The War Between Mice and Elephants

LIANG GUO IBRAHIM MATTA
Computer Science Department
Boston University
Boston, MA 02215, USA
{guol, matta}@cs.bu.edu

Abstract

Recent measurement based studies reveal that most of the Internet connections are short in terms of the amount of traffic they carry (mice), while a small fraction of the connections are carrying a large portion of the traffic (elephants). A careful study of the TCP protocol shows that without help from an Active Queue Management (AQM) policy, short connections tend to lose to long connections in their competition for bandwidth. This is because short connections do not gain detailed knowledge of the network state, and therefore they are doomed to be less competitive due to the conservative nature of the TCP congestion control algorithm.

Inspired by the Differentiated Services (Diffserv) architecture, we propose to give preferential treatment to short connections inside the bottleneck queue, so that short connections experience less packet drop rate than long connections. This is done by employing the RIO (RED with In and Out) queue management policy which uses different drop functions for different classes of traffic.

Our simulation results show that: (1) in a highly loaded network, preferential treatment is necessary to provide short TCP connections with better response time and fairness without hurting the performance of long TCP connections; (2) the proposed scheme still delivers packets in FIFO manner at each link, thus it maintains statistical multiplexing gain and does not misorder packets; (3) choosing a smaller default initial timeout value for TCP can help enhance the performance of short TCP flows, however not as effectively as our scheme and at the risk of congestion collapse; (4) in the worst case, our proposal works as well as a regular RED scheme, in terms of response time and goodput.

1. Introduction

The ubiquitous heavy-tailed distributions in the Internet implies an interesting feature of the Internet traffic: most (e.g. 80%) of the traffic is actually carried by only a small number of connections (elephants), while the remaining large amount of connections are very small in size or lifetime (mice). In a fair network environment, short connections expect relatively fast service than long connections. However, sometimes we can not observe such nice property in the current Internet. Users may spend a long time downloading a plain text webpage — a situation for which "World-Wide Wait" was coined [1].

A careful look at the behavior of the major transport layer protocol TCP reveals some of the factors that may affect the performance of transferring a short file:

- TCP tries to conservatively ramp up its transmission rate to the maximum available bandwidth. Therefore, the sending window is initiated at the minimum possible value regardless of what is available in the network.
- As a reliable transport protocol, TCP couples error control with congestion control. Packet loss is detected by either the expiration of a retransmission timer or the arrival of a sequence of duplicate acknowledgments. Usually a retransmission after timeout can severely degrade the transmission rate of TCP. For a short connection, since most of the time the congestion window has a relatively small value, it does not have enough packets to activate the duplicate ACK mechanism, and thus packet loss always requires a timeout to detect.
- More importantly, TCP relies on its own packet samples to estimate an appropriate retransmission timeout (RTO) value. For the first control packets (SYN, SYN-ACK), and the first data packet, since no sampling data is available, TCP has to use a conservatively estimated initial timeout (ITO) value as RTO[1]. Losing these packets can have a disastrous effect on short connection performance due to the large timeout period.

For these reasons, short TCP flows are generally more conservative than long flows and thus tend to get less than their fair share when they compete for the bottleneck bandwidth. In this paper, we propose to give preferential treatment to short flows[2] with help from an Active Queue Management (AQM) policy inside the network. We also rely on the proposed Differentiated Services (Diffserv) architecture [3] to classify flows into short and long at the edge of the network. More specifically, we maintain the length of each active flow (in packets[3]) at the edge routers and use it to classify incoming packets. At the core routers, we employ the RIO queue management policy [4] to isolate packets from short flows and reduce the losses experienced by these packets.

Through extensive simulations, we confirm that preferential treatment is necessary to guarantee prompt response to short TCP flows. Moreover, since our threshold-based classification method treats the initial packets from a long flow as packets from a short flow, all flows benefit from experiencing less drops of the first few packets (including control packets), which otherwise would cause unnecessary stall of the session. We also confirm that our proposed scheme can achieve better fairness and response time for short flows than schemes without preferential treatment even when the value of the ITO is

This work was supported in part by NSF grants CAREER ANI-0096045 and ANI-0095988.

[1] The recommended ITO value is 3 seconds and is implemented in most modern operating systems [2]. Some old systems use a more conservative value of 6 seconds.

[2] We use flow, connection and session interchangeably in this paper.

[3] Without loss of generality we assume fixed-size packets and deal with packets rather than bytes.

reduced to match the actual configuration, as suggested in [2]. Since the RIO policy aims at maximizing statistical multiplexing, our proposed scheme does not hurt the utilization of the network. On the contrary, it sometimes can achieve better network goodput than traditional Drop Tail or RED [5] policies since it effectively avoids some unnecessary backoff of the sessions. In addition, RIO guarantees ordered delivery of packets at each link, and thus eliminates the possibility of packet reordering which may occur with other isolation-based policies (e.g. CBQ [6]).

Related Work: In a preliminary work [7], we study the interaction between short and long flows through a control theoretic approach. We then propose to isolate short and long TCP flows to enhance response time and fairness of short flows. Our results also reveal that class-based flow isolation combined with threshold-based classification at the edge may cause packet reordering, which may severely degrade TCP performance. Moreover, the bandwidth (load) control in [7] is performed at each core router, while in this paper we push bandwidth control to the edges of the network. Ng *et al.* [8] explore the idea of preferential treatment of *non-TCP* bursty flows over long CBR flows to guarantee different QoS service profiles. Nandy *et al.* [9] and Firoiu *et al.* [10] study service differentiation between delay-sensitive and throughput-sensitive *long-lived* flows through active queue management.

