Active Queue Management for Web Traffic

Matt Hartling, Mark Claypool, Robert Kinicki
{hartling|claypool|rek}@cs.wpi.edu

Computer Science Department
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609

May 22, 2002

Abstract

The focus in Active Queue Management (AQM) research has been on FTP-type traffic that is long-lived, insensitive to delay, and often has large TCP congestion windows. However, the majority of the flows on the Internet today are from Web traffic, which is short-lived, more sensitive to delay, and typically has small TCP congestion windows. In particular, short-lived flows with small congestion windows frequently suffer from timeouts when encountering packet loss. AQM approaches to date fail to adequately protect short-lived flows, resulting in relatively low benefits from AQM techniques for more realistic traffic mixes of both long- and long-lived flows. This paper presents and evaluates a new AQM scheme, SHort-lived flow friendly RED (SHRED), targeted at providing better network performance for short-lived Web traffic. Using an edge hint to indicate the congestion window size in each packet provided by the flow source or by an edge router, SHRED preferentially drops packets from short-lived Web flows less often than packets from long-lived flows. We present a wide-range of simulation results and analysis that demonstrate that SHRED protects short-lived flows from unnecessary timeouts, improving the response time characteristics for Web traffic when the router becomes congested, while not degrading overall network performance.

1 Introduction

Since the 1980's, Internet architects have addressed the problem of congestion control by modifying the TCP protocol and investigating queuing schemes at bottlenecked routers. While mechanisms such as slow-start, fast retransmit, and fast recovery were added to TCP, subnet routers employed Drop Tail and Random Early Detection (RED) queuing to avoid internal congestion and provide good performance. However, this combination of source mechanisms and queuing algorithms at internal routers are unable to meet the QoS requirements of today's applications.

The continually growing World Wide Web is significantly changing the nature and characteristics of Internet traffic. An Internet traditionally made up of FTP and email applications is being overwhelmed by newer, more-sophisticated and more-demanding Web-based applications[1] that require a range of network services, from small object transfers such as advertisements and buttons, to retrieval of streamed multimedia. Whereas applications such as FTP bulk transfers are concerned with throughput and packet loss rates, applications such as Web browsers that frequently request many short downloads care about delay and overall response time as well as throughput. This paper focuses on improving the performance of Web traffic.

Active Queue Management (AQM) research has focused on evaluating the benefits of congestion control algorithms by analyzing the performance of traffic mixes dominated by long-lived TCP flows. Most of the improvements and variations proposed for RED [11] have been evaluated using predominately long-lived flows. However, [4] showed that RED provides little benefit when traffic consists solely of short-lived Web flows, and under highly congested circumstances RED can perform worse than Drop Tail. This paper proposes an AQM solution based on RED to improve Web traffic performance.

A typical Web page is composed of text, pictures, advertisements and other embedded Web objects, where each Web object is transfered over its own TCP connection. Since an entire Web page is downloaded using multiple concurrent TCP connections, each TCP connection tends to be a short-lived flow. Transmitting only a small number of packets, these short-lived flows spend most of their lifetime in TCP's slow-start phase with small TCP window sizes. In contrast, long-lived flows transmit a larger number of packets predominantly during TCP's congestion avoidance phase with larger TCP window sizes.

Small TCP window sizes can significantly affect short-lived flows in several ways. First, a small TCP window yields a lower transmission rate which increases delay. Second, short-lived flows are more sensitive to packet drops because small windows increase the likelihood that a packet drop will result in a retransmission timeout that adds significant delay to the transfer of a small object. Finally, TCP's fast retransmit is ineffective at preventing retransmission timeouts when a packet is dropped at the beginning and end of the transmission.

To address the problems with Web traffic, this paper proposes a SHort-lived flow friendly variant of RED, called *SHRED*, that provides preferential dropping for flows with small windows. Sources mark each packet with their current window size, allowing SHRED to drop packets from flows with small TCP windows with a lower probability. This reduces the negative effects of small windows sizes, protects short-lived flows from low transmission rates, and provides fairer bandwidth allocation among flows.

This paper uses ns-2 simulations [19] to compare SHRED's performance to RED and Drop Tail under traffic scenarios that include Web traffic only, mixed Web and FTP traffic, and FTP

[1] Measurements on the MCI backbone [15] show that about 75% of traffic in terms of number of flows and number of bytes is HTTP.

traffic only. Additionally, we briefly discuss a Web traffic generator that was developed to facilitate properly analyzing SHRED with Web traffic.

Our analysis shows that: for Web only traffic, SHRED performs slightly better than Drop Tail for low to moderate congestion levels, whereas RED performs worse than Drop Tail; RED always performs better than Drop Tail for mixed traffic environments; and SHRED performs significantly better than RED and Drop Tail with mixed traffic and Web only traffic for moderate to high levels of congestion.

This paper is structured as follows: Section 2 reviews related research; Section 3 describes the SHRED architecture and algorithm details; Section 4 describes the experiments designed to evaluate SHRED, RED and Drop Tail over a range of traffic conditions; Section 5 analyzes the performance of SHRED and compares it to RED and Drop Tail; Section 6 discusses some issues to address in deploying SHRED; and Section 7 summarizes our conclusions and presentes possible future work.

2 Related Work

In this section, Section 2.1 provides a background of Random Early Detection, probably the most well-known AQM technique, that our technique, SHRED, extends (see Section 3). Section 2.2 describes work that applies RED and RED variants to Web traffic, comparable to analysis similar to that of this paper (see Section 5). Section 2.3 presents results from work that measure the characteristics of Web traffic, results which we use to construct a new Web traffic generator (see Section 4.1).

2.1 Random Early Detection

Random Early Detection [11] detects and reacts to incipient congestion before the router queue overflows. When the average queue size is above a minimum threshold (min_{th}) but below a maximum threshold (max_{th}), a packet is dropped with a probability between 0 and max_p using:

$$p_b = max_p(q_{avg} - min_{th})/(max_{th} - min_{th}) \quad (1)$$

When the average queue size exceeds max_{th}, the packet is dropped with probability 1. When the RED average queue size hovers close to max_{th}, RED drops many packets and performance degrades. [9] proposes gentle RED, a RED variant, where if the average queue size is above max_{th} but below $2 \times max_{th}$, the packet is dropped with a probability between max_p and 1. If the average queue size is above $2 \times max_{th}$, the packet is dropped with probability 1. As described in Section 3, SHRED extends gentle RED to provide preferential dropping for short-lived flows.

2.2 Random Early Detection and Web Traffic

Christiansen et al. [4] examine tuning RED parameters for Web traffic and demonstrate that RED provides little benefit over Drop Tail when the router forwards only Web traffic. The authors conclude there is little difference in performance between RED and Drop Tail for loads less than 90 percent. They show that above 90 percent load, only a carefully tuned RED router provides superior performance over a Drop Tail router and that a poorly-tuned RED router can negatively impact Web traffic. While our analysis presented in Section 5 also includes experiments with only Web traffic, we consider other traffic scenarios that include mixtures of Web traffic and long-lived FTP flows traveling through a congested router.

