformant TCP that decreases its congestion window by at least half, and, after the congestion window has been decreased by half, increases the congestion window by at most one packet per roundtrip time.[5] Thus, this upper bound also applies to a TCP restricted by the receiver's advertised window, or by TCP variants such as Vegas TCP which sometimes refrain from increasing the congestion window during the congestion avoidance phase. Assuming a steady-state packet drop rate of p, and thus in the steady-state model that the TCP connection gets to send $1/p$ packets between packet drops, clearly the TCP connection maximizes its average throughput by increasing its congestion window by the maximum allowed amount each roundtrip time.

This might at first seem counter-intuitive. However, the purposes of the steady-state model in this section are to explore the relationship between the steady-state packet drop rate and the steady-state arrival rate from the TCP connection. Certainly in a specific scenario with all else being equal, a TCP that refrains from increasing its congestion window from time to time might increase its own throughput by decreasing the aggregate packet drop rate. This does not change the fact that the inequality in equation 1 still describes the relationship between the packet drop rate and the arrival rate for that connection.

¿From the appendix, for TCP connections where the data receiver sends at most one ACK for every two packets, we show a stronger upper bound on the sending rate of

$$T \leq \frac{1.5\sqrt{1/3} * B}{R * \sqrt{p}}. \tag{6}$$

[5] The same result was derived by [OKM96], using a more rigorous model, with a constant of 1.3 instead of 1.22 ($\approx 1.5\sqrt{2/3}$).

Equation 5 does not take into account TCP delays due to waiting for retransmit timers to time out. Thus, equation (5) drastically overestimates the bandwidth for steady-state scenarios when the congestion window W is less than four packets when a packet is dropped. From equation (4), this occurs when the packet drop rate is 16% or higher. (If the congestion window is four or higher, the TCP connection can recover from a single packet drop using Fast Retransmit, after receiving several duplicate acknowledgements. If the congestion window is smaller, then the TCP connection generally has to wait for a retransmit timeout. [FF95]) For a packet drop rate of 100%, our steady-state model would assume that the TCP connection stubbornly sends one packet every roundtrip time, and equation (5) (because it used an approximation in equation (2)) gives a TCP sending rate of slightly over one packet per roundtrip time. Incorporating the notion of retransmit timer backoff in the model would give a much more realistic result.

Although the language in this paper refers only to packet drops, proposals have been made to add explicit congestion notification to TCP/IP [Flo94]. If explicit congestion notification were deployed, then instead of dropping a packet to provide feedback about congestion, a router could simply “mark” packets by setting the the Explicit Congestion Notification bit in packet headers.

A.1 Simulations verifying the “TCP-friendly” characterization

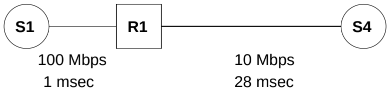

Figure 8: Simulation network.

Figure 8 illustrates the simulation topology used to evaluate the “TCP-friendly” characterization. The solid line in Figure 9 shows the TCP-friendly bandwidth from equation (5) as a function of the packet drop rate. Figure 9 assumes a TCP connection with minimum roundtrip time of $R = 0.06$ seconds and a maximum packet size of $B = 1460$ bytes. The x-axis shows p, the fraction of arriving packets that are dropped, and the y-axis shows T, the upper bound on TCP arrival rate in KBps.

Each dashed line in Figure 9 shows the results from a single simulation set. Each simulation consists of two competing connections, one TCP and the other UDP, from node S1 to node S4. For each simulation set the sending rate of the UDP flow ranges from zero up to the available bandwidth of the congested link. The router uses FIFO scheduling and RED queue management. The RED packet drop mechanisms are generally able to prevent both the FIFO buffer from overflowing and RED's average queue size from exceeding its maximum threshold. The TCP connection sees a roundtrip time, including queueing delay, of roughly 60 ms.

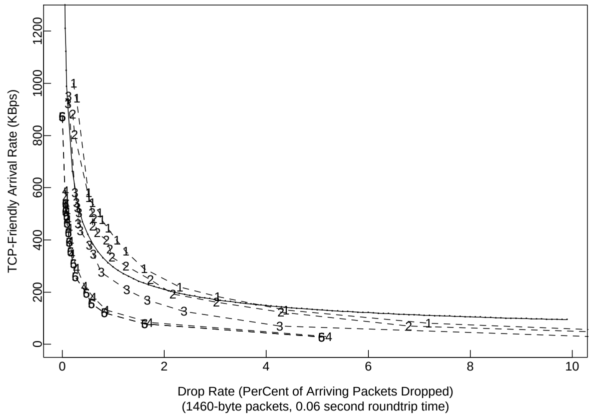

Figure 9: TCP-friendly bandwidth for a 60-ms roundtrip time and 1460-byte packets.