Analytical results of short TCP flow latency under different network conditions are given in [11]. These results reveal the vulnerability of short TCP flows. Seddigh *et al.* [2] show the negative impact of the initial timeout value on the short TCP flow latency and propose to reduce the default recommended value. To enhance the performance of short TCP flows, a number of proposals, e.g., [12], [13], [14], [15] suggest to either use a large initial window value or share the network measurement information from previous records or communication with neighbors. Notice that these proposals require modification of the TCP protocol and may lead to congestion collapse. Our proposed scheme differs from these proposals because our control is placed inside the network, and thus is more fair to clients who are unaware of such changes.

Crovella *et al.* [16] and Bansal *et al.* [17] explore the heavy-tailedness nature of the job size distribution and comment that size-aware job scheduling helps enhance the response time of short jobs without hurting the performance of long jobs. Their work inspires us to perform size-based service differentiation in the Internet for TCP flows.

Many studies have been conducted in the area of active queue management (AQM), most of them are variants of RED [5] or CBQ [6]. In our proposed scheme, we require the internal bottleneck routers to execute RIO (RED with In and Out) policy [4], which is a pioneering work by Clark and Feng.

Our Contribution: Our study provides an alternative usage of the Differentiated Services (Diffserv) architecture in [4]. The goal in [4] is to provide *long-lived* flows with "expected" bandwidth or fairness guarantees. Thus, packets are classified at the edge of the network as In or Out for that purpose, which may lead to suboptimal utilization of network resources [18]. On the other hand, our goal is to provide a *new* better-than-best-effort TCP service that attempts to *enhance the competence of short TCP flows*. This in turn creates a more fair network environment and better utilizes network resources, especially for Web traffic, the dominant traffic in the current Internet. We classify packets as In or Out to distinguish the length of the flow as Short or Long in order to preferentially treat short TCP flows inside the network.

In Section 2, we show in detail the necessity and benefit of preferential treatment of short TCP flows. We then present the network architecture and mechanisms required to implement our proposed scheme in Section 3. Section 4 uses simulations to confirm the advantage of our scheme. In Section 5, we discuss the issue of incremental deployment and extensions to the architecture. Section 6 concludes this paper with future work.

2. Analyzing Short TCP Flow Performance

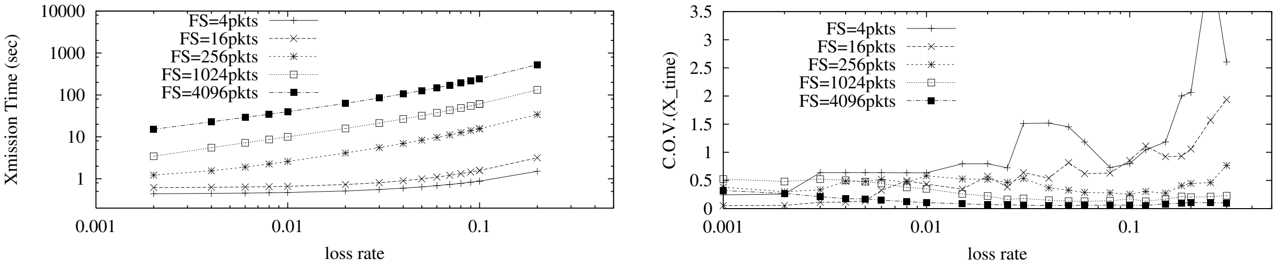
In this section, we show the relationship between loss rate and TCP flow transmission time through analytical and simulation methods. We then argue that, to achieve transmission rate comparable to that of long connections, short TCP connections should experience much less packet drops, which requires preferential treatment at the bottleneck link queues.

2.1. Sensitivity Analysis for Short and Long TCP Flows

In this section, we give the analytical results on the transmission time for TCP flows of different sizes. The derivation is similar to that in [11] and is omitted here due to space limitation. Detailed analysis can be found in [19]. Figure 1(a) gives in a log-log plot the average total latency with average $RTT = 0.1$ second, average $RTO = 4 \times RTT$ and the default initial retransmission timer $ITO = 3$ seconds, for a TCP flow of a fixed size FS for various loss rates.

We observe that the average transmission time of short TCP flows is not very sensitive to loss when the loss rate is relatively small, but it increases drastically as loss rate becomes larger (when persistent congestion happens). For extremely long flows, the average transmission time grows consistently with loss rate. Since it is hard, if not possible to compute the variance of transmission times, we use the ns simulator [20] to study the coefficient of variation (C.O.V.)[4] of transmission times for different flow sizes. In the simulation, a single TCP-Newreno connection transmits a file of size FS over a lossy link which randomly drops incoming packets with probability p. Packets never get dropped due to queue overflow. Once the file transfer finishes, a new connection of the same size immediately starts. We thus can measure the statistics of the

[4] C.O.V. is defined as the ratio between standard deviation and the mean of a random variable—smaller values imply less variability.



(a) Average Transmission Time (b) Coefficient of Variation

Fig. 1. Sensitivity Analysis of Transmission Time

transmission time. Figure 1(b) plots the C.O.V. against loss rate. We can observe an interesting trend: for small-size TCP flows, increasing the loss probability can lead to increased variability, while for long TCP flows, large loss rate reduces the variability of transmission times. We can give a possible explanation to this interesting behavior:

- When loss rate is high, TCP congestion control is more likely to enter the exponential backoff phase, which can cause significantly high variability [21] in the transmission time of each individual packet of a flow.
- When loss rate is low, depending on when the packet loss happens, TCP can either transmit a significant amount of packets in slow-start phase or have to transmit them in the less aggressive congestion avoidance phase. This adds another dimension to variability, which affects the length of each congestion epoch and how many packets can be transmitted in each epoch (until another loss is detected).
- Since the first source of variability is on individual packets of a flow, the law of large numbers indicates that its impact is more significant on short flows. On the other hand, the second source of variability is more pronounced for long flows since most short flows finish their transmission in slow-start phase.