Guo and Matta [12] describe differences in characteristics and performance concerns between short-lived Web flows (mice) and long-lived flows (elephants). They propose a mechanism, called RIO-PS, whereby edge routers classify packets as short or long based on a per-flow packet count and preferentially drop packets based on the length classification. By maintaining separate RED parameters for both classes, RIO-PS favors the mice over the elephants with a much lower drop probability. Yilmaz and Matta [22] extend this class-based queuing approach by isolating short-lived, long-lived, and UDP flows into separate queues. A drawback of these approaches is that the ratio of service allocated to the classes is fixed regardless of the makeup of the traffic mix. As described in Section 3, SHRED uses edge hints to permit core routers to adapt to changes in the average window sizes of arriving flows to provide better treatment to mice without needing the fixed partition associated with class-based solutions.

2.3 Web Traffic Characteristics

A proper model for generating short-lived Web flows is important to evaluate new AQM algorithms. Typically, the majority of traffic in a Web client-server exchange is from the server sending requested objects back to the client. Thus, the critical step, in terms of evaluating AQM techniques, in developing a realistic Web traffic model is to properly reflect the flow characteristics between server and clients.

Mah [17] develops an empirical model of HTTP/1.0 traffic by analyzing packet traces on a university LAN. He reports the mean object reply size between 8-10 KB, the median reply sizes around 1.5-2.0 KB and the maximum reply size over 1 MB. Mah claims that Web reply size distributions are heavy-tailed and can be modeled effectively using a Pareto distribution with $1.04 < \alpha < 1.14$. Barford and Crovella [2] and Crovella and Bestavros [5] also study the characteristics of Web-traffic and Web-traffic generation. Their measurements show median reply sizes between 3-4 KB. They show that the reply size distribution tail can be modeled using a Pareto distribution with $\alpha = 1.06$ and that the reply size distribution body can be modeled using a log-normal distribution with $\mu = 9.357$ and $\sigma = 1.318$. Guo and Matta [12] use a bounded Pareto function with $\alpha = 1.2$ to generate Web replies. As described in Section 4.1, the Web traffic generator developed for our

investigation combines the model in [17] with results from [2], [5], and [12].

3 SHRED

This section presents background issues with short-lived flows in Section 3.1 and details on the SHRED architecture in Section 3.2.

3.1 Short-lived Flow Issues

For both congestion and flow control, TCP uses a congestion window (*cwnd*) to limit a flow's sending rate. TCP initializes the value of *cwnd* to one at the beginning of the connection and, each round-trip time (RTT) either doubles cwnd (during slow start) or increases cwnd by one (during congestion avoidance). However, since short-lived flows send a only a few packets for each connection, they typically have only small cwnds. Therefore, TCP flows with small cwnds receive a lower effective data rate than TCP flows with larger cwnds since the smaller window-sized flows require more RTTs to send and acknowledge packets.

A second problem with short-lived TCP flows is that fast retransmit is ineffective when the cwnd is small. Fast retransmit requires the receipt of three duplicate acknowledgments (acks) by the TCP sender to trigger retransmission of a lost packet instead of waiting for a full retransmission timeout period (RTO). TCP flows with a small cwnds, such as during the first three data packets, are unable to send enough packets to generate three duplicate acks. Moreover, loss of the first packet (a SYN packet) costs small cwnd flows even more time since the initial retransmission timeout (ITO) is typically conservatively set to three seconds.

A third problem with short-lived TCP flows occurs at the end of a TCP connection. If one of the last three packets is dropped, the TCP source does not send enough additional packets to trigger three duplicate acks and an RTO occurs. Typically, the last three packets consist of the last two data packets and the TCP FIN packet. Therefore, for small Web objects (i.e., 5 KB objects for a 1 KB packet size or 7.5 KB objects for a 1.5 KB packet size) all packet drops at a congested router will result in a costly RTO. Even for larger objects (i.e, 10 KB objects for a 1 KB packet size), many dropped packets result in an RTO. For example, half of all randomly dropped packets on result in an RTO for 10 KB objects (assuming a 1 KB packet size).

Overall, short-lived Web flows are more likely to suffer significant performance degradation when their packets are dropped than are long-lived flows. It is this inherent disadvantage to short-lived flows that SHRED attempts to reduce.

3.2 SHRED Algorithm

The fundamental idea of SHRED is to use a lower drop probability for flows with small cwnds and have the drop probability increase linearly with a flow's increased relative cwnd size. As in the core-edge architecture presented in [21], an edge hint provides flow information, such as sending

rate, to core routers to more efficiently manage congestion. Edge hints reduce the need for per flow state information in core routers. With SHRED, the sending host or the edge router inserts the the current value of cwnd, a TCP state variable, into the IP packet header. The SHRED core router extracts the cwnd value ($cwnd_{sample}$) from the arriving packet and computes an exponentially weighted average cwnd ($cwnd_{avg}$) as shown:

$$cwnd_{avg} = (1 - w_c)cwnd_{avg} + (w_c)cwnd_{sample} \tag{2}$$

where w_c is set to 0.002, the same setting as originally recommended for RED's w_q [8].

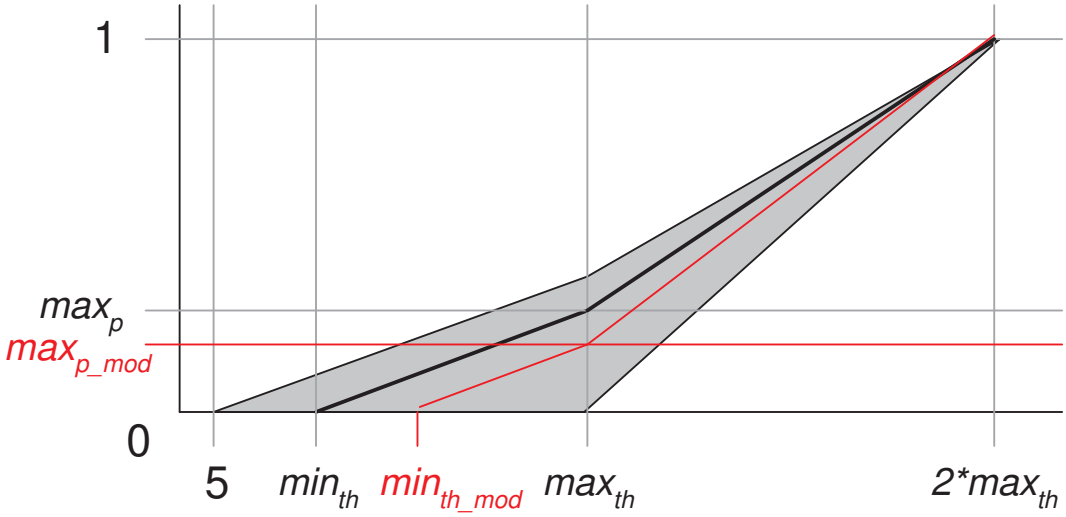


Figure 1: Computing Drop Probability in SHRED.

In SHRED, the packet drop probability is adjusted from that of gentle RED (see Section 2.1) based on a ratio of $cwnd_{sample}$ to $cwnd_{avg}$. As depicted in Figure 1, based on this cwnd ratio, SHRED modifies min_{th} and max_p such that the slope of the drop probability line remains the same for a given set of RED parameters:

$$min_{th-mod} = min_{th} + (max_{th} - min_{th}) * (1 - cwnd_{sample}/cwnd_{avg}) \tag{3}$$

If the cwnd ratio is equal to 1, min_{th-mod} remains at min_{th}. If the cwnd ratio is greater than 1, min_{th-mod} is set lower than min_{th}. If the cwnd ratio is less than 1, min_{th-mod} is set to a value between min_{th} and max_{th}. Based on recommendations [8], the lower bound for min_{th-mod} is set at five packets. The upper bound for min_{th-mod} is asymptotic to max_{th}.

