INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 666-2900 • FAX (510) 666-2956

Controlling High Bandwidth Flows at the Congested Router

Ratul Mahajan[*] **and Sally Floyd**[†]
AT&T Center for Internet Research at ICSI (ACIRI)

TR-01-001

April 2001

Abstract

FIFO queueing is simple but does not protect traffic from flows that send more than their share or flows that fail to use end-to-end congestion control. At the other extreme, per-flow scheduling mechanisms provide max-min fairness but are more complex, keeping state for all flows going through the router. This paper proposes RED-PD (RED with Preferential Dropping), a flow-based mechanism that combines simplicity and protection by keeping state for just the high-bandwidth flows. RED-PD uses the packet drop history at the router to detect high-bandwidth flows in times of congestion and preferentially drop packets from these flows. This paper discusses the design decisions underlying RED-PD, and presents simulations evaluating RED-PD in a range of environments.

[*]Ratul is reachable at ratul@cs.washington.edu, and is also affiliated with the University of Washington.
[†]Sally is reachable at floyd@aciri.org.

1 Introduction

The dominant congestion-control paradigm in the Internet is one of FIFO (First In First Out) queueing at routers in combination with end-to-end congestion control. FIFO queueing is simple to implement, and because it involves no requirements for any uniformity of packet queuing, dropping, and scheduling in the routers along a path, it is well-suited to the heterogeneity and decentralized nature of the Internet. But FIFO scheduling provides little protection from high-bandwidth flows that consume a lot of bandwidth at the expense of other flows at the router. These high-bandwidth flows can be flows with small round-trip times, or worse, flows not using end-to-end congestion control. During times of congestion it is important to control the high-bandwidth flows to improve the performance of the rest of the traffic.

At the other extreme, per-flow scheduling mechanisms provide max-min fairness, but keep state (some of them even keep separate queues) for all the flows. This is an unnecessarily complex solution, particularly for best effort traffic, where most of the flows going through the router are small Web mice.

This paper addresses only best-effort traffic, and does not consider traffic protected by QoS mechanisms such as Differentiated Services. The vast majority of the best-effort traffic in the current Internet uses conformant end-to-end congestion control (i.e., TCP). However, there is substantial agreement that additional mechanisms are needed at routers to protect the Internet from "misbehaving" flows that don't use conformant end-to-end congestion control.

In this paper, we present RED-PD (RED [FJ93] with Preferential Dropping), a light-weight mechanism combining the simplicity of FIFO with some of the protection of full max-min fair techniques. RED-PD achieves this by keeping state for the high-bandwidth flows only. We call this *partial flow state*.

RED-PD is intended to operate in an environment dominated by end-to-end congestion control. When there is congestion at the router, RED-PD would control the throughput of high-bandwidth flows. These high-bandwidth flows could be TCP flows with short round-trip times, or misbehaving flows not using end-to-end congestion-control. Thus, in times of congestion RED-PD acts as a protection mechanism at the routers for the rest of the traffic.

RED-PD identifies high-bandwidth flows and controls the throughput of these flows using preferential dropping. RED-PD's identification mechanism is based on the RED packet drop history. The packet drops from active queue management are a reasonably-unbiased sample of the incoming traffic, and at the same time represent flows that have been sent congestion indications by the router. If a flow has many drops in the recent packet drop history, that flow is also likely to have a high arrival rate [FFT98]. The amount of drop history kept by RED-PD depends on the drop rate at the router and a configurable parameter that specifies the *target bandwidth* above which flows should be identified. Flows identified using this process are called *monitored* flows.

RED-PD probabilistically drops packets from a monitored flow at a pre-filter placed before the output queue. The dropping probability for a flow is determined using the identification process itself. If a monitored flow's arrival rate into the output queue is more than the target bandwidth, the flow is identified again and its dropping probability is increased. If the monitored flow's arrival rate into the output queue is much less than the target bandwidth, the flow's dropping probability is decreased. RED-PD suspends preferential dropping when there is insufficient demand from other traffic in the output queue, for example, when RED's average queue size is less than the minimum threshold.

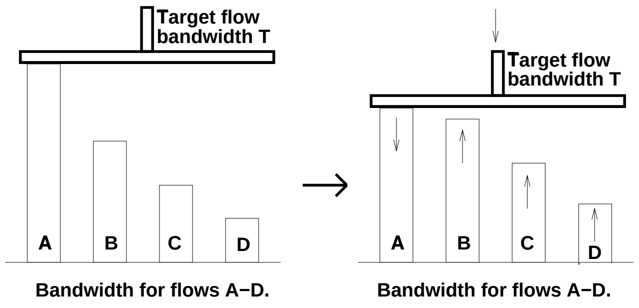

Figure 1: **Restricting flows to a target bandwidth T.**

Figure 1 illustrates RED-PD's impact on incoming traffic. Assume that flows are identified when their arrival rate is more than the target bandwidth T, and, when monitored, are restricted to T if there is enough demand from other flows. RED-PD has no effect when T is set higher than the maximum arrival rate of a flow. As T is pushed down, the bandwidth obtained by the monitored flows will be curtailed. This reduces the *ambient* drop rate, defined as the drop rate at the output queue, and enables the non-monitored flows to receive more bandwidth. Figure 1 shows the bandwidth for Flow A restricted to the target bandwidth T. As T is decreased, Flows B, C and D can receive increased bandwidth at the router.

In the next section we discuss existing proposals that use preferential dropping to improve fairness among flows. Section 3 discusses some trace-based results showing that controlling the small number of high-bandwidth flows can give significant control over the bandwidth distribution to a router. Section 4 describes RED-PD in detail. In Section 5, we evaluate RED-PD using analysis and simulations. A discussion of issues related to RED-PD is contained in Section 6, and we conclude in Section 7.

2 Related Work

In this section we briefly describe some existing proposals for achieving complete or limited fairness at a router.

2.1 Scheduling vs Preferential Dropping

Mechanisms for per-flow treatment at the router can be classified as based on either scheduling or preferential dropping. *Scheduling* approaches place flows in different scheduling partitions (there might be more than one flow in a partition), and the scheduling mechanism determines the bandwidth received by each partition. In contrast, *preferential dropping* mechanisms vary the dropping rate of a flow to control its throughput.

While scheduling mechanisms generally offer more precise control than preferential dropping mechanisms, scheduling mechanisms also generally have higher state requirements. At the same time, preferential dropping mechanisms have several advantages over scheduling-based schemes:

- Preferential dropping mechanisms preserve FIFO scheduling, which is good for low-bandwidth flows with bursty arrival processes. Scheduling mechanisms can introduce unnecessary delays for packets from such flows.
- Preferential dropping mechanisms work with active queue management to limit persistent queueing delay at the router.
- Preferential dropping mechanisms can easily be amended to actively punish high-bandwidth flows in times of congestion that are not using conformant end-to-end congestion control.

Scheduling mechanisms could possibly be modified to address each of the above issues, but it would make them more complex and we don't know of any research effort that has attempted to make these enhancements.

Figure 2 classifies the existing approaches based their control approach, and roughly places them along the continuum of per-flow treatment. The amount of flow state kept increases from right to left. Approaches with limited per-flow treatment generally start with the *identification* of exceptional flows for special treatment, while approaches with full per-flow treatment are generally based on the direct *allocation* of bandwidth.

2.2 Related Proposals

Floyd and Fall in [FF99] briefly discuss mechanisms for identifying high-bandwidth flows from the RED [FJ93] drop his-

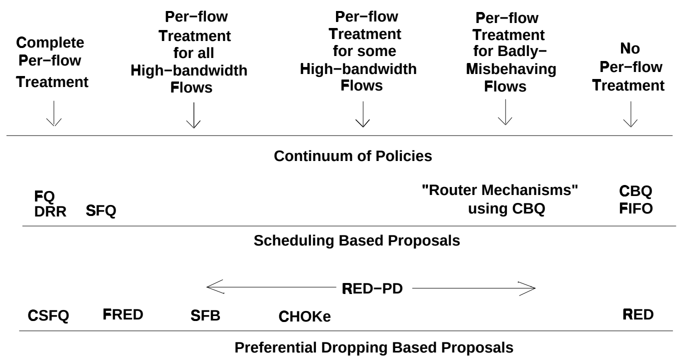
Complete Per−flow Treatment
Per−flow Treatment for all High−bandwidth Flows
Per−flow Treatment for some High−bandwidth Flows
Per−flow Treatment for Badly−Misbehaving Flows
No Per−flow Treatment
Continuum of Policies
FQ
DRR
SFQ
"Router Mechanisms" using CBQ
CBQ
FIFO
Scheduling Based Proposals
RED−PD
CSFQ
FRED
SFB
CHOKe
RED
Preferential Dropping Based Proposals

Figure 2: **A continuum of per-flow treatment at the queue.** The top line denotes the range of possible policies; the second and third lines show the rough placement along the continuum of proposals that use scheduling and preferential dropping respectively.

tory, using CBQ scheduling mechanisms to partition misbehaving and conformant flows in different classes. However, [FF99] did not present a complete solution, and this approach is limited by the choice of aggregate scheduling-based mechanisms instead of the per-flow preferential dropping mechanisms used in RED-PD. The RED-PD paper is in some sense a successor to [FF99], but using the per-flow preferential-dropping mechanisms explored in FRED and CSFQ.

The work in the paper draws heavily from Core-Stateless Fair Queuing (CSFQ) [SSZ98] and Flow Random Early Detection (FRED) [LM97], two approaches that use per-flow preferential dropping in concert with FIFO scheduling. The goal of CSFQ is to achieve fair queuing without using per-flow state in the core of an *island* of routers (an ISP network, for instance). On entering the network, packets are marked with an estimate of their current sending rate. A core router estimates a flow's fair share and preferentially drops a packet from a flow based on the fair share and the rate estimate carried by the packet. A key impediment to the deployment of CSFQ is that it would require an extra field in the header of every packet. Other drawbacks of CSFQ include the requirement that for full effectiveness, all the routers within the island need to be modified.

FRED is similar to CSFQ in that it uses FIFO scheduling, but instead of using information in packet headers, FRED constructs per-flow state at the router only for those flows that have packets currently in the queue. FRED was one of the first proposals to add preferential-dropping for selected flows to an environment with FIFO scheduling and active queue management. The dropping probability of a flow depends on the number of packets that flow has buffered at the router. FRED's fair allocation of buffers can yield very different fairness properties from a fair allocation of bandwidth [SSZ98]. In addition, the results obtained by FRED are not predictable, as they depend on the packet arrival times of the individual flows.

CHOKe [PPP00] is a recent proposal for approximating fair bandwidth allocation. An incoming packet is matched against a random packet in the queue. If they belong to the same flow, both packets are dropped, otherwise the incoming packet is admitted with a certain probability. The rationale behind this scheme is that high-bandwidth flows are likely to have more packets in the queue. CHOKe is not likely to perform well when the number of flows is large (compared to the buffer space) and even the high-bandwidth flows have only a few packets in the queue. The simulations in [PPP00] show that CHOKe achieves limited performance; for example, in the simulations the high-bandwidth UDP flows get much more than their fair share.