We thus conclude that reducing the loss probability is more critical to help short TCP flows experience less variations in transmission (response) time. Observe that the C.O.V. of transmission times is closely related to the fairness of the system—smaller values imply higher fairness. Such interesting behavior motivates us to give preferential treatment to short TCP flows.

2.2. Preferential Treatment to Short TCP Flows

In this section we use a simple simulation set up to illustrate that, assuming the arrival of requests does not depend on the service time, giving preferential treatment to short TCP flows can *significantly enhance their transmission time, without degrading long flow performance*.

For this purpose, we use the ns simulator [20] to set up the following scenario: 10 long (10000-packet) TCP-Newreno flows and 10 short (100-packet) TCP-Newreno flows are competing for bandwidth over a 1.25Mbps link[5]. Flows start periodically and stay in the network until they finish transmission. We then vary the queue management policy at the bottleneck link and measure the instantaneous portion of bandwidth taken by each class of flows[6] to show the effect of preferential treatment. We consider traditional Drop Tail queue, RED queue and our proposed RIO-PS (RIO with Preferential treatment to Short flows) queue (see Section 3 for details), and plot the results in Figure 2.

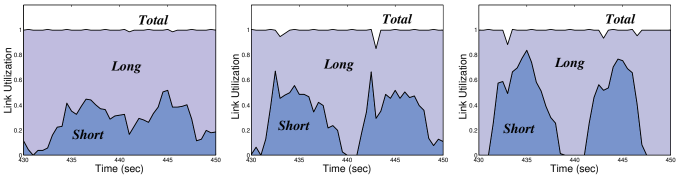


Fig. 2. Impact of Preferential Treatment— Link utilization under Drop Tail (left), RED (middle), and RIO-PS (right)

It is not surprising that a Drop Tail queue fails to give fair treatment to short TCP flows. Since most of the time short flows have a small congestion window compared to long flows, and also because a Drop Tail queue favors aggressive flows (with larger windows) [5], short flows can sometimes lose their fair share. A RED queue gives almost fair treatment to all flows, while our RIO-PS queue gives short flows more than their fair share.

As we can notice from the graphs, although when there are backlogs from both classes of flows, short flows under our scheme can temporarily grab ("steal") more bandwidth from long flows, in the long run, their early completion returns an equal amount of resources (if not more, avoiding packet retransmissions) to long flows. In other words, since a long-lived TCP flow can adapt well to the network state and so can acquire about the same amount of resources (bandwidth) to transmit a certain file, giving preferential treatment to short flows does not sacrifice the long-term goodput[7] of long flows.

[5] Typically a short TCP flow has less than 20 packets to transmit. We only consider in this motivation section larger flow sizes to have a sharper contrast in our illustrations.

[6] This instantaneous fraction of bandwidth is computed by the ratio of packets transmitted over link bandwidth over the last 0.5 second.

[7] Goodput is defined as the number of successfully delivered packets over the simulation period.

In fact, this preferential treatment might even enhance the transmission of long flows since they operate in a more stable network environment (less disturbed by short flows) for longer periods. On the other hand, in a congested network, reducing the packet drops experienced by short flows can significantly enhance their response time and fairness among them, as implied by the analysis in the previous section. Table I gives the measured network goodput over the 500-seconds simulation period.

Table I also gives the measured goodput for a less loaded network with bottleneck link bandwidth of 1.5 Mbps. We observe that when load is low, our RIO-PS scheme (as well as RED) produces a slightly lower goodput (less than 1.5%) than that of DropTail. However, as load becomes higher, our scheme can even achieve slightly higher goodput than all other schemes[8].

We thus propose to differentiate between short and long TCP flows inside the network. To this end, we employ a Diffserv-like network architecture [22].

Link B/W	Flows	DropTail	RED	RIO-PS
1.25Mbps	All	153479	154269	154486
	Short	40973	49897	49945
	Long	112506	104372	104541
1.5Mbps	All	185650	184315	183154
	Short	43854	49990	49990
	Long	141796	134325	133164

TABLE I
NETWORK GOODPUT UNDER DIFFERENT SCHEMES

3. Proposed Scheme:Architecture and Mechanisms

In this section, we discuss the detailed implementation of our proposed scheme, including the network architecture we assume and the supporting mechanisms required to differentiate between short and long TCP flows.

3.1. The Architecture

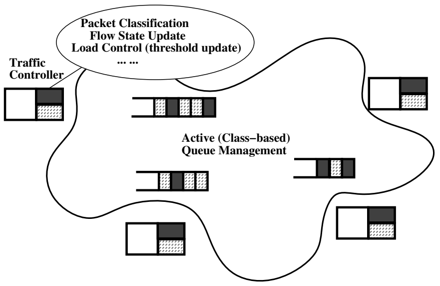


Fig. 3. Proposed Architecture

[8]Note that under Drop Tail, short flows are not protected well, so long flows achieve higher goodput than in other schemes.

We assume a Diffserv-like scalable active domain management architecture where edge routers of an administrative domain perform all the per-flow information maintenance tasks, and the core routers only need to actively manage *per-class* flows. Figure 3 illustrates the operations a packet needs to go through in such architecture. The edge router tags (marks) individual packets as belonging to the class of short flows or the class of long flows. Such information is then used by core routers at which any class-based active queue management policy can be implemented.