Once min_{th-mod} is adjusted, max_p is modified so as to maintain the slope of the original RED drop probability line:

$$max_{p-mod} = max_p * (max_{th} - min_{th-mod})/(max_{th} - min_{th}) \tag{4}$$

Given min_{th-mod} and max_{p-mod}, SHRED computes the packet drop probability:

$$p_b = max_{p-mod}(q_{avg} - min_{th-mod})/(max_{th} - min_{th-mod}) \quad (5)$$

with the remainder of the algorithm implementation identical to RED.

Figure 2 depicts the cumulative distributions of cwnd sizes for packets dropped in SHRED, RED, and Drop Tail. Flows with small cwnds experience very few packet drops with SHRED, while flows with small cwnds are dropped substantially more in RED and Drop Tail. Since SHRED drops fewer packets with small cwnds, it drops more packets with larger cwnds, and, correspondingly, SHRED's drop CDF crosses that of Drop Tail at approximately 15 packets.

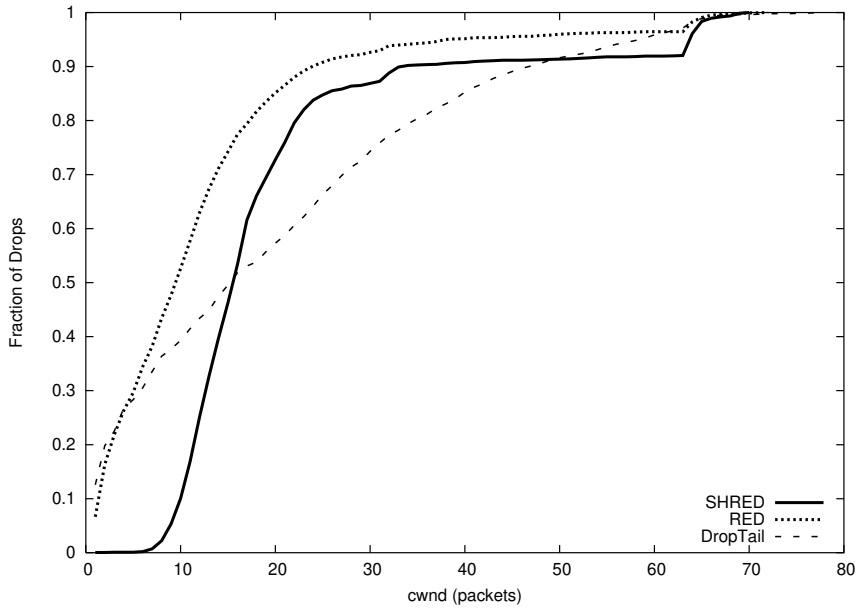


Figure 2: Cumulative Density Functions of Window Sizes for Dropped Packets.

4 Experiments

This section discusses the experimental methodology used to evaluate SHRED, Drop Tail and RED. Section 4.1 describes our Web traffic generator we developed in ns-2 [19] to properly analyze AQM architectures. Section 4.2 describes our performance evaluation metrics, and Section 4.3 explains our experimental setup.

4.1 Web Traffic Generation

Typically, Web traffic is simulated using one or more Web servers and a fixed number of Web clients. Web clients generate load on a router from empirically or mathematically derived parameters that

may include: reply size, response size, think time, inter-object arrival time, and the number of objects per Web page.

When the number of Web clients stays fixed, increased router congestion causes increased Web response times. This, in turn, reduces the client's request rate which effectively decreases the load generated by individual clients and the total network load seen by the router. While this accurately mimics the behavior of individual Web clients, it does not necessarily provide an accurate network representation from the router's perspective. Many Web servers, especially those from a content distribution network, serve only part of a Web page and do not receive subsequent requests from the same client. Clients often retrieve only one or two pages from a Web server before going to another Web site and so do not generate subsequent requests to the same server. Moreover, during flash crowds, the rate of users initially connecting is not influenced by the response time. Since existing ns-2 Web traffic models all decrease their offered load during congestion, we developed a Web traffic model that produces a fixed offered load through the router, independent of the level of congestion.

Our traffic generator sends Web pages, composed of multiple Web objects, to a traffic sink. We assume an HTTP/1.0 model [3] where each object in the Web page is a separate TCP connection.[2] The page generation rate is determined based on the previous page size and the aggregate load, in bits per second, specified by the user. The Web traffic generator sends all objects in a Web page and then waits for an amount of time determined by the page generation rate before sending the next page.

In the standard HTTP procedure, the client first requests the primary container page and then subsequently issues separate requests to transfer each embedded object. Since the model constructed for this research is a best case scenario in terms of Web page response time in that the primary object and all secondary objects are downloaded concurrently, actual response time differences due to AQM strategies (see Section 5) should be more dramatic than the simulation results presented here.

Building upon existing ns-2 Web traffic generators, our Web traffic uses the built in Pareto II function to generate Web reply object sizes. Based on previous findings in [17] and [12], we use 1.2 as the Pareto shape parameter with 10 KBytes as the average size parameter. The minimum and maximum object sizes are set to 12 Bytes and 2 MBytes, respectively. All the simulations used for the experiments use 1 object per page unless otherwise noted. In Section 5.5, separate simulations were run with the number of objects set to 1, 1-8 objects, 1-16 objects, and 1-32 objects to evaluate the impact of number of objects per page on response time.

[2]While HTTP/1.1 [20] enables multiple objects to be transferred over a single connection, in times of heavy server load, Web servers may close even persistent connections [16]. Moreover, [1] found nearly half of all TCP connections to a Web server use multiple connections.

4.2 Metrics

We use both user-centric measurements that allow us to study evaluate the impact of Drop Tail, RED and SHRED on Web traffic and network-centric measurements that allow us to evaluate the impact of the same AQM techniques on overall network dynamics and long-lived traffic.

For user-centric measurements, we define object *transmission time* as the time taken to transfer a single Web object from a server to the client and Web *response time* as the time taken to download all objects in a Web page. While users certainly desire low response times, users also desire low object transmission times since this indirectly results in a low response times. Moreover, typical Web browsers can display part of a page when complete objects are downloaded even if the entire page is composed of multiple objects. Routers should try to minimize object transmission time, not only to provide better end-user performance, but also to reduce the number of flows through the router.

For network-centric measurements, we evaluate the percentage of packets dropped per flow for both Web and FTP traffic and goodput and Jain's Fairness index [14] for FTP traffic.

4.3 Experiment Setup

The network configuration used for all the experiments in this investigation is shown in Figure 3. The base RED parameters were set according to the latest recommended values from [8] and [10]. Although ns-2 RED simulations often set queue thresholds based on packets, since many Web reply objects are smaller than a single packet, we used queue thresholds based on bytes. All the experiments were run for 160 seconds with measurements taken after the first 20 seconds for experiments using FTP traffic to allow long-lived flows to stabilize before performance data was collected. For all experiments, both the FTP traffic and the Web traffic used TCP Reno.