Each simulation is represented by a number in Figure 9. The simulations in a simulation set differ from each other only in the sending rate of the UDP flow. Numbers “1” through “3” show simulations where the TCP connection uses 1460-byte packets. Numbers “4” through “6” show simulations with 512-byte packets. Simulation sets “2” and “5” use Tahoe TCP, and the others use SACK TCP. Simulation sets “3” and “6” use data receivers with delayed ACKs (sending one ACK to acknowledge two data packets), and the others use single ACKS (sending an ACK for every data packet). For all of the simulations, the TCP clock granularity is 100 ms. The x-axis in Figure 9 shows the TCP connection's sending rate and the y-axis shows the fraction of its packets which are dropped.

For the SACK and Tahoe simulations with 1460-byte packets and single-ACK receivers (simulation sets “1” and “2”), the simulation results are a reasonable match to the computed TCP-friendly bandwidth. For drop rates lower than 2%, the SACK and Tahoe TCPs receive more than the computed TCP-friendly bandwidth. One factor contributing to the discrepancy is that at these drop rates, the average queue size is low, and the roundtrip time is closer to 58 than to 60 ms. A second factor is that the steady-state model makes the unrealistic assumption that packets are always dropped when TCP's congestion window is exactly W packets. From [OKM96], a more realistic model of packet drops raises the constant in equation (5) from 1.22 (that is, $1.5\sqrt{2/3}$) to 1.3. Our simulations would confirm that for lower packet drop rates, 1.3 is the more realistic value.

For packet drop rates greater than 5%, Figure 9 shows that the TCP-friendly bandwidth greatly overestimates the arrival rate of a TCP connection. As mentioned earlier, this is because the current version of the steady-state model does not take into account delays due to retransmit timers.

Simulations with 512-byte packets closely match (equation 5) using 512-byte packets. As seen in Figure 9, the more aggressive the TCP congestion control (i.e. a TCP with 1460-byte packets is more aggressive than TCP with 512-byte packets), the higher the steady-state packet drop rate needed to sustain the same per-connection bandwidth. A spiral of increasingly-aggressive congestion control would lead to a matching spiral of an increasingly-high steady-state packet drop rate, in the context of a fixed available bandwidth.

A.2 TCP with delayed acks

For TCP connections where the data receiver sends at most one ACK for every two packets, we could show a stronger upper bound on the sending rate of

$$T \le \frac{1.5\sqrt{1/3} * B}{R * \sqrt{p}}. \tag{7}$$

For a TCP connection with a delayed-ACK sink, the sender receives one acknowledgement for every two packets, and increases its window more slowly that a TCP connection that receives an ACK for every packet. With a delayed-ACK sink, the fraction of that connection's arriving packets that are dropped is

$$p = \frac{1}{\sum_{i=0}^{W}(W/2 + i/2)} \approx \frac{1}{(3/4)W^2}. \tag{8}$$

This gives a bandwidth of

$$T = \frac{1.5\sqrt{1/3} * B}{R * \sqrt{p}}. \tag{9}$$

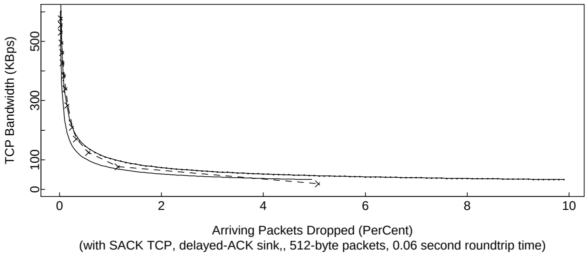

Figure 10: TCP bandwidth vs. steady-state drop rate, for SACK TCP with a delayed-ACK sink, a 60-ms roundtrip time and 512-byte packets.

Figure 10 shows the results for SACK TCP with a delayed-ACK sink with the simulated topology of figure 8. For a fixed throughput, a TCP connection with a delayed-ACK sink should receive half the packet drop rate of a TCP connection that receives an ACK for every packet. The top solid line shows the analytical results for an immediate-ACK sink, and the bottom solid line shows the analytical results for an delayed-ACK sink. For a given packet drop rate, a TCP connection with a delayed-ACK sink will receive less throughput than a TCP connection with an immediate-ACK sink.