3.2. Edge Router: Packet Classification and State Maintenance

We rely on the edge router to determine whether a packet is coming from a long or short flow. Accurate flow characterization can be very complicated due to the high variability of user behavior in the current Internet. Instead, we resort to a simple threshold-based approximation method. For each active flow, we simply maintain a counter that tracks how many packets have been observed so far. Once the counter exceeds a certain threshold (L_t), we consider the flow to be long, and classify the following packets as "Long" packets. Notice that under such classification method, the first L_t packets will be classified as "Short" packets. We discuss the pros and cons of such method in Section 5.

The per-flow state information are softly maintained to detect the termination of a flow. More specifically, the flow hash table is updated periodically. Every T_u, a flow is considered terminated (and its entry removed) if no packets from that flow is observed in the last T_u time units.

The value of the threshold L_t can be static or dynamic. However, since the edge router already maintains enough per-flow state information, in our proposed scheme, we let the edge router adjust the threshold dynamically so as to balance the number of active short and active long flows. For such purpose, the edge router is configured with one parameter called Short-to-Long-Ratio (SLR) which is the ratio between the number of active short flows and that of long flows. The edge router then periodically (every T_c) performs additive increase additive decrease control over the threshold to achieve the target SLR (See [19] for the pseudo-code). Other control policies are also possible, however, our simulation results show that the performance is not quite sensitive to SLR. Choosing appropriate values of T_u and T_c is an issue that needs further research (see Section 5). In our simulation, we empirically choose $T_u = 1$ second and $T_c = 10$ seconds.

3.3. Core Router: Preferential Treatment to Short Flows

We require the core routers to give preferential treatment to short flows. There are many candidate queueing policies for such purpose. We choose the RIO (RED with In and Out) [4] queue not only because of its comformity to the Diffserv specification [4], but also because of many advantages it has over other candidates[9]. Figure 4 illustrates the early dropping/marking function of a RIO queue.

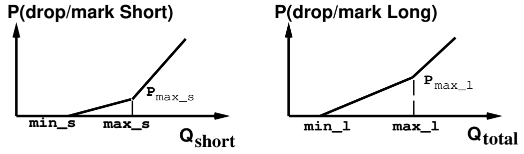


Fig. 4. RIO queue with Preferential treatment to Short flows

Briefly speaking, the operation of the queue on In (Short) packets is not affected by the arrival of Out (Long) packets since the dropping/marking probability for Short packets is computed based on the average backlog of Short packets (Q_{short}) only. On the contrary, for the Long packets, the total average queue size (Q_{total}) is used to detect incipient congestion and thus flows that carry Long packets have to give up some resources (backoff) when there is persistent backlog from both classes of packets. The pseudo-code of the RIO algorithm can be found in [4]. Note that we use the "gentle_" variant of the RED dropping function as recommended in [23].

Clark and Feng [4] give an excellent discussion on the design features of RIO queue, and we only highlight few of them here:

- Only a single FIFO queue is used for all the packets, thus packet reordering will not happen at a link even when packets from the same flow are classified differently.
- RIO inherits all the features of RED, which include protection of bursty flows, fairness within each class of traffic and detection of incipient congestion.
- RIO performs soft prioritization, thus does not lose the benefit of statistical multiplexing.

Henceforth, we will call such queue RIO-PS (RIO with Preferential treatment to Short flows). When persistent congestion happens, how much bandwidth share short flows can receive depends on the configured parameters for each class of traffic. In our simulation, we choose the RIO-PS parameters such that short TCP flows are guaranteed around 75% of the total link bandwidth in times of congestion.

4. Simulation

We use the *ns* [20] simulator to study the performance of our proposed scheme and compare it to size-oblivious Drop Tail and RED schemes.

4.1. Simulation Setup

We assume network traffic is dominated by Web-like transactions. To this end, we adopt the Web traffic model developed in [24]. In this model, randomly selected clients initiate sessions which involve surfing several web pages of different sizes from randomly chosen websites. Each page may contain several objects, each of which requires a TCP connection for delivery (in other words, an HTTP 1.0 model is assumed). To request a page, the client sends a request packet to the server (simulating the GET message in a typical HTTP session), the server responds with an acknowledgment and then starts to transmit the web page requested by the client. The distributions of inter-page and inter-object time (in seconds), page size and object size (in packets) are given in Table II. The load is carefully tuned to be close to the bottleneck link capacity[10]. The topology used in our simulation is shown in Figure 5.

Name	inter-page	objs/page	inter-obj	obj size
Exp1 client 1	Exponential mean 9.5	Uniform min 2 max 7	Exponential mean 0.05	Bounded Pareto [4,200000] shape 1.2
Exp2 client 1	Exponential mean 9.5	Uniform min 2 max 7	Exponential mean 0.05	Bounded Pareto [4,500] shape 1.2
client 2	Exponential mean 192	Uniform min 1 max 3	Exponential mean 1.5	Bounded Pareto [400,200000] shape 1.2

TABLE II
WEB TRAFFIC CONFIGURATION

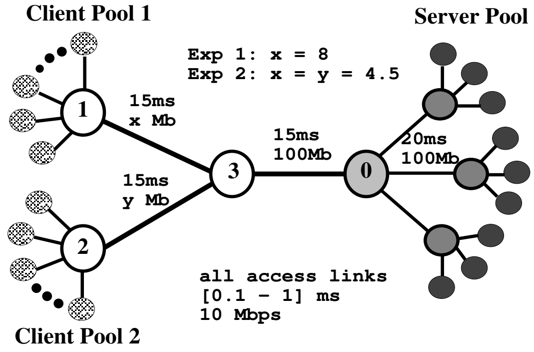


Fig. 5. Simulation Topology

Node 0 acts as the entry edge router of the bottleneck domain, and nodes 1,2,3 are core routers. Two sets of clients are connected to the server pool through individual bottleneck links (links 1-3 and 2-3, respectively). Other links are configured to be lossless. The buffer size of each bottleneck link and the configuration of its queue management policy are carefully tuned according to [25] so as to maximize power[11]. For our RIO-PS queue, we empirically choose its parameters so that short TCP flows experience less packet drops without starving long flows[12]. Recall that since the flow states are updated periodically at the edge routers, some terminated short

[9] We also discuss its disadvantages in Section 5.