The first set of experiments, referred to as *Web-only*, analyzes AQM performance with only Web traffic generated, allowing evaluation similar to that in [4]. The second set of experiments, referred to as *Web-mixed*, analyzes AQM performance with both Web and long-lived traffic, a traffic mix more representative of the current Internet. MCI backbone measurements [15] show that 75% of the flows and bytes are Web, and Sprint backbone measurements [6] show 90% of the flows send fewer than 20 packets. Thus, in the second set of experiments, the number of FTP flows is fixed at 10 while the load attributed to Web-traffic is varied between 40 and 80 percent of the bottleneck bandwidth (10 Mbps). In third set of experiments, referred to as *FTP-mixed*, the Web traffic load is fixed at 50 percent while the number of FTP flows is varied between 0 and 40. The fourth set of experiments, referred to as *FTP-only*, analyzes AQM performance when all traffic is long-lived, with the number of FTP flows varied between 10 to 100.

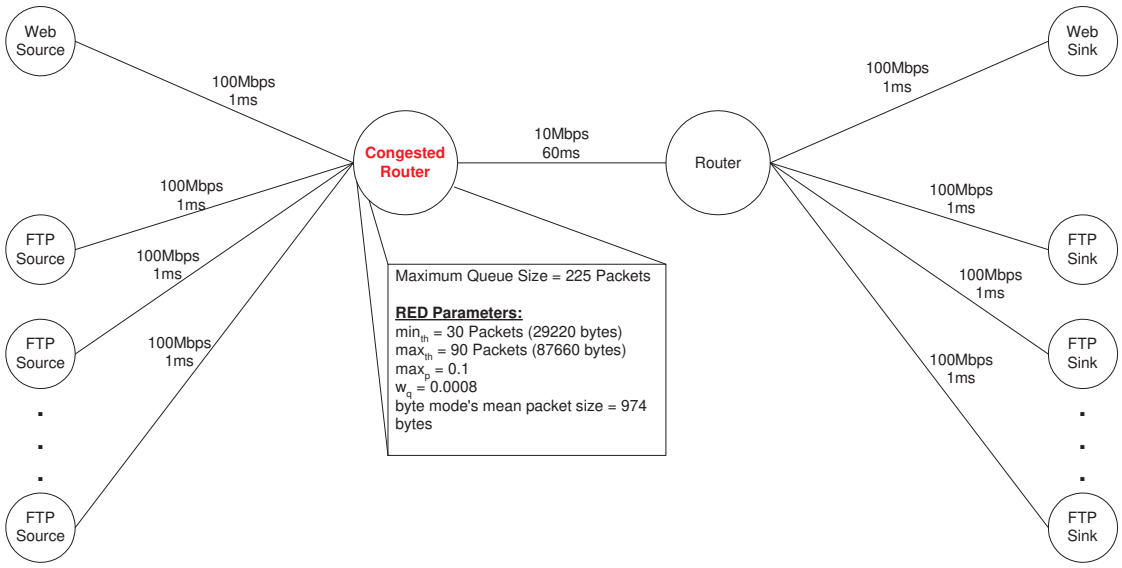


Figure 3: Network Configuration.

5 Analysis

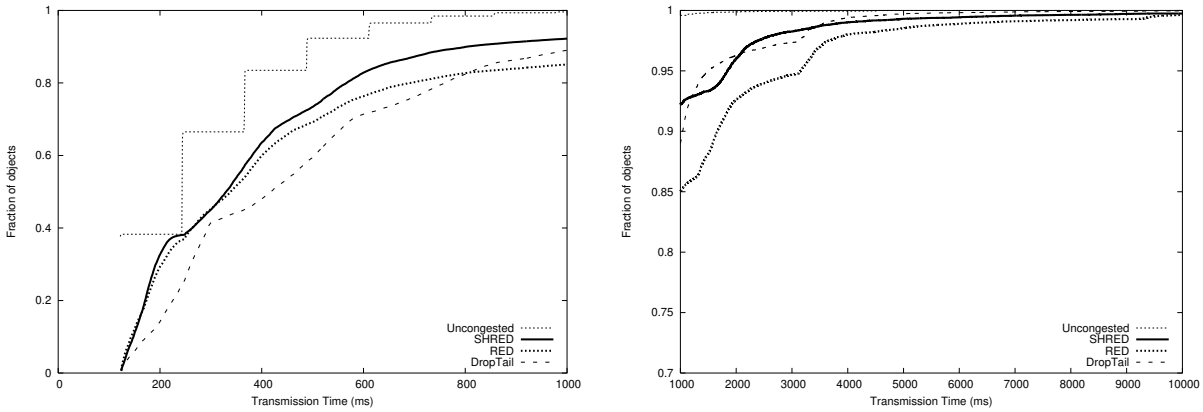
This section compares the performance of SHRED to Drop Tail and RED by measuring object transmission time and drop percentage for Web traffic, and goodput and Jain's Fairness for FTP traffic. Detailed results are presented for Web-only (Section 5.1), Web-mixed (Section 5.2) and FTP-only (Section 5.4) experiments. Due to space constraints, only summary results are presented for FTP-mixed (Section 5.3) experiments. Lastly, Web response times for pages with a range of objects (Section 5.5) are also analyzed.

5.1 Web-only Experiments

To provide an upper bound on Web traffic performance, we first measured object transmission times for an uncongested router. The bottleneck link capacity was set to 100 Mbps to ensure little queuing delay and no packet drops. The cumulative density function of transfer times for the uncongested router has "steps", (as seen in Figure 4 and subsequent Figures) caused by an initial window size of 1 and the slow-start phase of TCP. Specifically, an object small enough to fit in a single packet is transmitted in one round-trip time (RTT); an object as large as two or three packets is transmitted in two RTTs; and an object as large as four to seven packets is transmitted in three RTTs. This effect continues throughout the slow-start phase.

Figures 4 through 6 present cumulative density functions (CDFs) of transmission times for Web loads varying from 95 to 105 percent. Each CDF is split into a body displaying the CDF of transmission times up to 1 sec and a tail displaying the CDF of transmission times between 1 and 10 seconds. Each body-tail pair of graphs is for a single load, where a 95 percent load is 9.5 Mbps

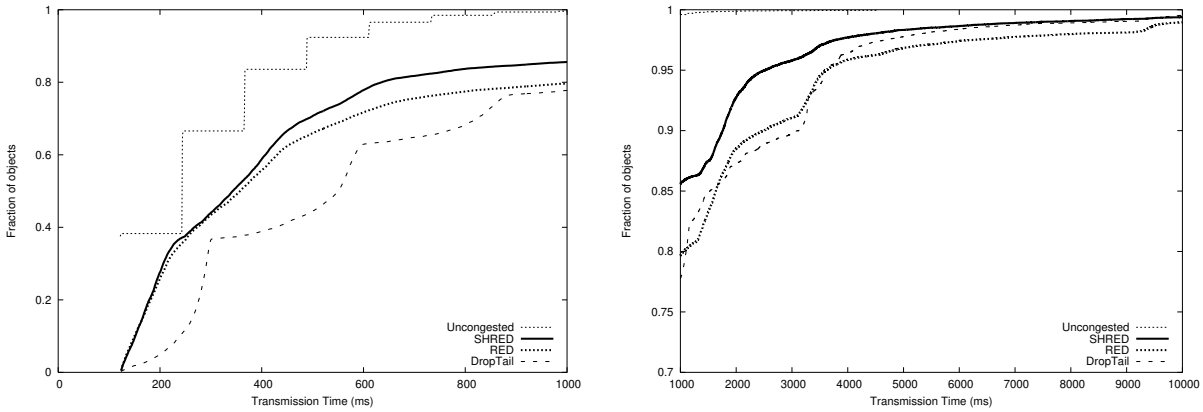
of a 10 Mbps link, for example.



