Understanding the End-to-End Performance Impact of RED in a Heterogeneous Environment

Yin Zhang, Lili Qiu
Department of Computer Science
Cornell University
Ithaca, NY 14853, USA
{yzhang, lqiu}@cs.cornell.edu

Abstract— **Random Early Detection (RED) is the recommended active queue management scheme for rapid deployment throughout the Internet. As a result, there have been considerable research efforts in studying the performance of RED. However, previous studies have often focused on relatively homogeneous environment. The effects of RED in a heterogeneous environment are not thoroughly understood. In this paper, we use extensive simulations to explore the interaction between RED and various types of heterogeneity, as well as the impact of such interaction on the user-perceived end-to-end performance. Our results show that overall RED improves performance at least for the types of heterogeneity we have considered.**

Keywords— **Random early detection (RED), explicit congestion notification (ECN), simulation, performance evaluation**

I. Introduction

The Internet Research Task Force (IRTF) is promoting deployment of *active queue management* to improve the performance of today's Internet [3]. In particular, *random early detection*, better known as RED [10], is singled out as the recommended scheme for use in the Internet.

With RED, a router will detect congestion before the queue overflows, and provide an indication of this congestion to the end nodes. It may use one of several methods for indicating congestion to end-nodes. One is to use packet drops. Alternatively, it can set a Congestion Experienced (CE) bit in a packet header as an indication of congestion, instead of relying solely on packet drops. The latter methods is commonly referred to as *explicit congestion notification* (ECN) [15], [34], [35]. The major advantage of active queue management mechanisms like RED is that the transport protocols with congestion control (e.g., TCP) do not have to rely on buffer overflow as the only indication of congestion. This can potentially reduce unnecessary queuing delay for all traffic sharing that queue.

There have been considerable research efforts in studying the performance of RED [6], [7], [8], [10], [18], [26], [24], [25], [38]. However, most previous studies only consider relatively homogeneous environment. Moreover, as pointed out in [6], the performance metrics have largely been network-centric measures such as network link utilization and aggregate TCP throughput. The end-to-end performance impact of RED in a heterogeneous environment has not been thoroughly explored.

It is well-known that the Internet is a highly heterogeneous environment and that heterogeneity can have significant impact on the network performance. Therefore, in order to understand the performance of RED in the real Internet, it is crucial to consider heterogeneity. In this paper, we use extensive simulations to explore the end-to-end effects of RED in a heterogeneous environment. In particular, we are interested in understanding the interaction between RED and different types of heterogeneity and quantifying the impact of such interaction on user-perceived end-to-end performance.

We consider the following five types of heterogeneity in this paper:

- *Mix of long-lived and short-lived TCP connections*. Numerous measurements show that the Internet traffic is now dominated by short flows involving small Web objects 10-20KB in size [20], [37] (the so called *mice*). Even with Persistent HTTP [31], the flow length is unlikely to change significantly, because the average Web document is only around 30KB [23]. While most Internet flows are short-lived, the majority of the packets and bytes belong to long-lived flows (the so called *elephants*), and this property persists across several levels of aggregation [4], [5], [37], [19]. Therefore, it is very important to understand the performance of RED on a workload consisting of both long-lived bulk transfers and short-lived Web data transfers.
- *Mix of TCP and UDP traffic*. Due to the proliferation of streaming media content over the Internet, UDP-based real-time traffic forms a significant portion of today's Internet traffic [40]. Real-time traffic is often sensitive to network latency and packet losses, but is not necessarily responsive to network congestion. Therefore, it is important to understand the performance impact of RED in an environment with both TCP and UDP traffic.
- *Mix of ECN-capable and non-ECN-capable TCP connections*. With ECN, a router set the CE bit in the packet header as an indication of congestion, instead of dropping the packet. The use of the CE bit would allow the receiver(s) to receive the packet, avoiding the potential for excessive delays due to retransmission after packet losses. However, ECN has not yet been widely deployed in today's Internet. If it is ever going to be widely deployed, the deployment will be incremental. That is, ECN-capable and non-ECN-capable TCP connections will coexist for a long period of time. Therefore, it is important to understand the effect of RED in an environment with competing ECN-capable and non-ECN-capable TCP traffic.
- *Different roundtrip times*. In the real Internet, different flows sharing the same bottleneck link can have different roundtrip times (RTT's). It is well-known that TCP has bias against long roundtrip time connections. We are interested in understanding whether RED can reduce such discrimination compared with the Drop-Tail (DT) policy.
- *Two-way traffic*. In the real Internet, there is typically data traffic in both directions. It is well-known that the presence of two-way traffic can cause ACK-compression [41], which can in

turn make TCP traffic much more bursty than in the one-way traffic case. In order to understand the performance of RED in the real Internet, it is important to consider the effect of two-way traffic.