[10] As revealed in [25], the advantage of RED is more pronounced in this load regime.

[11] Power is defined as the ratio between throughput and latency, so a high power implies low delay and high throughput.

[12] In our simulations, we choose the RIO-PS parameters such that short TCP flows are guaranteed around 75% of the total link bandwidth. Even if

flows may still be treated as "active". Considering such measurement error due to lazy state update, we choose SLR to be 3 so that the actual number of short flows is approximately equal to that of long flows, however, we varied this ratio between 2 and 6 and didn't observe significant difference. The detailed (default) configuration is listed in Table III. We have repeated each simulation with different random seeds and produced similar results[13].

Description	**Value**
Packet Size	500 bytes
Maximum Window	128 packets
TCP version	Newreno
TCP timeout Granularity	0.1 seconds
Initial Retransmission Timer	3.0 seconds
B/W delay product (BDP)	$\approx$ 200 pkts (Exp1) $\approx$ 120 pkts (Exp2)
Bottleneck Buffer Size (B)	DropTail: $1.5\times$ BDP RED/RIO-PS: $2.5\times$BDP
Q. Parameters	$(min_{th}, max_{th}, P_{max}, w_q)$
RED	(0.15B, 0.5B, 1/10, 1/512)
RIO-PS short	(0.15B, 0.35B, 1/20, 1/512)
RIO-PS long	(0.15B, 0.5B, 1/10, 1/512)
RED & RIO-PS	ecn_ on, wait_ on, gentle_ on
Edge Router	$SLR = 3, T_u = 1\ sec, T_c = 10\ sec$
Foreground Traffic	
(Src, Dest)	(Server Pool, Client Pool)
Long Connection Size	1000 packets
Short Connection Size	10 packets

TABLE III
SIMULATION CONFIGURATION

4.2. Experiment 1: Single Client set

In this experiment, there is only one set of clients involved (client pool 1). Therefore, the traffic seen at the core router 1 is the same as that at the edge router 0.

4.2.1. Study of the web traffic

We run the experiment for 4000-seconds simulation time. After some warm-up period (2000 seconds), we start to record the response time for each successfully downloaded web object. Figure 6(a) plots the averaged response time for different sized objects[14].

The results indicate that preferential treatment can cut the average response time for short and medium sized files significantly (25-30 %). In [2], Seddigh and Devetsikiotis observed that the default ITO value of current implementations of TCP is too conservative and can thus significantly hurt the performance of short flows. They then propose to reduce the default ITO value from 3 seconds to 1 second. However, such aggressive value may lead to congestion collapse in some parts of the Internet where links are slow and have very large round trip propagation delay. Using aggressive retransmission timers at the ends of such links can cause unnecessary retransmissions, an undesirable phenomenon that TCP designers always wanted to avoid [27]. Nevertheless, we also run the simulation after setting the ITO to 1 second and plot the results in Figure 6(b). We observe that reducing the ITO value significantly reduces the performance gap between RED and our proposed scheme (now only 10-15%) for short connections (file size less than few dozens of packets). We still observe large performance improvements with RIO-PS for medium sized connections (15-25%).

Figure 7 plots the instantaneous queue size and drop rate in the last 20 seconds for the case of 3-seconds ITO. To compare with DropTail, which does not support ECN, for RIO-PS and RED, we actually plot the sum of the forced drop (drop due to buffer overflow) and ECN mark rate. Also for RIO-PS we plot such measure for each class of flows. Observe that due to the heavy-tailedness of the file size distribution (thus the session length distribution), the load at the bottleneck link shows high variability over time.

We observe that our RIO-PS scheme reduces the overall mark/drop rate without the risk of congestion collapse (overloading) at the bottleneck link. A closer examination of the packet record reveals that the reduced dropping/marking rate comes from the fact that short connections rarely experience packet drops, and the number of ECN notifications sent to these connections are much smaller than their rivals. We argue that short connections are not responsible for controlling congestion since the timescale at which they are operating is much smaller than the effective control regime of a TCP congestion control algorithm. Thus, preferential treatment to short connections does not hurt, if not benefits the congestion state of the network.

4.2.2. Study of foreground traffic

To better illustrate the effect of the queue management policy on fairness of TCP connections, we periodically injected 10 short (every 25 seconds) and 10 long (every 125 seconds) foreground TCP connections and recorded the response time T_i for each connection i. The fairness index of response time, computed as $FI = \frac{(\sum_{i=1}^{10} T_i)^2}{10\sum_{i=1}^{10} T_i^2}$, is reported in Figure 8[15]. Figure 9 plots the transmission time for each individual connection and their ensemble average[16].

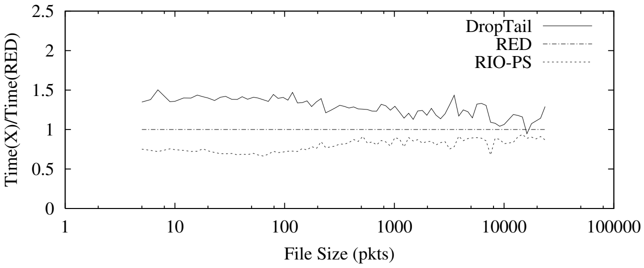
short flows are given *absolute* priority, long flows may still pay very little penalty because of the heavy-tailedness of the workload, in which most of the bytes are due to the small percentage of long flows [16].