(a) Body (b) Tail

Figure 4: Transmission Time CDF for 95 percent load (average of 81 active flows)

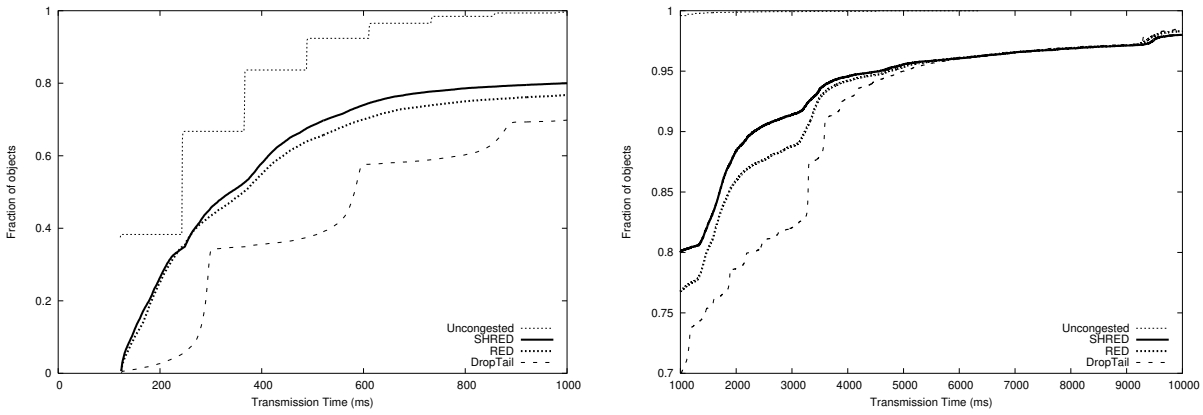
At 95 percent Web-only load, shown in Figure 4(a), at 1 second, SHRED performs slightly better than both Drop Tail and RED, and RED performs slightly worse than Drop Tail. However, the tail of the CDF in Figure 4(b) implies that Drop Tail performs slightly better than SHRED and better than Drop Tail by having fewer very high transfer times. Note the "step in the CDFs in Figure 4(b) between one and two seconds for both RED and SHRED, attributable to TCP retransmission time-outs (RTOs). The subsequent step in the RED CDF, and to some extent the Drop Tail CDF, after three seconds is due to initial time-outs (ITOs) that occur from dropping of the first packet in many flows.



(a) Body (b) Tail

Figure 5: Transmission Time CDF for 100 percent load (average of 140 active flows)

At 100 percent Web-only load, shown in Figure 5, the performance of Drop Tail begins to degrade as the queue is frequently full and the queuing delay for all packets becomes large. With a queue size of 225 packets, a packet takes approximately 175 ms to traverse the queue. Adding this queuing delay to the 120 ms RTT: objects consisting of a single packet take roughly 300 ms to transfer; objects of two or three packets in length take 600 ms to transfer; and objects between 4 and 7 packets in length take 900 ms to transfer. Furthermore, packet drop bursts induced by Drop Tail make it difficult for the small-windowed TCP flows to employ fast retransmit. At 100 percent load, RED performs slightly better than Drop Tail, while SHRED performs better than both RED and Drop Tail.



(a) Body (b) Tail

Figure 6: Transmission Time CDF for 105 percent load (average of 218 active flows)

At 105 percent Web-only load, shown in Figure 6, SHRED again performs better than both RED and Drop Tail. For such very high congestion levels, RED consistently outperforms Drop Tail.

Since general computer response times beyond one second disrupt the user's flow of thought [18], we summarize performance of RED and SHRED to that of traditional Drop Tail queuing by considering the fraction of transmission times that complete in one second or less. We compute the transmission time *benefit* by normalizing the fraction of RED and SHRED flows that complete in 1 second or less to the fraction of Drop Tail flows that complete in 1 second or less. Figure 7 summarizes the performance benefit of RED and SHRED for the Web-only experiments over a load range from 70% to 120%. RED and SHRED provide little benefit at low congestion levels. SHRED provides significant benefit to Drop Tail for moderate to high levels of congestion and always provides more benefit than does RED. For moderate to high levels of congestion, RED performs worse than Drop Tail, confirming results in [4] that concluded RED does not provide benefit over Drop Tail for Web traffic. However, for very high congestion levels, RED does provide significant benefit

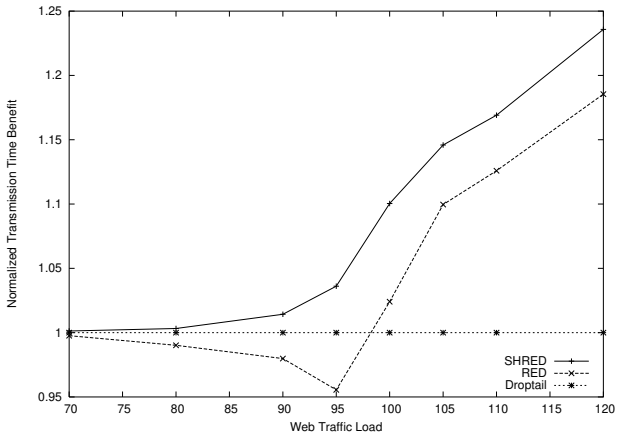


Figure 7: Normalized Transmission Time Benefit for Web-only

to Web traffic.

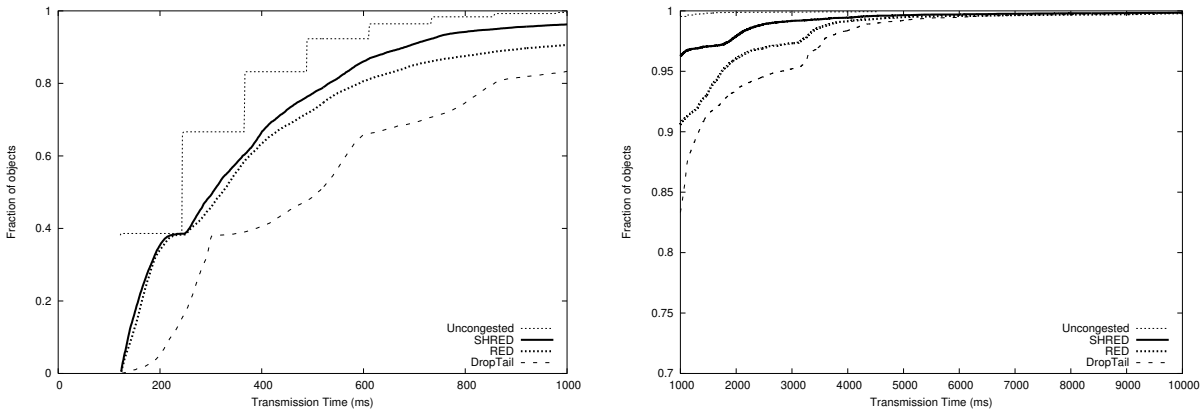
5.2 Web-mixed Experiments

The second set of SHRED experiments includes a more realistic traffic mix of long-lived and short-lived flows. Figures 8 through 10 show the performance of SHRED for the Web-mixed experiments for 60%, 70% and 80% Web load and 10 FTP flows.