[PBPS01] presents an approach that is an outgrowth of both CSFQ and CHOKe. The router keeps a sample of arriving traffic, and an incoming packet is matched against this sample. The dropping probability of the incoming packet is determined by the number of packets in the sample from the same flow. This is a simultaneous but independent work that is still unpublished. (Our comments are based on discussions with one of the authors.)

Stochastic Fair Blue (SFB) [FKSS99] does not use per-flow state, but relies on multiple levels of hashing to identify high-bandwidth flows. As the authors state in their paper, the scheme works well when there are only a few high-bandwidth flows. In the presence of multiple high-bandwidth flows SFB could end up punishing even the low bandwidth flows as more and more bins get polluted.

There have been other papers dealing with controlling high bandwidth flows. SRED [OLW99] and LRU-RED [SR01] rely on a cache of recently seen flows to determine the high bandwidth flows. The scheme presented in [AT99] drops packets based on the buffer occupancy of the flow, and ERUF [AR99] uses source quench to have undeliverable packets dropped at the edge routers.

3 Why a Partial Flow State Approach Works?

In this section we present trace results that justify our belief that approaches like RED-PD that keep state for only high bandwidth flows can be effective. The traces that we examined show the same results found by others, e.g. [CMT98], that a small fraction of flows are responsible for a large fraction of the bandwidth. We also show that identifying and preferentially dropping from this small number of flows can be a powerful tool. Given the skewed distribution of bandwidth, controlling the bandwidth obtained by this small number of flows gives significant control to the router, enabling it to bring down the ambient drop rate rate, and in turn leading

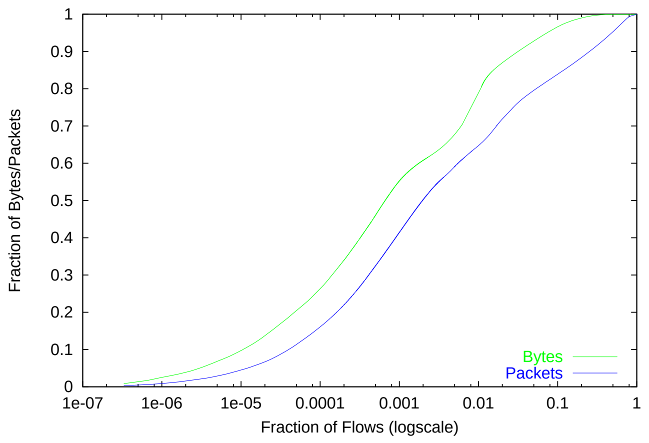

Figure 3: **Skewedness of bandwidth distribution**

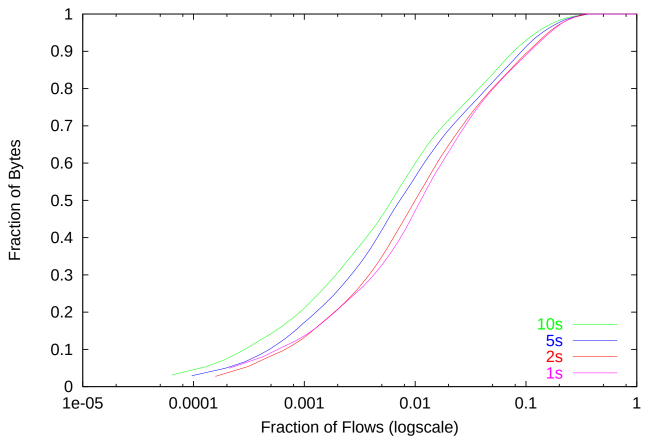

Figure 4: **Skewedness over smaller time scales.**

to higher throughput for other flows.

Figure 3 shows results from a one-hour-long trace taken from UCB DMZ in August 2000. The graph shows the fraction of flows responsible for a given fraction of bytes and packets in the trace. A flow here is defined by the tuple (source IP, source port, destination IP, destination port, protocol). A flow was timed out if it was silent for more than 64 seconds (the results with different timeout values are similar). It is clear from the graph that a mere 1% of the flows accounted for about 80% of the bytes and 64% of the packets. About 96% of the bytes and 84% of the packets came from just 10% of the flows. These numbers are similar to those obtained from NLANR [NLA], and to other results reporting on the heavy-tailed distribution of flow sizes.

The graph in figure 4 plots the same information for much shorter time windows. This shows the fraction of flows responsible for a given fraction of bytes and packets in a given

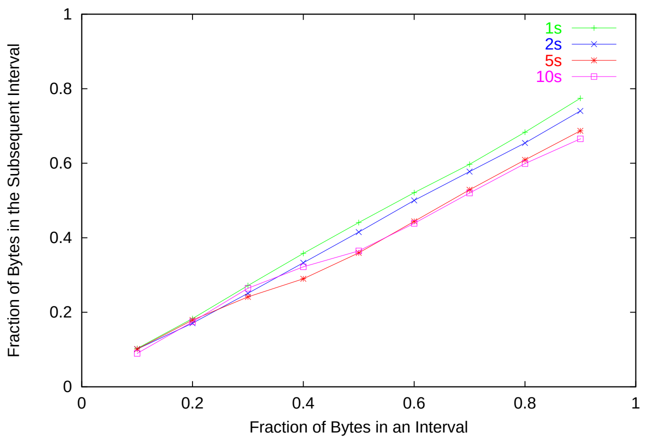

Figure 5: **Predictive nature of bandwidth usage.**

time interval. We can see that the skewedness holds not only for long time periods but also for smaller time windows, scales at which identification-based fairness approaches are likely to identify the high-bandwidth flows.

Another important property for an identification based approach to be successful is that the identified high-bandwidth flows in a given interval should be a good predictor of the high-bandwidth flows in the succeeding interval. Figure 5 plots the fraction of bandwidth consumed in the subsequent interval by flows which accounted for a particular amount of bandwidth (x-axis) in the current interval. For example, from the graph in Figure 4 we can see that in 5-second interval, 1% of the flows sent close to 50% of the bytes. Figure 5 tells us that these flows were responsible for 36% of the bandwidth in the next 5-second interval. If an identification-based scheme were to identify and restrict just this small fraction of flows, it would save a fair amount of bandwidth for other flows.

4 RED-PD

There are two components in RED-PD: identifying the high-bandwidth flows, and controlling the bandwidth obtained by these flows. The next two subsections discuss each of them respectively.

4.1 Identifying High Bandwidth Flows

RED [FJ93] (Random Early Detection) is a queue management algorithm that randomly drops (or marks) packets on detecting incipient congestion. Since RED drops are probabilistic, and not the result of a buffer overflow, they can be taken as random samples of the incoming traffic [FFT98]. RED-PD uses the RED drop history to identify flows sending more than the configured target bandwidth. The name RED-PD is something of a misnomer, as RED-PD is in fact not specific to RED, but can be used with any active queue management mechanism that distributes drops fairly.

A router with no limitations in terms of memory or CPU cycles could identify high-bandwidth flows by calculating directly the arrival rate for each flow over a given time interval. However, real routers do have limitations in terms of memory or CPU cycles, and keeping such a complete list of the arrival rate at the router for each flow is not necessary.

The packet drop history is a reasonably random sample of incoming traffic, and provides a cheap means for identification. A flow with more drops in the history is very likely to have a higher sending rate. At the same time, the drop history represents flows that have been sent congestion signals. Thus, RED-PD uses packet loss samples to roughly estimate the arrival rate of a flow and to confirm that the identified flow has in fact received loss events.

Now, we define the *target bandwidth* above which the flows should be identified, and describe how we restrict identification to high-bandwidth flows. The target bandwidth is defined as the bandwidth obtained by a *reference TCP flow* with the *target round-trip time* R and the drop rate p at the output queue.

Let $f(r,p)$ denote the average sending rate in pkts/s of a TCP flow with a round-trip time r and a steady-state packet drop rate p. From the deterministic model of TCP in [FF99], we have:

$$f(r,p) \approx \frac{\sqrt{1.5}}{r\sqrt{p}}. \tag{1}$$

Reasons for choosing this equation instead of the more precise one given in [PFTK98] are discussed in Appendix A.

RED-PD's goal is to identify flows that are sending at a rate higher than $f(R,p)$, the sending rate of the reference TCP. In the deterministic model of periodic packet drops, a TCP congestion epoch contains exactly one packet drop, and therefore contains $\frac{1}{p}$ packets. Hence, the *congestion epoch length* $CL(r,p)$ of a TCP flow with RTT r and drop rate p is

$$CL(r,p) = \frac{1}{f(r,p)p} = \frac{r}{\sqrt{1.5p}} \tag{2}$$

seconds.

Flows sending at a rate higher than $f(R,p)$ will have, on average, more than one drop in $CL(R,p)$ seconds, given a steady-state packet drop rate p. RED-PD maintains the packet drop history over $K \times CL(R,p)$ seconds, for some small integer K, and only identifies flows with K or more drops in this history.

Instead of keeping the drop history as a single big list, RED-PD partitions the history into multiple lists containing drops from consecutive intervals of time. RED-PD keeps M lists, where the length of a list is

$$\frac{K}{M} CL(R,p) = \frac{K}{M} \frac{R}{\sqrt{1.5p}} \tag{3}$$

seconds. Instead of identifying flows with K or more losses in a history of $K \times CL(R,p)$ seconds, RED-PD identifies flows with losses in at least K of M lists. Because of the use of multiple lists, flows with losses concentrated in only one or two lists are not identified.

There are several reasons why a flow might have multiple losses but not spread over K or more lists: because a single congestion event for that flow was composed of multiple drops from a window of data; because the flow reduced its sending rate after several round-trip times with drops; or because a low-bandwidth flow got unlucky and suffered more than its share of drops. In Appendix B we use simulations to show the advantages of using multiple lists.

The router estimates the packet drop rate at the queue in order to determine the size of a time interval. The router can directly measure its loss rate over a single interval as the number of drops divided by the number of arrivals. RED-PD uses exponential averaging to smooth the drop rate estimate over successive intervals.

4.1.1 Choosing Identification Parameters

We now discuss our choice of various parameters used in the identification scheme. We first consider K, the number of congestion epoch lengths $CL(R,p)$ that make up the drop history. Larger values of K make identification more reliable and make it more likely that the flow's arrival rate reflects the response of end-to-end congestion control to the packet losses received during that period. These benefits come at the expense of an increase in the time required to identify high-bandwidth flows. Smaller values of K increase the chances of catching unlucky flows, that is, low-bandwidth flows that happen to suffer more drops. Because we use the absence of a drop in all the lists as a criterion for decreasing the monitored flow's dropping probability (Section 4.2), a small value of K would lead to frequent changes in the dropping probability. In our simulations, a value of $K = 3$ gives a reasonably prompt response along with a reasonable protection for unlucky flows.

The next parameter to consider is M, the number of separate lists. Clearly, M should be greater than K because we identify flows that have drops in at least K lists. To count drops in the same window as a single loss event, the drop-list interval should be longer than the typical round trip time of flows in the queue. Given our choice of $K = 3$, a value of $M = 5$ has worked well in our simulations.

The parameter R, the target round-trip time, is the single most important parameter for RED-PD's identification mechanism. The choice of R will vary at different installations depending on factors like the composition of traffic, state limitations, and the desired degree of fairness. Increasing R increases the degree of fairness (as more flows are identified), and at the same time increases the state kept at the router. The effects and guidelines of choosing a particular R are discussed in detail in Sections 5.4 and 6.3.