[13] Also note that we only present the results of ECN capable routers here since ECN is recommended to be used with RED queues [26]. We also simulated the case when ECN was turned off and observed more pronounced performance gain of our proposed scheme over traditional RED, since RED with ECN turned on potentially enhances short flow performance [26].

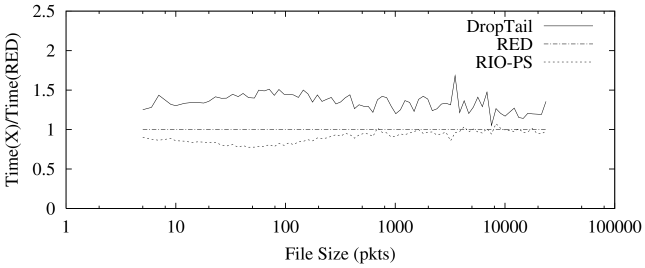
[14] The file sizes are grouped into 85 bins, each of which spans an interval $[x, 1.1x]$, and the average is taken over each bin.

[15] Due to space limitation, the results for the case of $ITO = 1$ second are not presented here. The conclusion doesn't change, though.

[16] For clarity, we don't plot the results of DropTail queues, whose performance (in terms of fairness and average transmission time) is much worse.

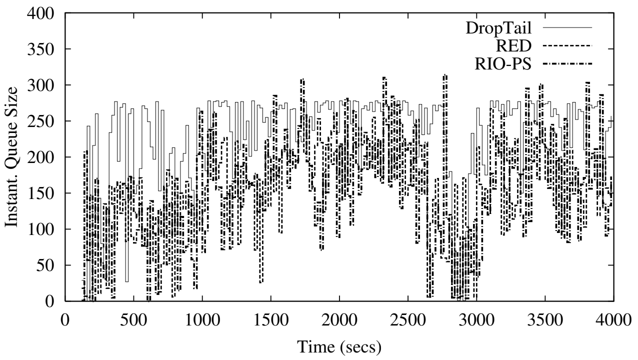


(a) Initial Retransmission Timer 3 seconds

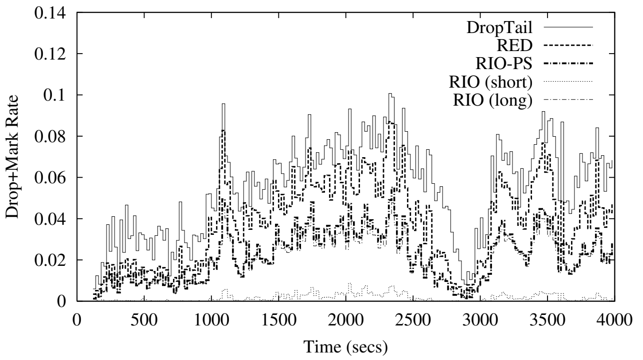


(b) Initial Retransmission Timer 1 second

Fig. 6. Average response time relative to RED

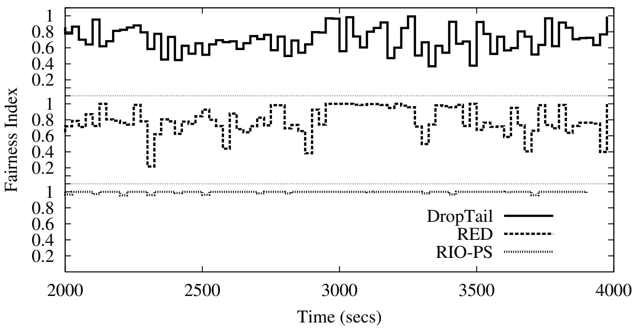


(a) Instantaneous Queue Size

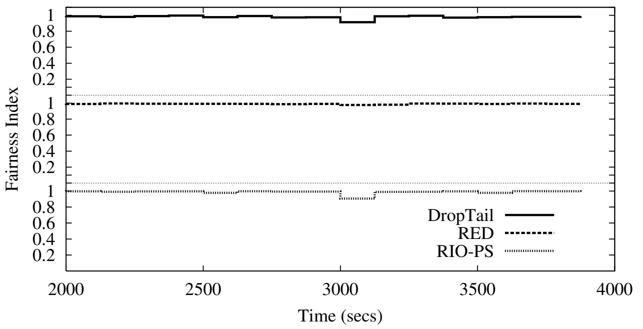


(b) Instantaneous Drop/Mark Rate

Fig. 7. Revealing the secret of better performance

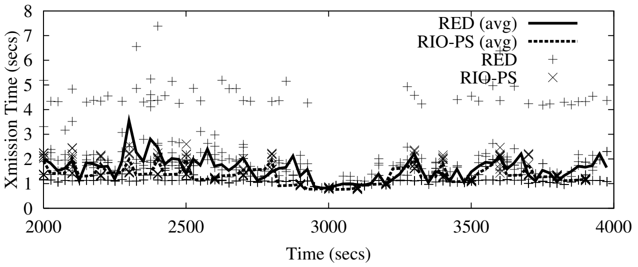


(a) Fairness Index of Short Connections

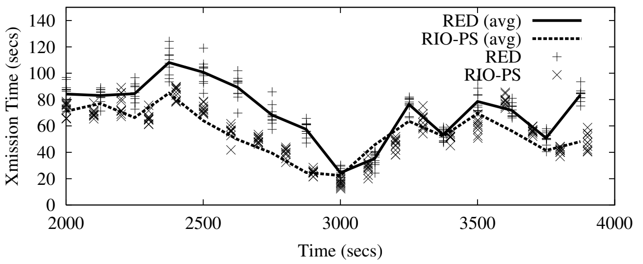


(b) Fairness Index of Long Connections

Fig. 8. Fairness of Transmission Time



(a) Transmission Time of Short Connections



(b) Transmission Time of Long Connections

Fig. 9. Transmission Time of Foreground Traffic

We observe that even with RED queues, many short flows experience at least one timeout, many of them (the "outliers" in Figure 9(a)) require waiting for a default ITO value (3 seconds) due to loss of the first packet[17]. On the contrary, our proposed preferential treatment scheme provides short TCP flows with much less drops and thus more fair service without hurting the performance of long connections.