The remainder of this paper is organized as follows. In Section II, we provide a more in-depth introduction to the RED algorithm. In Section III, we briefly overview the literature on the performance evaluation of RED. In Section IV we describe our simulation setup. In Section V, we present detailed simulation results. We end with conclusions and future work in Section VI.

II. Background

Consider a router with a buffer size of B packets. With the RED buffer management scheme, a router detects congestion by the average queue length ($\bar{q}$), which is estimated using an exponentially weighted moving average: $\bar{q} \leftarrow (1 - w_q) \cdot \bar{q} + w_q \cdot q$, where w_q is a fixed (small) parameter and q is the instantaneous queue length. When the average queue length exceeds a minimum threshold ($\min_{th}$), incoming packets are probabilistically dropped or marked with the Congestion Experienced bit [15], [34], [35]. The probability that a packet arriving at the RED queue is either dropped or marked depends upon several control parameters of the algorithm. An initial drop/mark probability P_b is computed using a drop function D based on the average queue length $\bar{q}$ and three control parameters $\max_p$, $\min_{th}$, and $\max_{th}$. The actual probability is a function of the initial probability and a count of the number of packets enqueued since the last packet was dropped: $P_a = P_b/(1 - count \times P_b)$.

In the original RED scheme, $D(\bar{q}) = 1$ if $\bar{q} \geq \max_{th}$, which means all incoming packets are dropped or marked when the average queue length exceeds $\max_{th}$. As shown by Firoiu *et al.*[8], this can lead to oscillatory behavior. Recently, Floyd recommended using the 'gentle_" variant of RED [17], which uses the following modified dropping function D (illustrated in Fig. 1):

$$D(\bar{q}) = \begin{cases} 0 & \text{if } \bar{q} < \min_{th} \\ 1 & \text{if } \bar{q} \geq 2 \cdot \max_{th} \\ \frac{\bar{q} - \min_{th}}{\max_{th} - \min_{th}} \cdot \max_p & \text{if } \bar{q} \in [\min_{th}, \max_{th}] \\ \frac{\bar{q} - \max_{th}}{\max_{th}} \cdot (1 - \max_p) + \max_p & \text{otherwise} \end{cases}$$

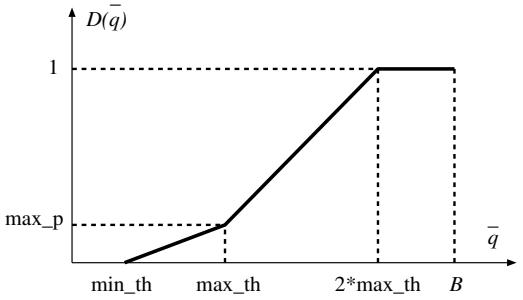


Fig. 1. Drop function of RED with the "gentle_" modification.

As shown by Rosolen *et al.*[32], [33], the "gentle_" option makes the RED much more robust to the setting of the parameters $\max_{th}$ and $\max_p$. Therefore, we turn it on for all simulations in this paper.

RED has four control parameters: $\min_{th}$, $\max_{th}$, $\max_p$, and w_q. How to properly configure these parameters has been the subject of many studies [6], [8], [11], [16]. The focus of our work is on understanding the interaction between RED and different types of heterogeneity. Therefore, instead of proposing any new recommendations on configuring RED parameters, we closely follow the guidelines by Floyd [16]. More specifically, we always use the recommended values of $\max_{th} = 3\min_{th}$, $\max_p = 0.1$, and $w_q = 0.002$. The recommended value for the last parameter $\min_{th}$ is 5 packets. However, as noted in [16], the optimal setting for $\min_{th}$ also depends partly on the link speed, propagation delay, and maximum buffer size. Therefore, besides the recommended value of 5 packets, we also experiment with two different values of $\min_{th}$ based on the buffer size: $B/6$ and $B/9$, where B is the maximum buffer size.