4.2 Preferential Dropping

After identifying high bandwidth flows, we need to reduce the arrival rate of these flows to the output queue. The control mechanism should have the following properties:

- It should be light-weight and FIFO compatible.
- It should not only protect other traffic from the monitored flows, but also provide relative fairness among the monitored flows. (Lumping all monitored flows in one aggregate would lack this property.)
- The monitored flows should not be starved, and at the same time, not given more bandwidth than they would have obtained in the absence of monitoring. This rules out solutions that give "leftover bandwidth" to the monitored flows, or that give a fixed amount of bandwidth to a monitored flow without regard to the level of unmonitored traffic.

We now describe a technique, per-flow preferential dropping, which has all the desired properties. Preferential dropping places a *pre-filter* in front of the output queue. Packets from monitored high-bandwidth flows pass through this pre-filter where they are dropped with a certain probability, and the survivors are put in the output queue. Unmonitored traffic is put in the output queue directly. Figure 6 shows the architecture of a RED-PD router.

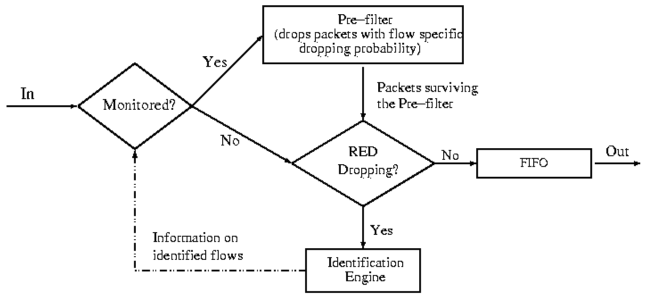

Figure 6: **Architecture of a RED-PD router.**

When a high-bandwidth flow is dropped in proportion to its excess sending rate, this simple mechanism provides relative fairness between monitored flows. RED-PD does not protect the monitored flows from the general congestion at the

```
Parameters
max_decrease: max probability reduction
                  in one step
$P_{minThresh}$:  max probability to free
                  a monitored flow

foreach $f$  (monitored flows that don't
              appear in any of the $M$ drop lists)
    $P$ = dropping probability of f
    if ($P > 2$*max_decrease)
        $P$ -= max_decrease
    else
        $P = P/2$
    if ($P \geq P_{minThresh}$)
        dropping probability of f = $P$
    else
        release f
```

Figure 7: **Pseudocode for reducing a flow's dropping probability.**

```
Variables
avg_drop_count: average number of drops for
                  flows identified in this round
    p:            current ambient drop rate

foreach $f$  (flows that appear in at least $K$
              of $M$ drop lists)
    if ($f$ is monitored)
        $P_f$ = dropping probability of f
    else
        $P_f = 0$
    $drop_f$ = number of drops of f
    $P_{delta}$ = ($drop_f$/avg_drop_count)*p
    if ($P_{delta} > P_f + p$)
        $P_{delta} = P_f + p$
    dropping probability of f += $P_{delta}$
```

Figure 8: **Pseudocode for increasing a flow's dropping probability.**

link, because the output queue does not differentiate between flows once the pre-filter has cut down on the monitored traffic. Thus, packets from the monitored flows can be dropped at the output queue as well as in the pre-filter.

There is a danger of transient link under-utilization if packets are dropped in the pre-filter despite low demand at the output queue. To avoid this, the pre-filter does not drop packets from monitored flows when there is insufficient demand at the output queue, as measured by RED's average queue size.

4.2.1 Computing the Dropping Probability

An important component of preferential dropping is determining the dropping probability of each monitored flow in the pre-filter.

One possible approach would be to measure directly the recent packet drop rate p in the output queue, and restrict each monitored flow's arrival rate to the output queue to the $target\_rate$ of $f(R, p)$. An obvious way to achieve this would be to use a token bucket instead of probabilistic dropping. A token bucket could lead to a undesirably-bursty pattern of losses for the monitored flow, but this is not necessarily a major problem, as the monitored flow has already been identified as high-bandwidth at the congested link. Another possible approach is measuring the arrival rate of the flow, and directly computing the dropping probability required to bring down the flow's rate into the output queue to target rate $f(R, p)$. We have not investigated either of these two approaches; some care would be required to respond appropriately to transient changes in the packet drop rate p.

In RED-PD, instead of enforcing a $target\_rate$ $f(R, p)$, we base each flow's dropping probability directly on the identification mechanism itself. The identification process only considers drops at the output queue, not in the pre-filter. Thus, the identification process is concerned with the flow's arrival rate to the output queue, not the arrival rate at the router itself; the two quantities would be different for a monitored flow.

A monitored flow is likely to continue to be identified if its arrival rate to the output queue is higher than $f(R, p)$. For such flows the dropping probability is increased. If the flow cuts down its sending rate and does not appear in any of the last M drop lists, its dropping probability is decreased. With this iterative increase and decrease, RED-PD settles around the right pre-filter dropping probability for a monitored flow. If the dropping probability of a flow becomes negligible, because the flow reduced its sending rate, the flow is unmonitored altogether.

The dropping probability for a monitored flow is not changed when the the flow appears in at least one but fewer than K of the M drop lists. This provides the necessary hysteresis for stabilizing the dropping probability. In addition, changes to the dropping probability are not made until a certain time period has elapsed after the last change to ensure that the flow has had time to react to the last change.

We now specify how RED-PD changes a flow's dropping probability. The pseudocode for reducing the dropping probability is given in Figure 7. The reduction in the dropping probability is bounded by a maximum allowable decrease in one step, *max_decrease*, to reduce oscillations. These oscillations could result from the reactions of the flow's end-to-end congestion control mechanisms to packet drops, or from the imprecision of using a flow's packet drop history as an estimate of its arrival rate. That is, the absence of the flow in all the drop-lists could be the result of it getting lucky, rather that from the flow's reduction in its sending rate. In such cases max_decrease ensures that control over a monitored flow is not loosened by a large amount in one step.

When increasing a flow's pre-filter dropping probability, both the ambient drop rate and the arrival rate of the flow need to be considered. When the ambient drop rate is high, high-bandwidth flows need to be brought down sooner, so the increase quanta should be large. In addition, different monitored flows will have different arrival rates to the output queue. The increase quanta should be larger for flows with higher arrival rates to the output queue.

Figure 8 shows the pseudocode for increasing a flow's dropping probability. At a given instant we have a group of identified flows whose dropping probabilities have to be increased. Let the drop rate in the output queue be p, and the average number of drops among the flows identified in this round be avg_drop_count. The equation below for a flow's increase quanta P_{delta} takes into account both the ambient packet drop rate and the relative sending rates of the monitored flows, as inferred from the ratio of drops.

$$P_{delta} = (drop_f / avg\_drop\_count) * p, \qquad (4)$$

where $drop_f$ is the number of drops for flow f. If this increase quantum is more than the flow's existing drop rate, we just double the flow's dropping probability (to make sure we don't increase a flow's drop rate all of a sudden). The existing drop rate for a flow is the sum of the drop rate at the pre-filter (zero for unmonitored flows) and the drop rate at the output queue.

5 Evaluation

We use a combination of analysis and simulation to evaluate RED-PD. Since identification is the first step in controlling a high-bandwidth flow, in §5.1 we study RED-PD's effectiveness in identifying high-bandwidth flows. Fairness is a crucial property for congestion control schemes. RED-PD's ability to enforce fairness using the iterative increase and decrease of a flow's dropping probability is investigated in §5.2. It is important that RED-PD react reasonably promptly to changes in a flow's sending rate, a property we analyze in §5.3. Finally, in §5.4, we demonstrate how the choice of R effects the degree of fairness and amount of state kept by the router. More simulation results are presented in Appendix E.

We carried out the simulations using the NS network simulator [NS][1]. Unless otherwise specified, the capacity of the congested link was 10 Mbps, RED-PD's target RTT R was 40 ms, the packet size was 1000 bytes, and RED was running in packet mode. The Selective Acknowledgement (SACK) [MMFR96] version of TCP was used, flows were started at a random time within the first 10 seconds, and aggregated results, where presented, were not taken before 20 seconds into the simulation.

[1]The source for running the simulations has been added to the ns distribution, and can be found in the directory ~ns/tcl/ex/red-pd.

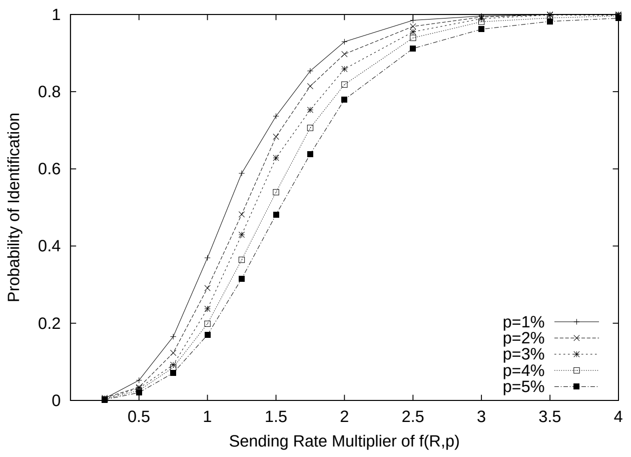

Figure 9: **The probability of identification of a TCP flow sending roughly at a rate $\gamma * f(R, p)$.**

5.1 Probability of Identification

In this section, we explore RED-PD's probability of identifying a TCP flow with a given round-trip time. The identification probability for CBR flows is analyzed in Appendix C. We show a flow's probability of being identified in a single identification round; the eventual throughput of the flow depends on whether the flow is persistently identified.

Figure 9 shows a TCP flow's probability of identification as a function of its sending rate and ambient drop rate at the queue. The simulations were done in a controlled environment where the ambient drop rate at the queue was fixed. The round-trip time of the TCP flow was varied to get flows sending at different rates. RED-PD used identification parameters $K = 3$ and $M = 5$. Figure 9 plots a TCP flow with a round-trip time of $\gamma * 40$ ms as sending at $\frac{40}{\gamma} * f(R, p)$ pkts/sec. For example, a TCP flow with a round-trip time of 80 ms is plotted as having a sending rate of $0.5 f(R, p)$ pkts/sec, and a TCP flow with a round-trip time of 20 ms is plotted as having a sending rate of $2.0 f(R, p)$ pkts/sec. Thus, the x-axis of Figure 9 gives an approximation to the flow's actual sending rate.