4.2.3. Does fairness hurt goodput?

Table IV summarizes the overall goodput[18] of the network. As we can see, our proposed scheme does not hurt, if not (slightly) improves the overall goodput over Drop Tail.

Scheme	DropTail	RED	RIO-PS
Exp1 (ITO=3sec)	4207841	4264890	4255711
Exp1 (ITO=1sec)	4234309	4254291	4244158

TABLE IV

NETWORK GOODPUT OVER THE LAST 2000 SECONDS

4.3. Experiment 2: Unbalanced Requests

We also simulated the case where the file requests on different routes are unbalanced. In other words, most small file requests are made along a certain route while large files are transferred over another route. In such extreme case, our proposed scheme reduces to traditional unclassified traffic plus RED queue policy. However, giving preferential treatment to the initial few packets still reduces the chance of retransmitting the first packet and thus helps most short connections finish quickly. The simulation results are omitted here due to space limitation. The reader is referred to [19] for details.

5. Discussion

5.1. Comments on the Simulation Model

Our simulation is performed using the "Dumbbell and Dancehall" (one-way traffic) model [28], and all TCP connections have similar end-to-end propagation delays, thus such model may not represent the common topology seen by Internet users. However, such assumption is only for the sake of easy comparison of individual TCP performance. When there is also reverse traffic present, our proposed scheme will even have more superior performance over other traditional policies. This is because the setup of a web session (and most other TCP sessions) requires the client side to exchange a short sequence of control packets with the server. Such a short exchange will be better protected under our proposed scheme, especially in the presence of heavy traffic in the same direction. More complicated topology models may impact the performance of our scheme due to unbalanced requests. However, as shown by our simulation results in [19], our scheme can still work at least as well as traditional policies even in extremely unbalanced cases.

5.2. The Queue Management Policy

To be conformant to existing Diffserv implementations, we chose RIO-like AQM policy to be used at the core routers. However, RIO queue provides neither absolute aggregate (class-based) guarantees nor relative flow guarantees. To attack these problems, one may resort to AQM policies like the PI controlled RED queue [29], or a service model like the Proportional Diffserv proposed in [30] for better control over the classified traffic and more predictable service.

5.3. Deployment Issues

Our proposed scheme requires edge devices to be able to perform per-flow state maintenance and per-packet processing. Fortunately, it has been shown in [31] that providing such functionality does not have a significant impact on the end-to-end performance. The other issue is on whether such scheme is incrementally deployable. Notice that our objective is to protect short delay-sensitive flows, and our success relies on how well the classification (isolation) is done at the edge (bottleneck) router. The scheme does not require the queue mechanisms to be implemented at each router. For example, the advanced edge device can be placed in front of a busy web server cluster, and RIO-PS only implemented at the busy bottleneck links to achieve the desired performance objective.

5.4. Flow Classification

We use a threshold-based classification method since the intermediate (edge) node can not predict in advance whether a flow is long or short. Such method thus mistakenly classifies the first few packets of a long flow as if they came from a short flow. However, such "mistake" may help to enhance performance. This is because the first few packets of a long TCP flow are under exactly the same control method as packets from a short flow. Thus, they are more vulnerable to packet losses and deserve to be treated with high preference. Such "mistake" also makes the system more fair to all TCP connections.

5.5. Controller Design

Our proposed scheme involves controller design issues at different places and timescales. The issue of edge load control is certainly a topic for further research, although our preliminary results indicate that the performance is not very sensitive to the target load ratio of active short to active long flows (the value of SLR at the edge). The "actual" SLR depends on the

[17] Note that although ECN marks, instead of drops, packets in times of congestion, it does not mark the first (SYN) packet since the negotiation of ECN-capability at both ends has not finished at this moment. Therefore, initiating a TCP session when the network is congested still faces the risk of a long timeout if the SYN packet is dropped. Although *ns* does not generate control (SYN, SYN-ACK) packets, it similarly treats the first data packet, *i.e.* a RED with ECN scheme does not mark, rather it may drop, the first data packet in times of congestion.

[18] The 95% confidence intervals are within 3% of the mean.

values of T_c and T_u, which determine how often the classification threshold L_t and active flow table are updated, respectively. Smaller values of T_c and T_u may increase accuracy at the expense of increased overhead.

5.6. Malicious Users

One concern regarding our proposed scheme may be that users are then encouraged to break long transmissions into small pieces so that they can enjoy faster services. This in turn may end up causing severe congestion. However, we argue that such incentive may not be so attractive to users given the large overhead of fragmentation and reassembly. In addition, a dynamic classification would adapt to the current distribution of transfer sizes.

6. Conclusions and Future Work

TCP traffic constitutes the majority of the bytes flowing over the Internet today. In this paper, we present a Diffserv-like architecture where edge routers implement a new TCP service by classifying TCP flows based on their size (lifetime). Core routers implement a simple RIO (weighted RED) policy to give preferential treatment to packets from short TCP flows. This has several advantages over size-oblivious Drop Tail or RED schemes: (1) the performance of the majority of TCP flows (the short transfers or mice) is significantly improved in terms of response time and fairness; (2) because short TCP flows can be rapidly served, the performance of the few TCP long flows (the large transfers or elephants) is also enhanced or at worst minimally affected; (3) the overall goodput of the system is thus improved or at least stays almost the same (a work-conservation feature); and (4) the proposed architecture is extremely flexible in that the functionality that defines the new TCP service can be largely tuned only at the edge routers.