B Identifying high-bandwidth flows for queues with drop-tail queue management

Figure 11 shows the results from a simulation that differs from the simulation in Figure 4 only in that the router uses Drop-Tail rather than RED queue management. With Drop-Tail queue management, the router only drops arriving packets when the buffer overflows. For the simulation in Figure 11, the buffer is measured in packets, with a buffer size of 25 packets. That is, the buffer can store exactly 25 packets, regardless of the size of each packet in bytes.

The two graphs in Figure 11 compare the packets and the bytes drop metric. Figure 11 shows that for a Drop-Tail queue measured in packets, the bytes drop metric is much better than the packets drop metric in indicating a flow's arrival rate in Bps. This is because a Drop-Tail queue with a queue measured in packets drops proportionately more packets from small-packet flows than from large-packet flows with the same arrival rate in Bps. However, Figure 11 also shows that for a Drop-Tail queue measured in packets, neither drop metric gives a very reliable indication of a flow's arrival rate in bytes per second. The appendix shows that when the Drop-Tail buffer is measured in bytes rather than packets, the packets drop metric gives a plausible indication of a flow's arrival rate in Bps.

Figure 12 shows a simulation where the Drop-Tail buffer is measured in bytes, with a buffer size of 12.5 KB. In this case, an almost-full buffer might give room for a small packet but not for a larger one. As Figure 12 shows, in this case the packets drop metric gives a plausible indication of a flow's arrival rate in Bps.

We should note that with very heavy congestion and unresponsive flows, even a RED queue will no longer be dropping all packets probabilistically, but will be forced to drop many arriving packets either because of buffer overflow (for a queue with a small buffer relative to the maximum threshold for the average queue size), or because the average queue size is too high (for queues with larger buffers). As the packet drop rate increases, the computational overhead of monitoring the dropped packets approaches the computational overhead of monitoring the packet arrivals directly. However, the identification and regulation of high-bandwidth unresponsive flows should be sufficient in many

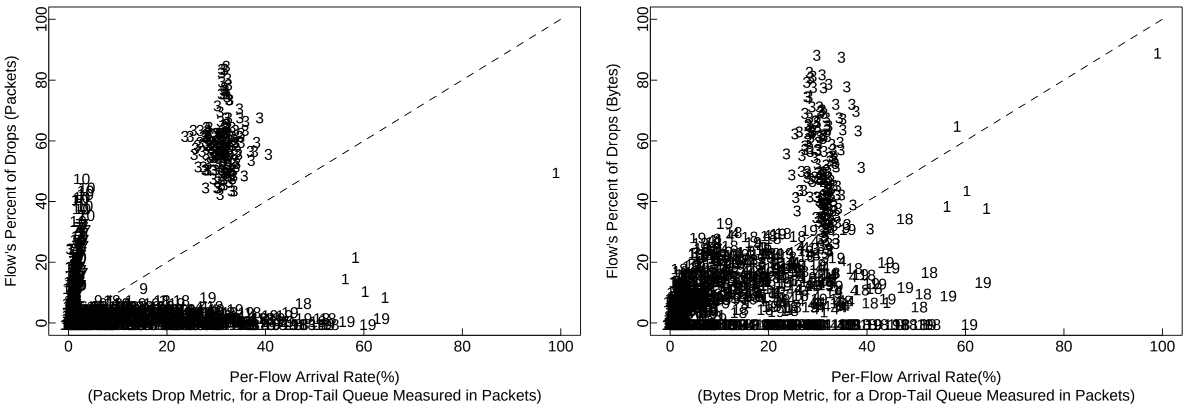

Figure 11: Comparing drop metrics for packet drops for a Drop-Tail queue measured in packets.