Figure 9 shows that a flow can be identified even if it is sending at less than $f(R, p)$ pkts/sec. This occurs when the flow has been unlucky, and has received more than its share of packet drops. Because RED is not biased in any way towards a particular flow, a flow sending at less than $f(R, p)$ pkts/sec is unlikely to be consistently unlucky in its packet drops. The

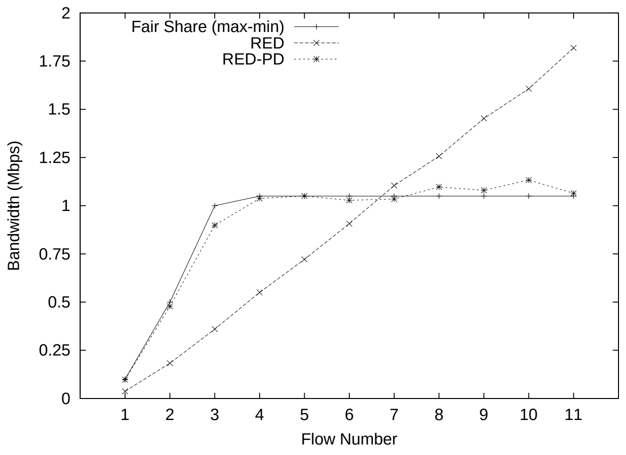

Figure 10: **Simulation with multiple CBR flows.** Flow 1 is sending at 0.1Mbps, flow 2 at 0.5 Mbps and every subsequent flow is sending at a rate 0.5 Mbps more than the previous flow.

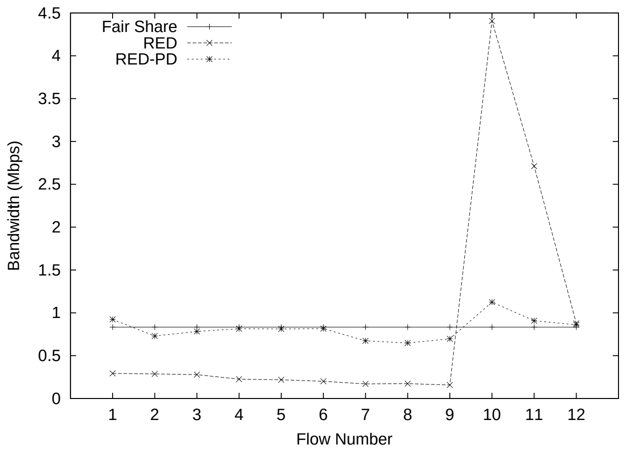

Figure 11: **Simulation with a mix of TCP and CBR flows.** Flows 1-9 are TCP flows with RTTs of 30-70 ms. Flow 10, 11 and 12 are CBR flows with sending rates of 5, 3and 1 Mbps respectively.

consequences of a flow getting identified once are not severe; it is monitored with a small initial dropping probability. Monitoring this flow further reduces its chances of being identified again, and thus this flow would soon be unmonitored.

Figure 9 also shows that a flow sending at more than $f(R, p)$ may escape identification in a particular round. This is not a concern, as this flow would be identified soon in another round in the near future.

5.2 Fairness

This section shows an important property of RED-PD: it is possible to approximate fairness among flows by iteratively increasing and decreasing the pre-filter dropping probability for the high-bandwidth flows. The simulations also show RED-PD's ability to protect the low-bandwidth flows and control the high-bandwidth ones.

The simulation in Figure 10 consists of 11 CBR flows of different rates. The sending rate of the first flow is 0.1 Mbps, that of the second flow is 0.5 Mbps, and every subsequent flow sends at a rate 0.5 Mbps higher than the previous flow (the last CBR flow sends at 5 Mbps). Separate lines in Figure 10 show the bandwidth received by each of the 11 CBR flows with RED and with RED-PD, while a third line shows each flow's max-min fair share. The graph shows that with RED, each flow receives a bandwidth share proportional to its sending rate, while with RED-PD all the flows receive roughly their fair share. Without RED-PD, the ambient drop rate is about 63%, while with RED-PD the ambient drop rate is reduced to roughly 4% by concentrating the dropping in the pre-filter for the high-bandwidth flows.

The next simulation has a mix of TCP and CBR flows. The aim is to study the effect of high-bandwidth CBR flows on conformant TCP flows and investigate RED-PD's ability to protect the conformant flows. There are 9 TCP flows and 3 CBR flows. The TCP flows have different round-trip times; the first three TCP flows have round-trip times close to 30 ms (there is some variation in the actual RTTs), the next three have RTTs around 50 ms, and the last three have RTTs around 70 ms. The CBR flows, with flow numbers 10, 11 and 12, have sending rates of 5 Mbps, 3 Mbps and 1 Mbps respectively. Again, Figure 11 shows the bandwidth of each of the 12 flows with RED and with RED-PD. With RED, the high-bandwidth CBR flows get almost all the bandwidth, leaving little for the TCP flows. In contrast, RED-PD is able to restrict the bandwidth received by the CBR flows to near their fair share. Given the target R of 40 ms, RED-PD monitors not only the CBR flows, but also the TCP flows with RTTs of 30 ms (and occasionally those with 50 ms as well). Each of the CBR flows received a different pre-filter dropping rate, as each CBR flow was successfully restricted to roughly its fair share.

5.3 Response Time

This section is devoted to studying the response time of RED-PD to a sudden increase or decrease of a flow's sending rate. A detailed analysis is presented in Appendix D. We present the results and verify them using simulations here.

Assume that a flow suddenly starts sending at $\gamma * f(R, p)$ pkts/s. Under the assumption that the ambient drop rate at the router is independent of the sending rate of a single flow, as would be the case with a high degree of statistical multiplex-

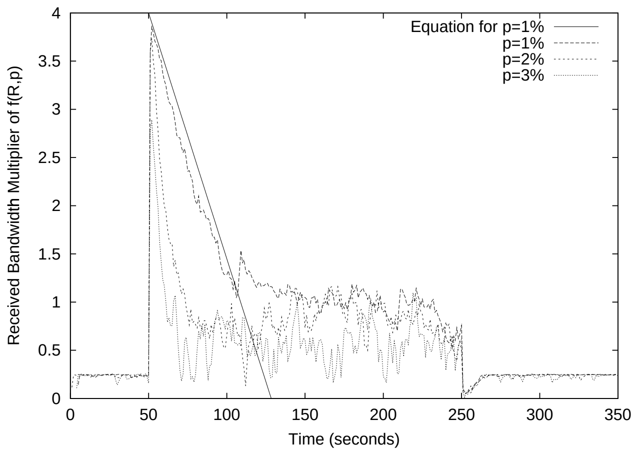

Figure 12: **The response time of RED-PD.** The CBR source increases its sending rate to $4*f(R,p)$ at $t = 50s$ and reduces it back to $0.25 * f(R,p)$ at $t = 250s$.

ing, Appendix D gives the time taken by RED-PD to clamp this flow to $\alpha * f(R,p)$ pkts/s as roughly:

$$t_{cutdown} = \frac{(\gamma - \alpha)RK(M-1)}{\gamma p\sqrt{1.5p}M} \tag{5}$$

Figure 12 shows a simple simulation to test Equation 5. A CBR source passed through a queue with a fixed loss rate. The CBR source initially sends at $0.25*f(R,p)$ and increases its sending rate to $4*f(R,p)$ (that is, $\gamma = 4$) at $t = 50s$. The line marked "Equation" is based on Equation 5, and the rest of the lines are simulation results for the received bandwidth averaged over 1-second intervals. It can be seen that the equation predicts the simulation results very closely.

Equation 9 in Appendix D.2 gives the time for RED-PD to completely release a flow after it reduces its sending rate. For the simulation in Figure 12, with the sending rate of $4 * f(R,p)$ and $p = 1\%$, the release time from Equation 9 comes out to be 10.58 seconds, which is very close to what we see in the simulations. (For Equation 9, $P = 0.75$ and p_d=0.05, giving $n = 13$ and $m = 5$.)

In order to verify the analysis, the above simulation was done in a controlled environment where the output queue had a constant configured drop rate. Figure 13 shows the results from a normal simulation with one CBR flow and 9 TCP flows over a 10 Mbps link. The CBR flow was started with the initial rate of 0.25 Mbps. At $t = 50s$ the CBR flow increases its sending rate to 4 Mbps, and at $t = 250s$ it decreases its sending rate back to 0.25 Mbps. The RTT of the TCP flows ranged from 30 to 70 ms.

In this simulation, RED-PD took just 7 seconds to bring down the throughput of the CBR flow. The big difference from the analysis comes from the fact that, in the simulation, the sudden increase in the sending rate of the CBR flow leads to an increased drop rate at the router (visible in the lower graph). Thus, if a flow's increased rate also increases the drop rate at the router, the flow would be brought down much sooner. If a flow's high sending rate doesn't change the drop rate significantly, then a fast reaction is not critical in the first place.

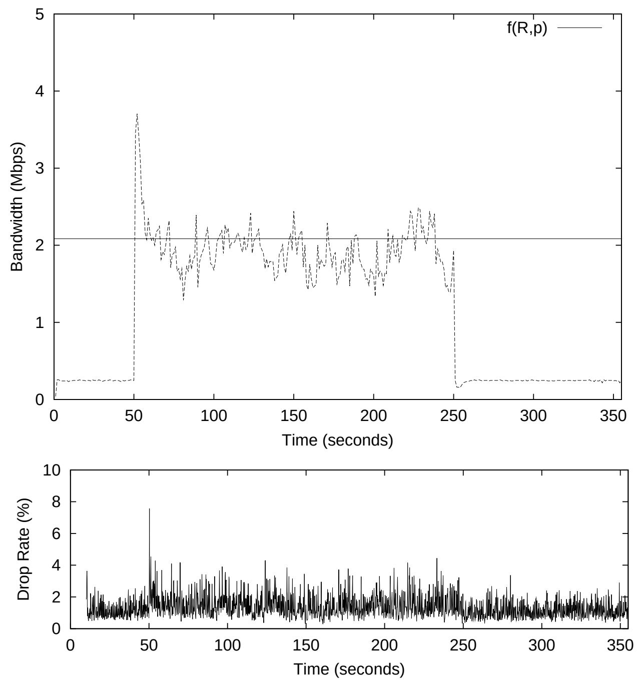

Figure 13: **Adapting the dropping probability.** The top graph shows the throughput of a CBR flow which changes its sending rate to 4 Mbps at $t = 50s$ and back to 0.25 Mbps at $t = 250s$. The line labeled $f(R,p)$ is based on the ambient drop rate seen over the whole simulation. The bottom graph plots the ambient drop rate over time.

The speed of RED-PD's reaction depends on both the ambient drop rate and the arrival rate of the monitored flow. If the ambient drop rate is high or the flow's arrival rate is higher than other monitored flows, the increase quanta is large and the dropping probability is increased faster. So, if a flow increases its sending rate to huge levels, it would be brought down fairly quickly.