The size-aware management of traffic has shown considerable promise. We have shown how differentiation of TCP flows based on size can be effectively used to isolate flows that mostly operate in different regimes: short TCP flows in slow-start and long TCP flows in congestion-avoidance. At the network layer, Shaikh *et al.* [32] investigated load-sensitive routing of only long-lived IP flows in order to improve routing stability and reduce overhead. However, they have not considered the effect on congestion-controlled sources (such as TCP and TCP-friendly applications). We are currently investigating an approach that integrates size-aware traffic management at both the network and transport layers.

References

[1] R. Khare and I. Jacobs, "W3C Recommendations Reduce 'World Wide Wait'," http://www.w3.org/Protocols/NL-PerfNote.html.

[2] N. Seddigh and M. Devetsikiotis, "Studies of TCP's Retransmission Timeout Mechanism," in *Proc. ICC 2001*, Helsinki, Finland, June 2001.

[3] Y. Bernet and et al, "A Framework for Differentiated Services," Internet Draft ⟨draft-ietf-diffserv-framework-02.txt⟩, Feburary 1999.

[4] D.D. Clark and W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, vol. 6(4), pp. 362–373, August 1998.

[5] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1(4), pp. 397–413, August 1993.

[6] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, August 1995.

[7] I. Matta and L. Guo, "Differentiated Predictive Fair Service for TCP Flows," in *Proc. ICNP'2000*, Osaka, Japan, November 2000.

[8] T.S.E. Ng, D.C. Stephens, I. Stoica, and H. Zhang, "Supporting Best-Effort Traffic with Fair Service Curve," in *Proc. IEEE GLOBECOM'99*, Rio de Janeiro, Brazil, December 1999.

[9] B. Nandy, N. Seddigh, A. Chapman, and J. Hadi Salim, "A Connectionless Approach to Providing QoS in IP Networks," in *Proc. IFIP Eighth Conference on High Performance Networking (HPN'98)*, Vienna, September 1998.

[10] V. Firoiu and X. Zhang, "Best Effort Differentiated Services: Trade-off Service Differentiation for Elastic Applications," Tech. Rep., Nortel Networks, September 2000.

[11] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[12] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," Internet RFC 2414, September 1998.

[13] V. Padmanabhan and R. Katz, "TCP Fast Start: a Technique for Speeding Up Web Transfers," in *Proc. IEEE Globecom'98 Internet Mini-Conference*, November 1998.

[14] Y. Zhang, L. Qiu, and S. Keshav, "Speeding Up Short Data Transfers: Theory, Architecture Support, and Simulation Results," in *Proc. NOSSDAV 2000*, Chapel Hill, NC, June 2000.

[15] J. Heidemann, "Performance Interactions Between P-HTTP and TCP Implementations," *ACM Computer Communication Review*, vol. 27(2), pp. 65–73, April 1997.

[16] M. Crovella, R. Frangioso, and M. Harchol-Balter, "Connection Scheduling in Web Servers," in *Proc. the 1999 USENIX Symposium on Internet Technologies and Systems (USITS '99)*, Boulder, Colorado, October 1999.

[17] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness," in *Proc. of ACM SIGMETRICS 2001*, Boston, MA., June 2001.

[18] T. Bonald and L. Massoulié, "Impact of Fairness on Internet Performance," in *Proc. ACM SIGMETRICS 2001*, Boston, MA, June 2001.

[19] L. Guo and I. Matta, "The War Between Mice and Elephants," Tech. Rep. 2001-005, Dept. of Computer Science, Boston Univ., May 2001, http://www.cs.bu.edu/techreports/2001-005-war-tcp-rio.ps.Z.

[20] E. Amir et al., "UCB/LBNL/VINT Network Simulator - ns (version 2)," Available at http://http://www.isi.edu/nsnam/ns/.

[21] L. Guo, M. Crovella, and I. Matta, "How does TCP Generate Pseudo-Self-Similarity," in *Proc. of 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Cincinnati, OH, August 2001.

[22] S. Black et al., "An Architecture for Differentiated Services," Internet RFC 2475, December 1998.

[23] S. Floyd, "Recommendation on using the "gentle_" variant of RED," http://www.aciri.org/floyd/red/gentle.html, March 2000.

[24] A. Feldmann, A.C. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP Traffic: A Study of the Role of Variability and the Impact of Control," in *Proceeding of ACM SIGCOMM'99*, Boston, Mass, September 1999.

[25] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, "Tuning RED for Web Traffic," in *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, Aug.-Sep. 2000.

[26] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, vol. 24(5), October 1994.

[27] V. Jacobson, "Congestion Avoidance and Control," in *Proc. ACM SIGCOMM'88*, Stanford, CA, August 1988.

[28] K. Nichols, "Dumbbells, Dancehalls, and Beyond: Understanding QoS Prior to Deployment," in *Proceedings of the Internet Bandwidth Management Summit (iBAND3)*, San Francisco, CA, October 1999.

[29] C.V. Hollot, V. Misra, D. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," in *Proc. IEEE INFOCOM 2001*, Anchorage, AL, April 2001.

[30] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional Differentiation and Packet Scheduling," in *Proc. ACM SIGCOMM '99*, Boston, Massachusetts, August 1999.

[31] R. Guérin, L. Li, S. Nadas, P. Pan, and V. Peris, "The Cost of QoS Support in Edge Devices: An Experimental Study," in *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.

[32] A. Shaikh, J. Rexford, and K.G. Shin, "Load-Sensitive Routing of Long-Lived IP Flows," in *Proceedings of SIGCOMM'99*, Boston, MA, September 1999.