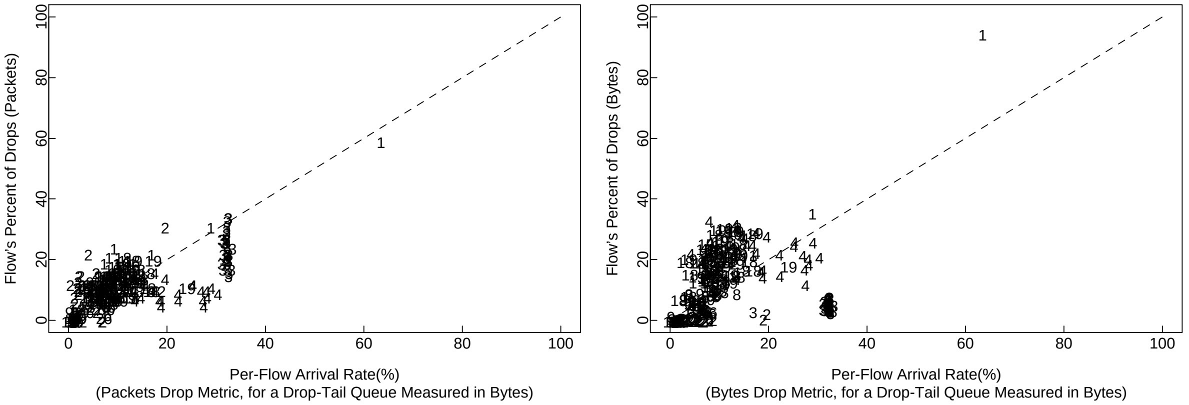

Figure 12: Comparing drop metrics for packet drops for a Drop-Tail queue measured in bytes.

cases to prevent this high packet drop rate, when coupled with sensible network provisioning.

C Illustrating the information available at the router

In this section we consider all of the information a router with RED queue management has easily available for detecting high-bandwidth flows. This consists of the recent history of packet drops, the average queue size, and, if the queue has a counter of packet arrivals, the recent fraction of arriving packets that have been dropped.

The simulation shown in Figure 14 uses the simulation network in Figure 13. For the simulation, we gradually start more TCP and CBR connections, each with infinite lifetime, so that the level of congestion gradually increases. The TCP flow have a range of packet sizes and receiver's advertised windows. One of the UDP flows has a CBR source that sends 190-byte packets every 3 ms, with an arrival rate that is roughly a third of the link bandwidth.

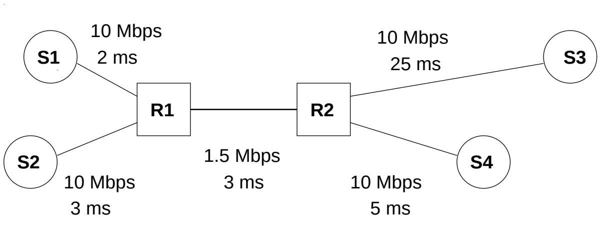

Figure 13: Simulation network.

We show that for flows using a significant fraction of the link bandwidth, the flow's drop metric is a good indicator for that flow's fraction of the arrival rate in bytes.

The top graph in Figure 14 shows the per-flow drop metrics computed for every 100 packet drops. The middle graph in Figure 14 shows the fraction of arriving packets dropped. The bottom graph in Figure 14 shows the average queue size. With the top two graphs, it is easy to single out the high-bandwidth flow that does not reduce its arrival rate as the level of congestion increases.

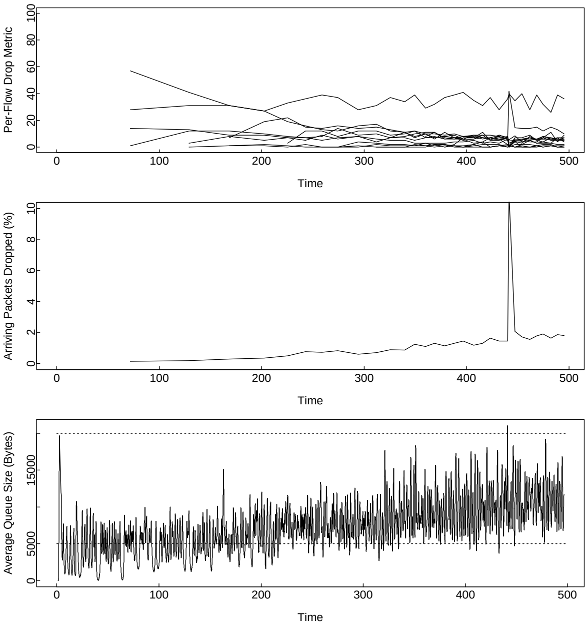

Figure 14: A simulation with rising congestion.

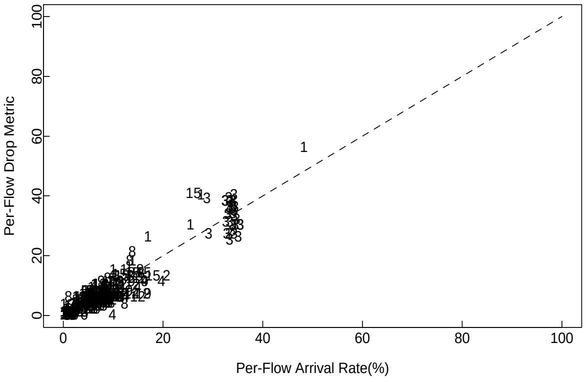

Figure 15: Verifying the drop metric.