5.4 Effect of R, the Target RTT

The simulations in this section illustrate how the choice of RED-PD's configured RTT parameter R affects both the identification of flows for monitoring and the bandwidth received by monitored flows. Each column in Figure 14 represents a different simulation, with a different value for R, ranging

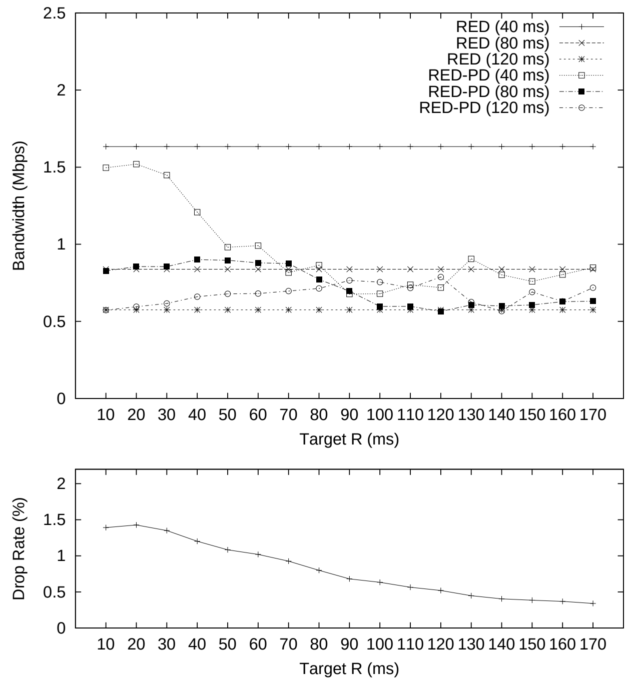

Figure 14: **The Effect of Target R.** The top graph shows the bandwidth received by 40 ms, 80 ms and 120 ms RTT TCP flows for different values of R. The bottom graph plots the ambient drop rate.

from 10 ms to 170 ms. In each simulation 14 TCP connections were started, two each with RTTs of 40 ms, 80 ms and 120 ms, and the rest with RTTs of 160 ms. The top graph of Figure 14 shows the average bandwidth received by the TCP flows with round-trip times from 40-120 ms, while the bottom graph of Figure 14 shows the ambient drop rate. The horizontal lines in Figure 14 show the bandwidth for each traffic type with RED.

For the simulations with R less than 40 ms, RED-PD rarely identifies any flows, and the bandwidth distribution is essentially the same as it would be with RED. However, for the simulations with R of 40 ms or higher, the short TCP flows with 40-ms RTTs start to be identified and preferentially dropped. Note that as R is increased, the bandwidth received by the short TCP flows is decreased, because the target bandwidth for a monitored flow, $f(R, p)$, decreases as R increases. In addition, as R is increased the ambient drop rate decreases and the throughput for the long TCP flows increases (though this is not shown in Figure 14).

As these simulations illustrate, increasing RED-PD's configured value of R results in more flows being monitored, and more drops occurring in the pre-filter for the monitored flows. Thus, as R is increased, RED-PD gets closer to full max-min fairness. In addition, increasing R decreases the ambient drop rate, and therefore increases the bandwidth available to web mice and other unmonitored flows. The simulations also show that, with a very small value for R, RED-PD has limited impact at the router, and can be used with the goal of controlling only egregiously-misbehaving flows or those conformant flows with very short round-trip times.

5.5 Additional Simulations

The following simulations have been included as Appendix E due to space considerations.

- §E.1: The throughput of monitored flows during a persistent ambient drop rate;
- §E.2: Simulations involving Web traffic.
- §E.3: The effect of RED-PD on a mix of TCP and TFRC [FHPW00] traffic.
- §E.4: RED-PD's effect on flows traversing multiple congested links;
- §E.5: RED with byte mode operation.

All of them have yielded results that show the effectiveness of RED-PD for the properties it was evaluated for.

6 Discussion

This section discusses some issues pertaining to RED-PD. We talk about the fairness properties, state requirements, choosing the target RTT, and dealing with unresponsive and evasive flows.

6.1 Fairness Properties

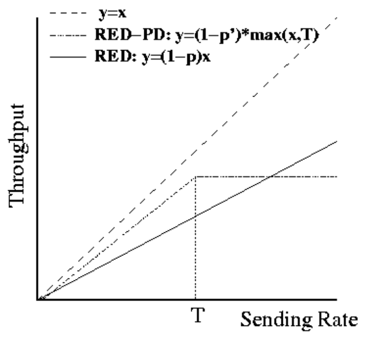

Figure 15: **The Ideal Fairness Properties of RED-PD.** In the figure p is the drop rate with RED, p' is the ambient drop rate with RED-PD, and T is the target bandwidth.

RED-PD provides limited max-min fairness, in that it controls the bandwidth allocated to the high-bandwidth flows. The effect of RED-PD is summarized in Figure 15. Figure 15 assumes an environment with a fixed packet drop rate p, and shows that with RED, the bandwidth received by a flow is proportional to the arrival rate of that flow. In contrast, a full max-min fairness scheme like Fair Queueing does not let a flow get more bandwidth than another flow whose demand has not been met. RED-PD aims to provide limited max-min fairness, in which we restrict the bandwidth received by high-bandwidth flows. Figure 15 shows that for the ideal RED-PD, the bandwidth received by a flow is limited by the target sending rate T, which, from Equation (1), is inversely proportional to the target RTT R.

In environments where all of the high-bandwidth flows are conformant and have round-trip times considerably larger than the target RTT R, RED-PD would have no impact. However, in environments with high-bandwidth non-conformant flows or with flows with round-trip times less than R, RED-PD changes the bandwidth allocation of the underlying system by restricting the throughput of high-bandwidth flows. RED-PD's control of the arrival rate of high-bandwidth flows to the output queue is accompanied by a reduction in the ambient drop rate, which is the drop rate seen by unmonitored flows. This decrease in the ambient drop rate results in an increase in the bandwidth received by unmonitored flows.

6.2 State Requirements

In addition to the state needed by a regular RED queue, RED-PD requires state for the identification engine and monitored flows. The identification engine stores M drop lists. The amount of memory required depends on the target RTT R, the ambient drop rate and the number of flows competing at the queue. For example, with $R = 40$ ms, and $p = 1\%$, the router needs to store information about packets dropped over the past 1 second, which should not be a problem even for high-speed routers. It should also be noted that fast memory is not required for storing drop lists as the identification engine does not run in the forwarding fast path. In rare cases when the router does not have enough memory to store the headers for all the drops, it can either randomly sample the drops, or sort and store just the drops from the high senders (as represented by the flows with the most drops) when retiring the current list to start a new one. This would restrict identification to just the highest-bandwidth flows, and should not lead to any major changes in the behavior of RED-PD.

In addition to the drop history, RED-PD keeps state for monitored flows. When a packet arrives, the router determines if it belongs to a monitored flow, and applies the appropriate preferential dropping to the packet before adding it to the output queue. Lookups matching the forwarding speed can be achieved using sparsely populated hash tables or perfect hash functions. It helps that RED-PD does not keep state for all the flows going over the link. A quick look at Figure 4 tells us that only 10% of the flows accounted for more than 80% of the link bandwidth. In this situation, by keeping state for only 10% of the flows RED-PD would be very effective in controlling the bandwidth distribution on the link. A detailed investigation of the state requirements and fairness tradeoffs involved in RED-PD under various traffic scenarios is a subject of future work.

The complexity of a scheme is not given by the amount of state alone, but is also dependent on the processing done on that state. Table 1 compares RED-PD's complexity with that of several other proposed mechanisms.

6.3 Choosing R, the Target RTT

As was illustrated by the simulations in Section 5.4, the choice of the target round-trip time R determines RED-PD's operating point along the continuum of greater or lesser per-flow treatment at the congested queue. A larger value for R results in greater per-flow treatment, requiring more state at the router, and coming closer to full max-min fairness. In contrast, a smaller value for R leads us to the opposite end of the spectrum.

Instead of a fixed, configured value for R, another possibility is to vary R dynamically, as a function of the ambient drop rate and/or of the state available at the router. We intend to explore techniques for dynamically varying R in later work.

6.4 Unresponsive Flows

It is important for schemes that provide differential treatment for flows to provide incentives for end to end congestion control by actively punishing misbehaving flows. However, in this work we have addressed this issue of actively punishing misbehaving flows only briefly.

RED-PD keeps a history of the arrival and drop rates for each monitored flow. A monitored flow is declared *unresponsive* when its arrival rate has not reduced in response to a substantial increase in its drop rate. For flows identified as unresponsive, RED-PD increases the drop probability more quickly, and decreases the drop probability more slowly, to keep the unresponsive flow under tighter control. However, RED-PD does not necessarily reduce the bandwidth obtained by an unresponsive flow, compared with the bandwidth it would have received from RED-PD without having been identified as unresponsive.

RED-PD's test for unresponsiveness can have false positives, in that it could identify some flows that are in fact responsive. The arrival rate of a flow at the router depends not only on the drops at that router, but also on the demand from the application, and the drops elsewhere along the path. In ad-

	State for What	What State	Fast-path Processing	When required
FQ	All flows	Queues	Queue management, scheduling	Packet arrival, departure
FRED	All buffered flows	Count of buffered packets	Drop probability computation, coin tossing	Packet arrival, departure
CSFQ (edges)	All flows	Arrival rate estimate, time of last packet	Update arrival rate estimate, update header	Packet arrival
RED-PD	High-bandwidth flows	Dropping probability, drop history	Coin tossing	Packet arrival

Table 1: **A comparison of complexity of some schemes.**

dition, the router does not know the round-trip time of the flow or the other factors (e.g., multicast, equation-based congestion control mechanisms) that affect the timeliness of the flow's response to congestion. The test for unresponsiveness can also have false negatives, in that it might not detect many high-bandwidth flows that are unresponsive.

With its iterative increase and decrease of a flow's drop rate, RED-PD provides an ideal framework for determining the conformance of a flow. Future work will include the investigation of a better unresponsiveness test, and of possibilities for decreasing the throughput for unresponsive monitored flows to significantly less than their fair share, as a concrete incentive towards the use of end-to-end congestion control.

6.5 Evasive Flows

Given a complete knowledge of the RED-PD identification mechanisms at the router, a high-bandwidth flow could possibly evade the identification procedure by restricting its sending bursts to at most $K - 1$ of M identification intervals. A flow is not likely to be able to do this without a precise knowledge of the length and start times of the identification intervals, which are not fixed but change with the drop rate at the router. However, it is true that the more bursty the sending pattern of a flow over successive identification intervals, the less likely it is to be detected by the identification mechanism. To protect against bursty flows, identification could extend to flows that receive drops in less than K of M drop lists, but have a very high number of packet drops in these intervals.

6.6 Explicit Congestion Notification

[RF99] is a proposal to add Explicit Congestion Notification (ECN) to the IP protocol, so that routers can indicate congestion to end-nodes by setting a bit in the IP packet header. RED-PD can operate in the presence of ECN by taking into account the packet mark history as well as the packet drop history at the router. In addition, for an ECN-capable flow, RED-PD's pre-filter could start with marking packets, and advance later to dropping packets for a non-responsive flow.

6.7 IPsec

This paper assumes that the router is able to identify flows by the IP source and destination addresses, protocol field, and source and destination port numbers in the packet header. For IPsec traffic, some of this information is not available to the router. In this case, routers could use the triple in the packet header defining the IPsec Security Association to identify flows or flow aggregates.

6.8 Aggregate-based Congestion

A flow-based congestion control mechanism can do very little when congestion is caused by an aggregate such as a flash crowd or denial of service (DoS) attack. Flow-based mechanisms at a router can be augmented with aggregate-based congestion control, to protect the rest of the traffic on that link from an overall increase in the packet drop rate.

At some level, aggregate-based congestion control can be thought of as a variant of RED-PD applied to aggregates rather than to individual flows, in that aggregate-based congestion control enforces an upper bound on the bandwidth given to an identified aggregate at the router in a time of congestion. However, there are substantial differences between flow-based and aggregate-based congestion control at the router. As an example, the use of the TCP throughput equation is appropriate for individual flows (as defined by source and destination IP addresses and port numbers) but not for aggregates of flows. In addition, there are no appropriate fairness metrics for aggregates, or even well-defined definitions of the aggregates themselves.