For all Web-mixed loads, SHRED consistently provides a substantial performance improvement over both Drop Tail and RED. The SHRED CDF is much closer to uncongested router CDF than either RED or Drop Tail, especially for the 60% load. As noted in Section 3.1, a packet drop in a small-windowed flow does not allow a fast retransmit, causing either an RTO or an ITO. Analysis of the CDF tails suggest that at least 5 to 10 percent of RED's performance degradation is due to RTOs (between 1 and 2 seconds) and an additional 2 to 5 percent degradation is from ITOs. Drop Tail suffers a performance degradation of at least 5 to 8 percent due to RTOs and an additional 4 to 10 percent degradation from ITOs. SHRED largely avoids performance degradation due to ITOs and is less affected by RTOs than both RED and Drop Tail. Lastly, contrary to results in Section 5.1, RED performs consistently better than Drop Tail for all Web-mixed loads, a more realistic traffic environment.

Figure 11 summarizes the SHRED performance benefit over RED and Drop Tail for the mixed traffic experiments. SHRED provides a 12 to 17 percent performance improvement over Drop Tail and this advantage increases slightly as the Web traffic load increases. The RED improvement over Drop Tail remains fairly constant around 8 percent regardless of the load. This is contrary to the Web-only experiments where RED performed worse than Drop Tail in many situations.

Figure 12 shows the percent drops for nine of the Web-mixed experiments. Each bar in the graph separates out the Web traffic drops from FTP traffic drops for a given experiment. For each three column group, the first column is the SHRED drop percentage while the second and



(a) Body (b) Tail

Figure 8: Transmission Time CDF for 60 percent Web load (average of 61 active flows) and 10 FTP flows

third column are drop percentages for RED and Drop Tail, respectively. SHRED drops fewer Web packets than either RED or Drop Tail and drops packets more proportionally to the actual traffic mix. SHRED yields the fewest drops at all three load levels.

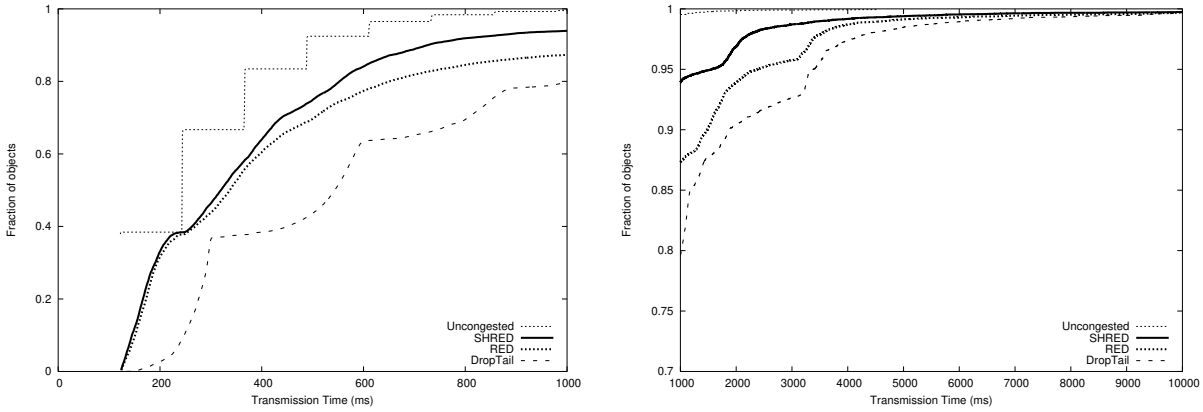
5.3 FTP-mixed Experiments

The results from the FTP-mixed experiments are similar to the results from the Web-mixed experiments. Due to space constraints, we only summarize these experiments as in Figure 7 in Section 5.1. Figure 13 summarizes the performance benefit of RED and SHRED over Drop Tail for the FTP-mixed experiments for 10 to 50 FTP flows and a Web load of 50%.

SHRED provides an increase in benefit versus Drop Tail with an increase in FTP load and SHRED performs significantly better than RED at higher levels of congestion. While RED performs consistently better than Drop Tail, the benefit of SHRED to Web traffic widens with an increase in the number of flows.

5.4 FTP-only Experiments

Although SHRED does not seek to improve performance of long-lived flows, it should not have a negative impact versus either Drop Tail or RED in the presence of only long-lived traffic. We ran experiments with only 10 to 100 FTP flows under RED, SHRED and Drop Tail, and compute the total goodput and Jain's fairness for all flows. Table 1 shows that SHRED's performance is similar to Drop Tail and slightly better than RED for most the experiments. Table 2 shows that SHRED yields a slightly higher Jain's Fairness value than both RED and Drop Tail, by providing a more uniform window size for all flows.



(a) Body

(b) Tail

Figure 9: Transmission Time CDF for 70 percent Web load (average of 84 active flows) and 10 FTP flows

Number of Flows	SHRED	RED	Drop Tail
10	9.4	9.6	10.0
20	9.7	9.7	10.0
30	9.8	9.8	9.9
40	9.7	9.7	9.9
50	9.7	9.7	9.8
60	9.6	9.6	9.8
70	9.5	9.6	9.7
80	9.5	9.3	9.6
90	9.4	8.7	9.6
100	9.3	8.4	9.5

Table 1: Goodput (Mbps)

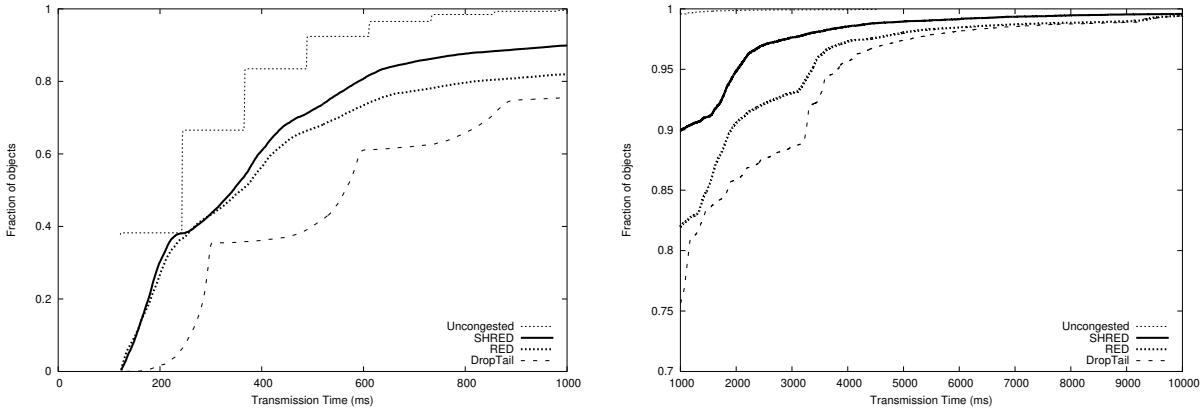
Number of Flows	SHRED	RED	Drop Tail
10	1.000	0.997	0.991
20	0.999	0.996	0.990
30	0.999	0.997	0.994
40	0.999	0.997	0.989
50	0.999	0.996	0.991
60	0.999	0.995	0.993
70	0.998	0.994	0.988
80	0.998	0.993	0.990
90	0.999	0.989	0.989
100	0.998	0.988	0.990

Table 2: Jain's Fairness

5.5 Web Response Times

Web response time is a more useful measure of performance for users than transmission time, since a Web page is not entirely usable until all the objects have been downloaded. While [12] consider between 1-3 and 2-7 objects per page, recent measurements [16] imply that today's Web pages have considerably more objects per page. Thus, the Web-only and Web-mixed experiments were re-run with a uniform distribution of 1 to 8, 1 to 16, and 1 to 32 objects per page.