7 Conclusions

We have presented RED-PD, a mechanism that uses the packet drop history at the router to detect high-bandwidth flows in times of congestion, and preferentially drops packets from these flows. We have shown that the proposed mechanism successfully controls the bandwidth obtained by high-bandwidth flows in a range of environments and increases the fairness among flows. We also showed that RED-PD has a reasonable response time to sudden changes in a flow's arrival rate, reacting faster when the flow is consuming too much bandwidth or the drop rate at the router is high. The level of fairness provided by RED-PD can be controlled using the target RTT R.

Acknowledgments

We are grateful to Scott Shenker for extensive discussions, and for reviewing an earlier version of this paper. Ratul is grateful to David Wetherall, who has been very supportive throughout and provided useful feedback. Vern Paxson helped with trace data and analysis. We are thankful to Yin Zhang for discussions that resulted in the multiple list identification approach, Neil Spring for reviewing a draft of this paper, Dina Katabi for helping with the simulations, Kevin Fall for his scripts from [FF99], and Mark Handley and Jitendra Padhye for TFRC.

References

[AR99] Anurag Acharya and Anand Rangarajan. Early Regulation of Unresponsive Flows, July 1999. UCSB Tech Report TRCS99-26.

[AT99] F. Anjum and L. Tassiulas. Fair Bandwidth Sharing among Adaptive and Non-adaptive Flows in the Internet. In *IEEE INFOCOM*, pages pp. 1412– 1420, March 1999.

[CMT98] Kim Claffy, Greg Miller, and Kevin Thompson. The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone. In *INET*, July 1998.

[FF99] Sally Floyd and Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, Vol. 7(4):pp. 458–473, August 1999.

[FFT98] Sally Floyd, Kevin Fall, and Kinh Tieu. Estimating Arrival Rates from the RED Packet Drop History, April 1998. http://www.aciri.org/floyd/end2end-paper.html.

[FHPW00] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-Based Congestion Control for Unicast Applications. In *ACM SIGCOMM*, August 2000.

[FJ93] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, Vol. 1(4):pp. 397–413, August 1993.

[FKSS99] Wu-Chang Feng, Dilip D. Kandlur, Debanjan Saha, and Kang Shin. Blue: A New Class of Active Queue Management Algorithms. Technical Report CSE-TR-387-99, University of Michigan, April 1999.

[LM97] Dong Lin and Robert Morris. Dynamics of Random Early Detection. In *ACM SIGCOMM*, September 1997.

[MMFR96] Matt Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow. TCP Selective Acknowledgement Options. RFC 2018, April 1996.

[NLA] NLANR Web Page: http://www.nlanr.net.

[NS] NS Web Page: http://www.isi.edu/nsnam.

[OLW99] Teunis J. Ott, T. V. Lakshman, and Larry Wong. SRED: Stabilized RED. In *IEEE INFOCOM*, March 1999.

[PBPS01] Rong Pan, Lee Breslau, Balaji Prabhakar, and Scott Shenker. Approximate Fairness through Differential Dropping, 2001. Work in progress.

[PFTK98] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *ACM SIGCOMM*, August 1998.

[PPP00] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *IEEE INFOCOM*, March 2000.

[RF99] K. K. Ramakrishnan and Sally Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481, Jan. 1999.

[SR01] Smitha and A.L. Narsimha Reddy. An Active Queue Management Scheme to Contain High Bandwidth Flows at a Congested Router. Masters Thesis, Texas A&M University, May 2001. http://www.isc.tamu.edu/ smitha/thesis.ps.

[SSZ98] Ion Stoica, Scott Shenker, and Hui Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *ACM SIGCOMM*, September 1998.

A Choice of TCP Response Equation

In this section we explain our use of $f(r,p)$ in Equation 1 for determining the congestion epoch length used by RED-PD in the identification phase.

Instead of Equation 1, RED-PD could instead use the equation $f_1(r,p)$ given in [PFTK98]:

$$f_1(r,p) = \frac{1}{r\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}. \quad (6)$$

The TCP retransmit timeout value t_{RTO} can be approximated as $4r$. This equation incorporates the effects of retransmit timeouts, and is based on a model of Reno TCP experiencing independent packet drops. While the TCP throughput equation $f_1(R,p)$ more accurately models TCP behavior, it basically gives the long-term sending rate of a TCP connection. A conformant TCP flow that has not suffered a retransmit timeout in the most recent several congestion epochs might send at a rate higher than $f_1(R,p)$ over that period. The equation for $f(R,p)$ is closer to the sending rate of the TCP flow over the short term (of several congestion epochs) in the absence of retransmit timeouts. For low to moderate levels of congestion, $f(R,p)$ and $f_1(R,p)$ give similar results, and the difference is negligible. However, for higher packet drop rates, when a congestion epoch is quite short, a flow could easily go for several epochs without receiving a retransmit timeout, and in this case it would seem desirable to use $f(R,p)$ to be properly conservative in our identification of high-bandwidth flows.

B Single List vs Multiple Lists

The *multiple-list identification* scheme identifies flows that receive losses in K out of M drop-list intervals. This could be compared to *single-list identification*, which would identify flows that receive the largest number of drops in a single, larger interval.

The main advantage of multiple-list identification over single-list identification is that multiple-list identification ignores flows that suffered drops in only a few lists. For example, there are several reasons why a flow might have several drops in one list, but no drops in other lists: because a single congestion event for that flow was composed of multiple drops from a window of data; because the flow reduced its sending rate after drops; or because of simple bad luck unrelated to the flow's sending rate.

In an environment with RED and a moderate packet drop rate, a flow is unlikely to receive multiple drops in a single window of data, and therefore each loss event for a flow would be likely to consist of a single packet loss. In such an environment, there might be little difference between a single-list and a multiple-list identification scheme. However, in environments with higher drop rates or with highly bursty arrival patterns, a multiple-list identification scheme could have significant advantages over a single-list identification scheme.

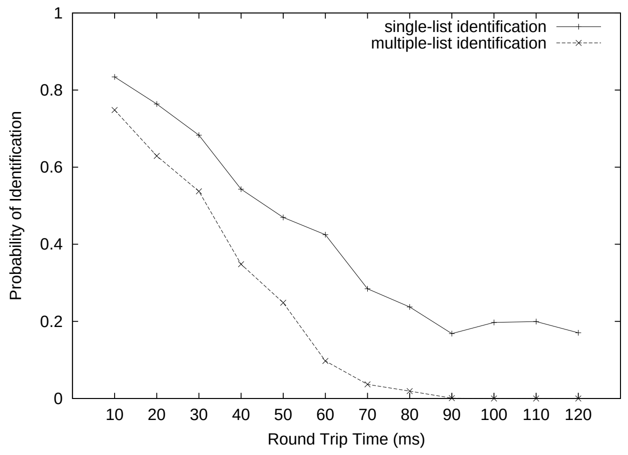

Figure 16: **The probability of identification for single and multiple list identification schemes for a bursty loss environment.** The target RTT was 40 ms.

In a simulation we created an environment likely to show the advantages of a multiple-list scheme over the single-list scheme. The congested link has a small buffer space, with the RED threshold min_{th} set to half of the buffer space. This scenario is characterized by frequent buffer overflow, with multiple packets dropped from a window of data. Figure 16 shows the fraction of times a TCP flow with the given RTT was identified by the two schemes. The single-list scheme identifies a flow when it experiences K or more drops in the last $K * CL(R,p)$ seconds. It can be seen that in this environment a single-list scheme based on individual losses has a high probability of identifying conformant flows with round-trip times greater than R. In contrast, the multiple list scheme does a better job of identifying only the higher bandwidth flows.

For the scenarios in Figure 16, a single-list identification scheme based on loss events would perform much better than the single-list scheme based on individual losses. This would be a single-list scheme that identifies a flow if the flow receives K or more loss *events* in a single detection interval of duration $K * CL$, where a loss event is defined as one or more losses in a short period of time (such as a typical round-trip time). However, a single-list identification scheme based on loss events would require a more complex implementation than a single-list scheme based on individual losses, as timing information would be required with every drop.

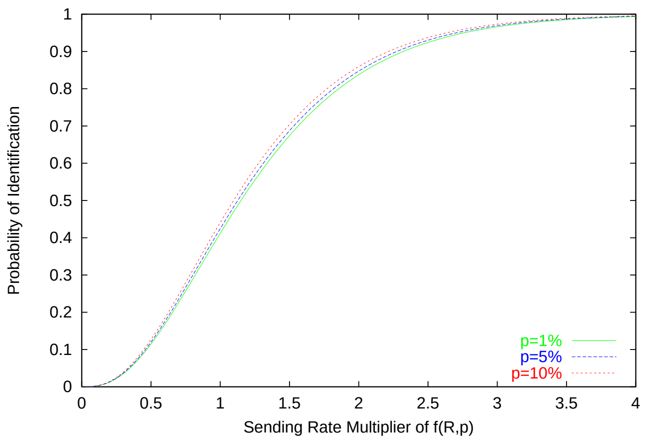

Figure 17: **The probability of identification of a flow sending at a rate $\gamma * f(R,p)$.**

C Probability of Identification for CBR flows

Section 5.1 used simulations to investigate RED-PD's probability of identifying a TCP flow. In this section, we use both analysis and simulations to give RED-PD's probability of identifying a CBR flow.

Consider a CBR flow sending at $\gamma \times f(R,p)$ pkts/s, where p is the ambient drop rate and R is RED-PD's target RTT. Given the length of a drop-list interval from Equation 3, the flow sends $\frac{\gamma K}{Mp}$ packets per drop-list interval. The probability $P(1)$ that the flow suffers at least one drop in a drop-list interval is

$$P(1) = 1 - (1-p)^{\frac{\gamma K}{Mp}}$$

For a flow to be identified, it has to suffer at least K drops in M drop-list intervals. So, the probability of this flow being identified is

$$\begin{aligned} P_{identification} = \\ & C(M,K)P(1)^K P'(1)^{M-K} + \\ & C(M,K+1)P(1)^{K+1}P'(1)^{M-K-1} + \\ & \cdots + C(M,M)P(1)^M, \end{aligned} \tag{7}$$

for $C(x,y)$ the number of ways of choosing y out of x objects.

Figure 17 plots a flow's probability of identification as a function of its sending rate using the equation above, for a range of values for the ambient drop rate. RED-PD's identification parameters in the simulation are $K = 3$ and $M = 5$. This result was confirmed using simulations with a fixed drop rate at the queue. The graph should be interpreted carefully, as the x-axis is the rate multiplier of $f(R,p)$, which itself is a function of the ambient drop rate p. As the ambient drop rate increases, $f(R,p)$ decreases, and a flow sending at a fixed rate in pkts/s becomes more likely to be identified.

D Response Time

In this section we do a simplified analysis of the time taken by RED-PD to control a high-bandwidth CBR flow, as well as the time taken to release a flow which has reduced its sending rate.

D.1 Time to Cutdown

Assume that a flow increases its sending rate all of a sudden to $\gamma * f(R,p)$ pkts/s, for $(\gamma > 1)$, where p is the prevalent drop rate at the output queue and R is the target RTT. We make the following simplifying assumptions in the analysis:

1. The loss rate at the output queue is independent of this flow's arrival rate in the queue. In reality, the loss rate at the output queue can go up when the CBR flow suddenly starts sending at a high rate, and come back down again when the flow is controlled in the pre-filter.

2. This is the only flow whose dropping probability is being increased. From Equation (4), this means that the increase quanta of the dropping probability would be p.

3. The flow is successfully identified in each round. This is likely to be true until the flow is brought down to about twice $f(R,p)$. As seen in Figure 9 and 17, the probability of identifying the flow is high for a flow sending at twice $f(R,p)$.

We calculate the time required to bring down the arrival rate of the flow in the output queue to $\alpha * f(R,p)$. The dropping probability required in the pre-filter in this case is $\frac{\gamma-\alpha}{\gamma}$. Because of Assumption 2, the dropping probability increase is in quantum of p. Hence the number of rounds required are $\frac{\gamma-\alpha}{\gamma p}$. Each round is $M-1$ intervals long because after increasing the pre-filter dropping probability, we wait for $M-1$ intervals and see if there are drops in K of the last M drop-list intervals. Substituting the length of a drop-list interval from Equation 3, the total time required is

$$t_{cutdown} = \frac{(\gamma-\alpha)RK(M-1)}{\gamma p\sqrt{1.5p}M} \tag{8}$$

seconds.

D.2 Time to Release

We now estimate the time required to release a flow, that is, the time taken to transfer a monitored flow to the unmonitored category after it reduces its sending rate. The time estimate tells us not only how long this flow will be penalized after a rate reduction, but also the time required by RED-PD to forget a monitored flow which ceases to exist. The only assumption we make in this computation is that the flow is no longer identified after it cuts down its sending rate. The assumption holds as long as the reduced sending rate is much less than $f(R,p)$.

Consider a flow being monitored with a pre-filter dropping probability of P. This flow would become unmonitored when the pre-filter dropping probability goes below $P_{minThresh}$. In each round the dropping probability is reduced by either a factor of β or a fixed amount p_d, whichever leads to a lesser reduction; p_d is the upper bound on the probability reduction in one step. Assume that there are n subtractive reduction rounds followed by m multiplicative reduction rounds.

The subtractive reduction rounds go in the series $P, P - p_d, ..., P - np_d$ and end when the dropping probability $P - np_d$ goes below $2 * p_d$. Roughly, this gives us

$$n \geq \begin{cases} 0 & \text{if } P <= 2 * p_d \\ \frac{P}{p_d} - 2 & \text{otherwise} \end{cases}$$

The multiplicative reduction of the flow would go in the series $P - np_d, \frac{P-np_d}{\beta}, \frac{P-np_d}{\beta^2}, \cdots, \frac{P-np_d}{\beta^m}$, where $\frac{P-np_d}{\beta^m} \leq P_{minThresh}$. This gives us

$$m \geq \frac{log(\frac{P-np_d}{P_{minThresh}})}{log(\beta)}$$

The time required for each round is l drop-list intervals; this is the minimum wait between two successive decrements. Taking the drop-list interval length from Equation 3, the total release time is

$$t_{release} = \frac{(m+n)lRK}{\sqrt{1.5p}M} \quad (9)$$

We use (β, p_d, $P_{minThresh}$, l) = (2, 0.05, 0.005, 3) for general monitored flows. For flows identified as unresponsive, we use (β, p_d, $P_{minThresh}$, l) = (1.5, 0.05, 0.0025, 5), which makes the release slower for unresponsive flows.

E Additional Simulations

This section describes additional simulations done to evaluate RED-PD.

E.1 Received Throughput

The simulations in this section explore the control RED-PD would wield on monitored flows in the presence of a persistent ambient drop rate at the output queue. In order to have a controlled environment, the output queue in these simulations is configured to drop each arriving packet with a fixed probability p, rather than as determined by RED dynamics. Each simulation has a single CBR source, which is started with a sending rate of $\gamma * f(R,p)$ pkts/sec, where R is RED-PD's target RTT (40 ms) and p is the fixed drop rate at the output queue.

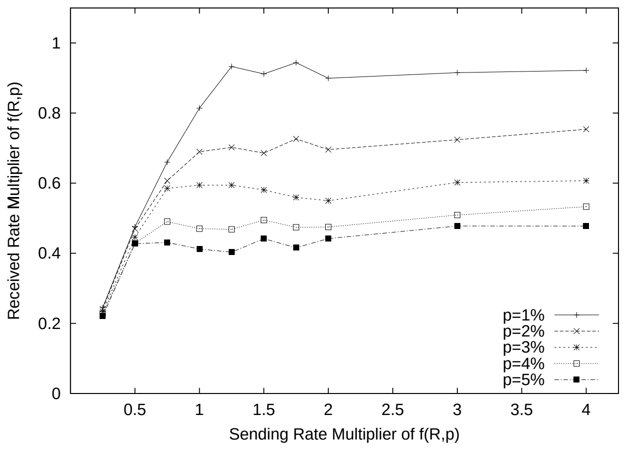

Figure 18: **Throughput of a CBR flow.**

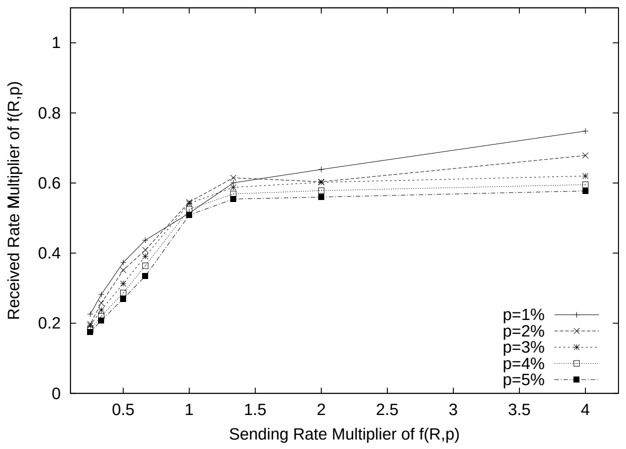

Figure 19: **Throughput of a TCP flow.**

The x-axis of Figure 18 gives the sending rate of the monitored flow, and the y-axis gives the flow's received throughput, both given as multiples of $f(R,p)$ pkts/sec. For example, the line labeled "p=1%" shows the throughput received by the CBR flow when the ambient packet drop rate is 1%. Figure 18 shows that RED-PD successfully controls the bandwidth received by aggressive flows.

The most striking feature in the graph is the reduction in bandwidth for the CBR flow as the ambient drop rate increases. With higher ambient drop rates, RED-PD increases

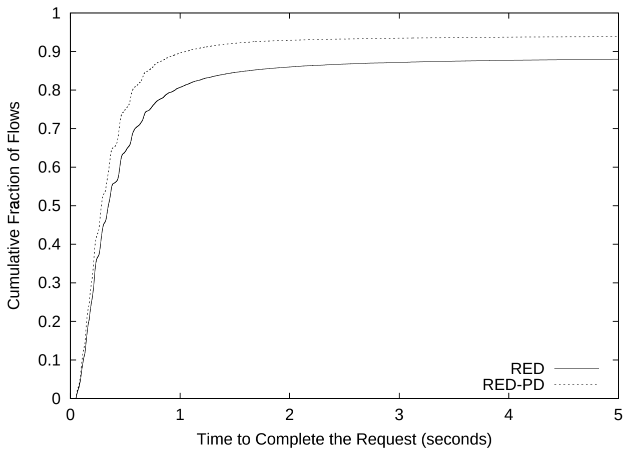

Figure 20: **Simulation with Web Traffic.** The cumulative fraction of requests which were completed before a given time.

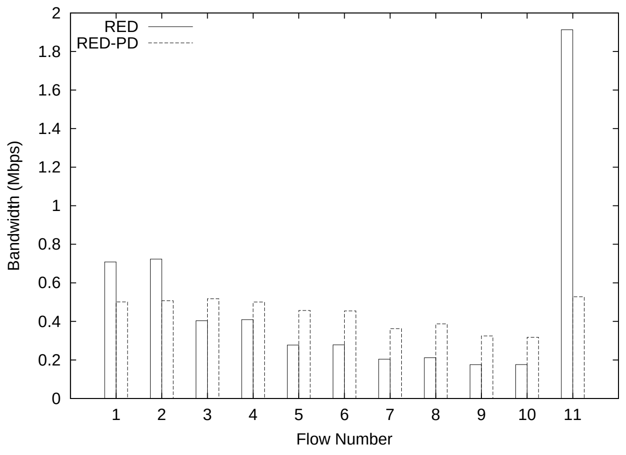

Figure 21: **Simulation with Web Traffic.** The bandwidth by the long flows. Flow numbers 1-10 are TCP flows, 2 each with RTT in ms of 20, 40, 60, 80 and 100. Flow 11 is a CBR flow sending at 2 Mbps.

the dropping probability of monitored flows more rapidly, resulting in a higher dropping probability in the pre-filter. Thus, at higher drop rates, RED-PD pushes down monitored flows below $f(R, p)$ in an attempt to reduce the ambient drop rate at the queue.

Figure 19 shows a similar graph for TCP traffic. It is interesting to note that with low drop rates, TCP flows get less bandwidth than CBR flows with roughly the same sending rate, while with high drop rates the TCP flows do better than the CBR flows.

E.2 Web Traffic

The simulations in Figure 20 show the effectiveness of RED-PD in a dynamic environment in the presence of web traffic (as represented by the web traffic generator in *ns*) .

The object size distribution for the web traffic generator is Pareto with average 24 packets and shape parameter 1.2, and the packet size is 500 bytes. The long term average of the generated web traffic is about 5 Mbps, roughly 50% of the link bandwidth. A dumbbell topology with 5 nodes on each side was used. The RTTs for flows on this topology ranged from 20 to 100 ms, in 20 ms increments. In addition to the web traffic, traffic included one CBR flow with a sending rate of 2 Mbps and ten infinite demand TCP flows, two of each RTT.

Two simulations were run, one with and one without RED-PD. Figure 20 shows the cumulative fraction of web requests completed by a given time. The use of RED-PD results in more bandwidth available for the web traffic, in spite of the fact that short-RTT TCP flows carrying web traffic are also occasionally monitored (if they last sufficiently long to be identified). By monitoring the CBR flow and short-RTT TCP flows, RED-PD reduces the ambient drop rate, which does a lot of good for other traffic. Figure 21 shows the bandwidth obtained by each of the infinite-demand flows. Apart from the CBR flow, RED-PD also reduces the bandwidth obtained by the 20-ms TCP flows (1 and 2), as their RTT is less than the target R of 40ms.

E.3 Other Congestion Control Models

In this section we explore RED-PD's impact on congestion control models other than TCP. We use TFRC [FHPW00], a TCP-Friendly Rate Control mechanism that is less aggressive than TCP in its rate increases, and also responds to congestion more slowly. The broad conclusion from these simulations is that RED-PD functions well with both TCP and with TFRC, and does not adversely affect the relative fairness of TCP and TFRC.