Figures 14 summarizes the response time benefits of SHRED over Drop Tail for the Web-only experiments. At lower congestion levels, the number of objects per page does not significantly affect SHRED's benefit, with SHRED providing only slightly more benefit than Drop Tail. Between loads of 90 and 105 percent, SHRED provides significant response time benefits for Web pages with



(a) Body (b) Tail

Figure 10: Transmission Time CDF for 80 percent Web load (average of 121 active flows) and 10 FTP flows

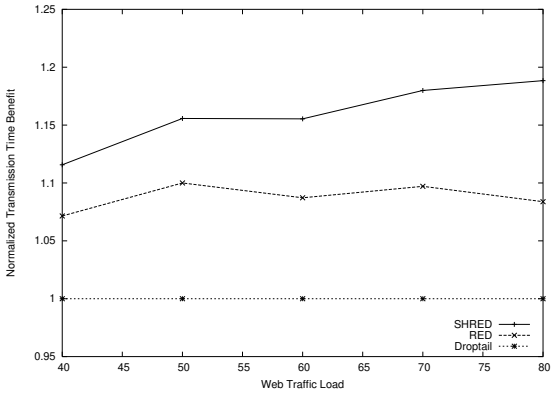


Figure 11: Normalized Transmission Time Benefit for Web-mixed

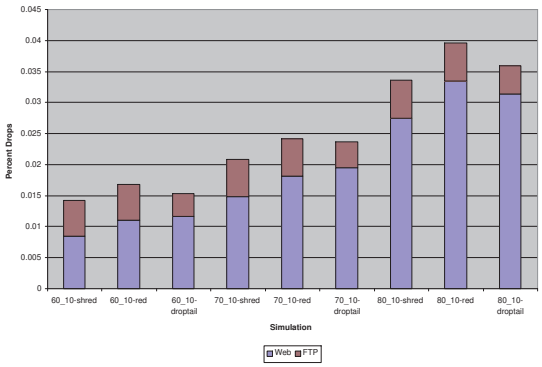


Figure 12: Percent Drops for Web-mixed

multiple objects. However, the benefit is similar for all pages with multiple objects. For higher congestion levels, response times for Web pages with more objects decreases slightly, most likely because when a large number of objects are transmitted concurrently, the queue receives a burst of traffic causing bursts of dropped packets.

Figure 15 summarizes the response time benefit of SHRED over Drop Tail for the Web-mixed experiments for 40% to 80% Web load and 10 FTP flows. Since our Web traffic generator transmits all objects concurrently, Drop Tail routers see traffic as a burst of packets, consisting of a large number of objects belonging to a page, that are more likely to overflow the router queue. Unlike the Web-only results, the response time benefit increases approximately linearly as the number of objects increases and remains constant over all loads except for the experiments with 1-32 objects per page. SHRED provides over 60 percent response time benefit over Drop Tail for Web pages with

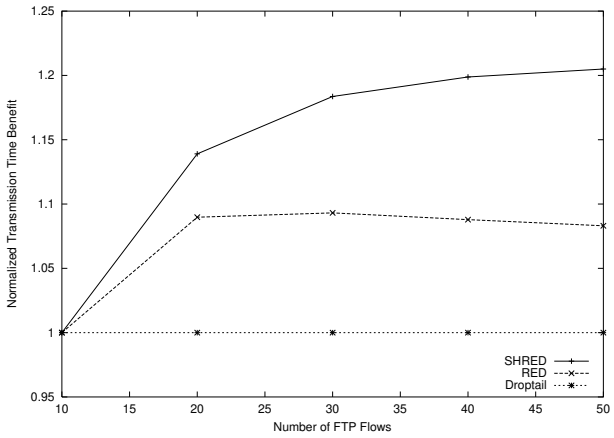


Figure 13: Normalized Transmission Time Benefit for FTP-mixed

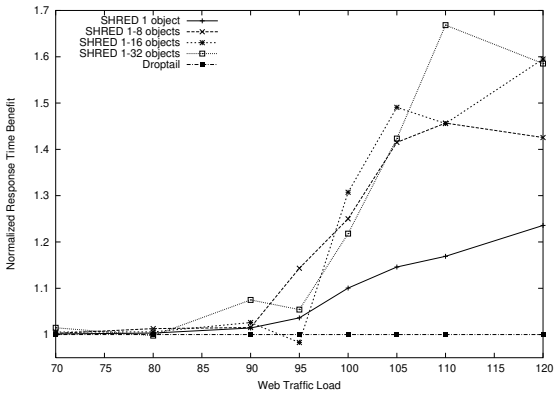


Figure 14: Web Response Time Benefit for Web-only

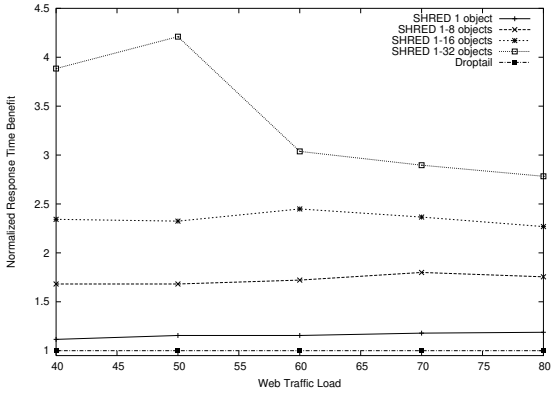


Figure 15: Web Response Time Benefit for Web-mixed

1-8 objects and over 140 percent response time benefit over Drop Tail for pages with 1-16 objects. For pages with 1-32 objects and loads of 40% or 50%, SHRED provides response time benefits greater than 250 percent. For higher loads, the SHRED response time benefit reduce somewhat to between 175 to 200 percent.

6 Discussion of Deployment

The research presented here is not intended for immediate deployment into the Internet, but rather is meant to propose and evaluate the benefits to Web traffic for preferentially dropping packets based on TCP window size. The benefits of SHRED can be realized by incorporating SHRED as one of the many facets in the next generation of Internet routers. In light of this, this section discuss practical deployment issues facing SHRED.