TFRC is a rate-based protocol which attempts to smooth the sending rate while maintaining the same long term sending rate as TCP, as given by the TCP equation in [PFTK98]. Instead of halving its sending rate in response to each congestion indication, TFRC estimates the average loss rate, and adapts its sending rate accordingly. To maintain a smoother sending rate, TFRC responds more slowly to congestion than TCP. This raises the questions of interactions between RED-PD and TFRC, which we explore here.

For each simulation set we started $4 * n$ sources, for n ranging from 2 to 8. We used four traffic types, TCP and TFRC with an RTT of 30 ms, and TCP and TFRC with an RTT of 120 ms, and each simulation had n flows of each type. Each

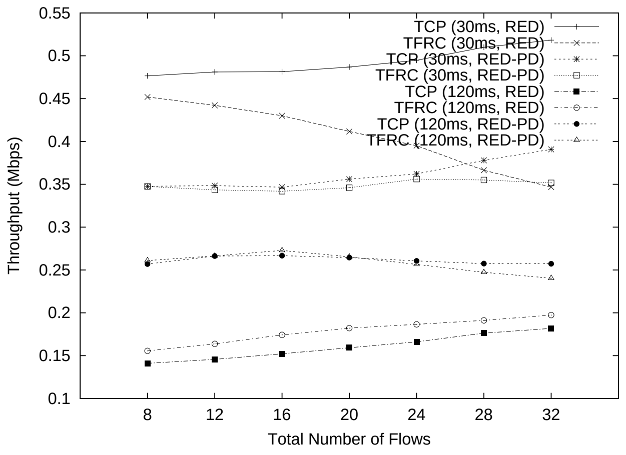

Figure 22: **The throughput of TCP and TFRC flows.** The graph plots the total throughput received by the four traffic types, both with and without RED-PD.

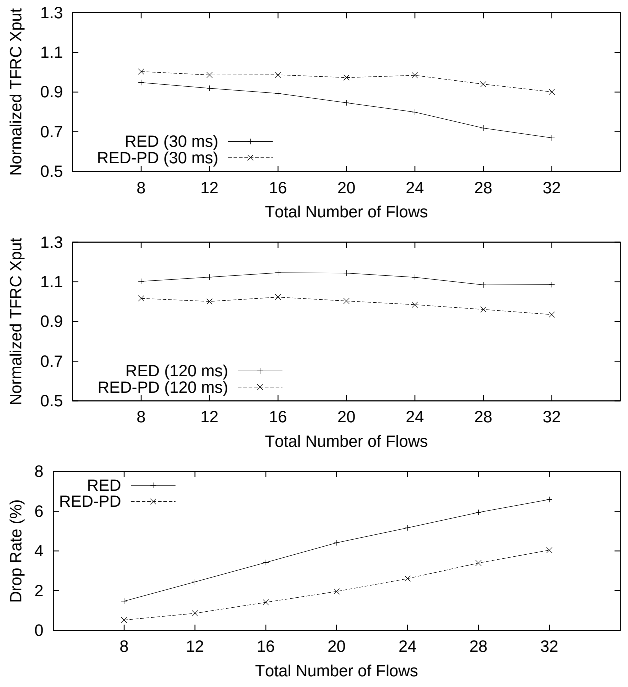

Figure 23: **TFRC performance compared with TCP.** The top graph plots the throughput of the 30-ms TFRC, normalized w.r.t. the throughput of the 30-ms TCP. The middle graph plots the same for 120-ms TFRC. The bottom graph shows the drop rate for each simulation.

simulation set was run with and without RED-PD. Figure 22 shows the throughput received by four traffic types, averaged over five simulation sets, as a function of the number of flows ($4n$). Figure 22 shows that RED-PD significantly increases the bandwidth available for the 120-ms TCP and TFRC flows, while restricting the bandwidth available to the 30-ms flows.

Figure 23 gives the same data as in Figure 22, in a different aspect. It shows the average TFRC throughput normalized with respect to the average throughput of the TCP flow with the same RTT. It is evident that RED-PD is fairer than plain RED in its bandwidth allocation for both the monitored and unmonitored flows.

E.4 Multiple Congested Links

The simulations in this section explore the impact of RED-PD on flows traversing multiple congested links. Each congested link has a capacity of 10 Mbps. On each link eight TCP sources and two CBR sources were started, with round-trip times ranging from 20 to 80 ms. The CBR flows were each sending at 4 Mbps. We study the bandwidth obtained by a flow passing through all the congested links, as the number of congested links is increased. The flow passing through multiple congested links is either a CBR flow with sending rate of 1 Mbps or (in a separate simulation) a TCP flow with an RTT of 80 ms. The RTT of the TCP flow was kept the same irrespective of the number of congested links it passed over by adjusting the delay of the connecting node, to factor out a throughput decrease due to an increasing RTT.

Figure 24 shows the bandwidth obtained by the flow passing through multiple congested links, with and without RED-PD. The throughput for the multiple-links flow goes down as the

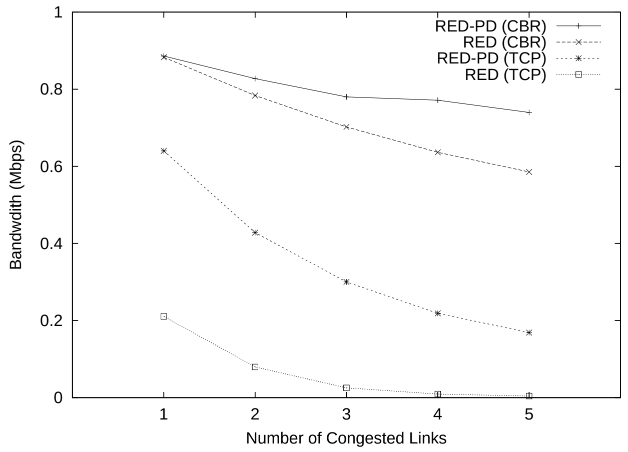

Figure 24: **Multiple Congested Links.** The graph shows the throughput of the CBR or the TCP flow which goes over all the congested links.

number of links increases, but is much better with RED-PD than with RED, because RED-PD decreases the ambient drop rate for each of the congested links. Unlike complete allocation schemes like FQ, RED-PD does not provide full max-

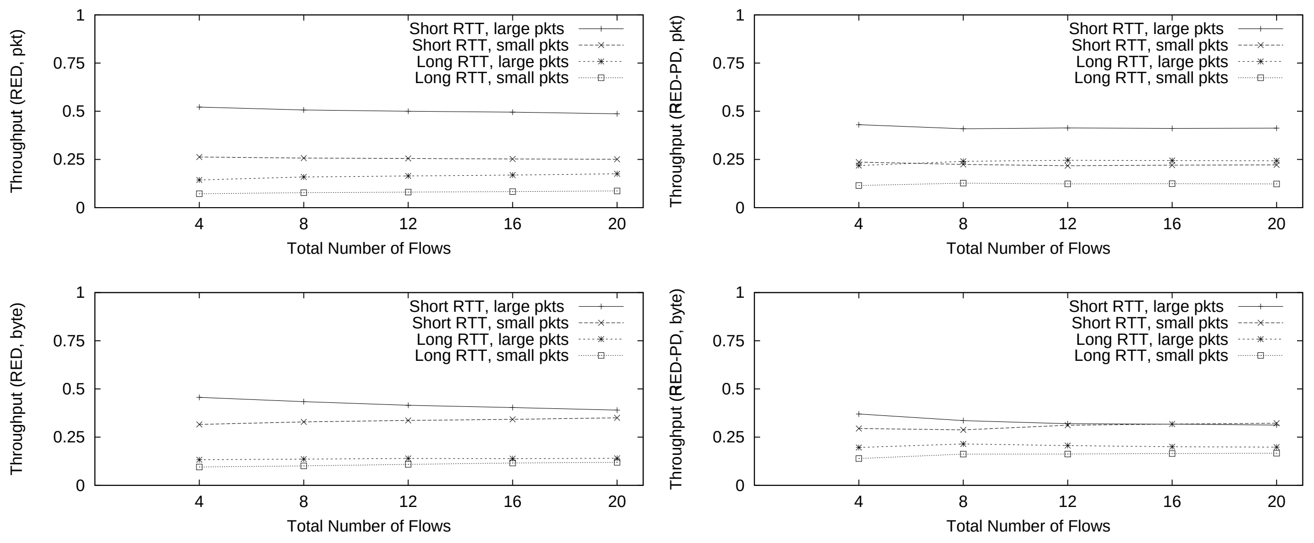

Figure 25: **RED-PD with RED in both packet and byte mode.**

min fairness. However, by controlling the high-bandwidth flows on each link, RED-PD brings the ambient drop rate down to manageable levels, and thus reduces the overall drop rate seen by flows traversing all congested links.

E.5 Byte Mode

So far, we have not addressed the difference between the sending rate in bytes/s and pkts/s for flows with different size packets. There is no consensus in the networking community about whether fairness between flows should be in terms of packets or in terms of bytes, but a queue management system should be able to operate in both modes. Because RED-PD uses identification-based preferential dropping, the biases of RED-PD in terms of packet size are largely determined by the biases of the underlying Active Queue Management (AQM) mechanism.

If the underlying AQM mechanism is in packet mode, and drops each packet with the same drop probability regardless of the packet size in bytes, then RED-PD's treatment of a flow will be based on its sending rate in packets per second. In contrast, if the underlying AQM is run in byte mode, where the dropping probability for an individual packet is a function of the packet size in bytes, then a flow's packets are marked in proportion to its arrival rate in bytes/sec, rather than in proportion to its arrival rate in packets/sec. In this case, the probability that a flow is monitored is a function of its sending rate in bytes/sec.

We used simulations to illustrate RED in both packet and byte mode. The TCP flows had two different round-trip times, 20 and 80 ms, and two different packet sizes, 1000 bytes (large packets) and 500 bytes (small packets). In each simulation $4 * n$ flows were started, n for each combination of round-trip time and packet size. The graphs in Figure 25 show the results with n ranging from 1 to 5. They plot the fraction of the throughput (in bytes/sec) received by each traffic type. The upper left graph shows simulations with RED in packet mode, and the lower left graph shows simulations with RED in byte mode. As we would expect, for RED in packet mode the short flows received higher throuphput that the long flows, and within flows of the same round-trip time, the large-packet flows receive higher throughput that the small-packet flows. When RED is run in byte mode, the throughput difference between the large-packet and the small-packet flows is diminished considerably, as each flow receives drops in proportion to its sending rate in bytes/sec.

A preferential dropping mechanism based explicitly on a computed $target\_rate$ in either bytes/s or in pkts/s could give the preferential dropping mechanism more control in terms of flows with different packet sizes. Because RED-PD uses the packet drop history to detect high-bandwidth flows, RED-PD's identification of high-bandwidth flows depends on the packet-dropping decisions of the underlying active queue management mechanism. The upper right graph shows RED-PD's performance for RED in packet mode, and the lower right graph shows RED-PD's performance for RED in byte mode. Both the upper and lower right graphs show that with RED-PD, the short-RTT large-packet flows receives less bandwidth, and the long-RTT flows receive significantly more bandwidth. However, when the underlying active queue management is run in byte mode, both of the short-RTT flows are high-bandwidth flows, and have their throuphput restricted by RED-PD. Thus, to first order, RED-PD inherits the biases (in terms of bytes/sec or pkts/sec) of the underlying active queue management.