The cwnd edge hint can be implemented in IPv4 using the TOS byte in the IP header. The

TCP source maintains cwnd in bytes, but can map it to a value in packets using TCP's maximum segment size (or any mapping where a value of 1 yields the lowest drop probability and a value of 255 yields the highest). Using the TOS byte gives 255 possible values for cwnd. SHRED does not provide significant benefits for TCP flows with cwnds larger than 255 packets, and therefore any TCP sources with cwnds larger than 255 automatically set the cwnd hint to 255. For IPv6, the cwnd edge hint can be implemented using an extended header.

SHRED supports incremental deployment. Only users and Internet Service Providers (ISPs) seeking improved Web performance need to upgrade. If the TCP source does not support SHRED, the SHRED router uses normal RED dropping probabilities. Although there far more TCP sources than routers, source operating systems are easier to update since popular operating systems issue updates multiple times a year. An alternative to upgrading TCP in all end-host operating systems would be to place a packet classifier at an edge router that keeps per flow TCP state and inserts cwnd hints into each packet.

Unfortunately, using cwnd hints provides opportunities for misbehaving hosts to deliberately keep their cwnd hint low to get reduced drop rates from SHRED routers. Stagnant cwnd hints could be policed by the first downstream router from the source keeping per flow TCP state since all packets and TCP acknowledgments for the source must pass through that router.

Another possible source of misbehaving is between an ISP and a backbone provider, where the ISP could set low cwnd hints such that all packets receive a lower dropping probability, yielding better overall performance for network traffic for the ISP. However, as backbone router's are typically over-provisioned to avoid packet drops [13], such false hints would not significantly affect performance.

While SHRED does provide a substantial improvement in response time over Drop Tail in simulations with multiple objects per page (see Section 5.5), the actual benefit to SHRED may be lower since browser implementations typically establish a maximum of four parallel connections at a time [1]. Also, a Web page may still be usable to the user prior to all the objects being displayed by the browser (e.g., users don't need ad banners when viewing a Web page). Thus, the Web response times presented here may represent worst case times as far as a user is concerned.

7 Conclusions

The majority of the traffic on the Internet today is Web traffic, which is typically composed of many short-lived flows. Typical AQM research uses long-lived flows to evaluate performance and often ignores the issues with dropping packets from long-lived flows versus short-lived flows. In particular, short-lived flows often have small cwnds, making them sensitive to packet drops which cause retransmission timeouts and high response times.

This paper presents SHort-lived flow friendly RED (SHRED), an AQM mechanism designed to improve response time for short-lived Web traffic. Using a cwnd hint from a TCP source, SHRED

computes the cwnd ratio of an arriving packet to the cwnd average and reduces the probability of dropping packets during the sensitive periods when a flow's cwnd is small. We ran extensive simulation experiments modeling Web-only traffic, FTP-only traffic, and mixes of Web and FTP traffic to evaluate the performance of SHRED versus RED and Drop Tail routers.

Our analysis shows that SHRED produces lower object transmission times than either RED or Drop Tail in both Web-only and mixed traffic environments. SHRED provides over 40 percent improvement in Web response times over Drop Tail for Web-only traffic and between 70 and 300 percent improvement in Web response times over Drop Tail. Moreover the Web traffic performance improvements are achieved without negatively impacting long-lived FTP traffic.

In contrast to results reported in [4] that show little improvement for RED over Drop Tail in Web-only traffic experiments, our mixed traffic simulations indicate that RED consistently outperforms Drop Tail, although RED still performs consistently worse than SHRED.

Although SHRED protects short transfers and the beginning of longer transfers, there remains the issue of fast retransmit ineffectiveness at the end of the transfer that can cause retransmission timeouts. One possible extension to SHRED is to incorporate an indicator in the cwnd hint when a flow is nearly finished and to avoid dropping the last few packets of a transfer.

A second area of future work is integrating SHRED with Adaptive RED [10] and ECN [7]. SHRED combined with Adaptive RED should operate well over a wider range of traffic loads and SHRED with ECN should result in even fewer drops for Web traffic and should further improve Web response times.

The effects of AQM on Web traffic performance is dependent on number of Web traffic characteristics, including size of objects, number of objects on a page, number of persistent connections, number of objects transmitted in one persistent connection, browser's and server's support for persistent connections, number of objects per server for a page (when objects are distributed across multiple servers), Web cache performance, and user's behavior. For further improvements in the treatment of Web traffic, more data collection studies of Web traffic characteristics are needed. These results could also be used to improve on the Web traffic generator used here.

References

[1] M. Allman. A Web Server's View of the Transport Layer. *ACM Computer Communication Review*, 30(4), Oct. 2000.

[2] P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proceedings of Measurement and Modeling of Computer Systems*, pages 151–160, July 1998.

[3] T. Berners-Lee, R. Fielding, and H. Frystyck. Hypertext Transfer Protocol – HTTP/1.0. *IETF Request for Comments (RFC) 1945*, May 1996.

[4] M. Christiansen, K. Jeffay, D. Ott, and F. Smith. Tuning RED for Web Traffic. In *Proceedings of ACM SIGCOMM Conference*, Aug. 2000.

[5] M. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[6] C. Diot, G. Iannaccone, and M. May. Aggregate Traffic Performance with Active Queue Management and Drop from Tail. Technical Report TR01-ATL-012501, Sprint ATL, Jan. 2001.

[7] S. Floyd. TCP and Explicit Congestion Notification. *Computer Communication Review*, Oct. 1994.

[8] S. Floyd. Red: Discussions of setting parameters. URL: http://www.aciri.org/floyd/REDparameters.txt, Nov. 1997.

[9] S. Floyd. Recommendation on using the *gentle* variant of RED. http://www.icir.org/floyd/red/gentle.html, Mar. 2000.

[10] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED, 2001.

[11] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, Aug. 1993.

[12] L. Guo and I. Matta. The War Between Mice and Elephants. In *Proceedings of the 9th IEEE International Conference on Network Protocols*, Nov. 2001.

[13] G. Iannaccone, M. May, and C. Diot. Aggregate Traffic Performance with Active Queue Management and Drop from Tail. *ACM Computer Communication Review*, July 2001.

[14] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley and Sons, Inc., 1991.

[15] k claffy, G. Miller, and K. Thompson. the nature of the beast: recent traffic measurements from an internet backbone. In *Proceedings of ISOC INET '98*, July 1998.

[16] B. Krishnamurthy and C. E. Wills. Analyzing Factors that Influence End-to-End Web Performance. *WWW9 / Computer Networks*, 33(1-6):17–32, 2000.

[17] B. A. Mah. An Empirical Model of HTTP Network Traffic. In *Proceedings of INFOCOM*, pages 592–600, Apr. 1997.

[18] J. Nielsen. *Usability Engineering.* Morgan Kaufmann Publishers, Oct. 1994.

[19] U. of California Berkeley. The Network Simulator - ns-2. Interent site `http://www.isi.edu/nsnam/ns/`.

[20] R.Fielding, H. Frystyck, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. *IETF Request for Comments (RFC) 2616*, June 1999.

[21] I. Stoica, S. Shenker, and H. Zhang. *Core*-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *Proceedings of ACM SIGCOMM Conference*, Sept. 1998.

[22] S. Yilmaz and I. Matta. On Class-based Isolation of UDP, Short-lived and Long-lived TCP Flows. In *Proceedings of International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, Aug. 2001.