III. Previous Work

Since RED was initially proposed in 1993 by Floyd *et al.*[10], there has been a vast volume of research on studying the performance of RED [6], [7], [8], [10], [18], [26], [24], [25], [38]. There have also been numerous proposed modifications and alternatives to RED, such as BLUE [12], SRED (Stabilized RED) [28], Adaptive RED [11], FRED (Fair Random Early Drop) [22], and BRED (Balanced RED) [2]. The results from these studies have provided valuable insights to the RED algorithm. Unfortunately, they also suffer from some notable limitations:

- Most previous studies have focused on relatively homogeneous environment. Many studies either only consider a moderate number of long-lived TCP connections such as (huge) file transfers (e.g., [11], [38]), or only examine the environment where all flows are short-lived data transfers (e.g., [6]). There are some studies that did consider some types of heterogeneity. For example, recent work at INRIA has studied the effect of RED on mixes of "bursty" (TCP) and "smooth" (UDP) traffic [24]. However, the types of heterogeneity considered are often limited. In particular, the effect of two-way traffic is rarely examined. Moreover, little efforts have been made on pinpointing the interaction of RED with each particular type of heterogeneity.
- As pointed out in [6], the performance metrics used in previous studies have largely been network-centric measures such as the network link utilization and aggregate TCP throughput. While such information is valuable for network operators and service providers, end users tend to be more interested in the performance received by each individual flow.
- Most previous studies have focused on the performance of RED in absence of the "gentle_" modification. This, of course, is largely due to the fact that the "gentle_" modification was not proposed until very recently. The "gentle_" option makes RED much more robust to the setting of the parameters $\max_{th}$ and $\max_p$. It can significantly reduce the unwanted effects due to parameter misconfiguration. Therefore, it is important to run simulations with the "gentle_" variant of RED.

In this paper, we address these limitations by conducting extensive simulations to explore the effects of RED (with the "gentle_" modification) on the user-perceived end-to-end performance in a heterogeneous environment.

IV. Simulation Setup

To study the performance impact of RED in a heterogeneous environment, we conduct extensive simulations on a variety of network topologies and workload using the ns network simulator [27]. In this section, we give an overview of the network topology and various simulation settings. Then we describe the performance metrics used in evaluation.

A. Simulation Topology

In order to thoroughly investigate the interaction between RED and various types of heterogeneity, we believe it is necessary to keep the simulation topology relatively simple, yet sufficiently representative. Fig. 2 illustrates a simple single-bottleneck network topology that we use for most of our simulations. The bottleneck bandwidth (BW) is either 1.544 Mbps or 10 Mbps. The one-way propagation delay (D) of the bottleneck link is either 25 msec, 50 msec, or 100 msec. As suggested in [38], the bottleneck router has a maximum buffer size (B) of either 1, or 1.5, or 2 times the bandwidth-delay product. (Since our topology is symmetric, the bandwidth-delay product equals $2 \times BW \times D$). All the other links have 10 Mbps capacity and 1 msec one-way propagation delay. Normally, data traffic is only present in the *forward* path (from left to right in Fig. 2). When studying the effect of two-traffic, we also consider data traffic in the *reverse* path (from right to left in Fig. 2).

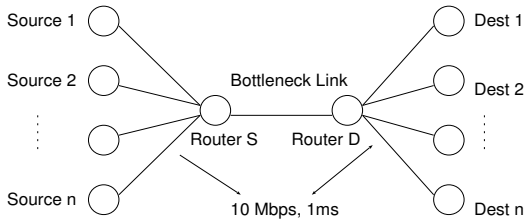


Fig. 2. Simulation topology.

For interest of space, in this paper we only present the results for the scenario where BW = 1.544 Mbps, D = 50 msec, and $B = 2 \times BW \times D$. The results for the other network scenarios are qualitatively similar to what we present here.

B. Queue Management Schemes

In all simulations, the bottleneck router uses either the Drop-Tail FIFO queue management or the RED queue management (with or without support for ECN). As noted in Section II, we closely follows the guidelines by Floyd [16] when configuring various RED parameters. More specifically, we always use $\max_{th} = 3 \min_{th}$, $\max_p = 0.1$, and $w_q = 0.002$. For the last parameter $\min_{th}$, besides the recommended value of 5 packets, we also experiment with two different values based on the buffer size: $B/6$ and $B/9$, where B is the maximum buffer size. The details for different queue management schemes used in this paper are summarized in Table I.

Scheme	Type	$\min_{th}$
dt	DT (Drop-Tail)	N/A
r1	RED w/o ECN	$B/6$
r2	RED w/o ECN	$B/9$
r3	RED w/o ECN	5 packets
r1e	RED with ECN	$B/6$ (same as **r1**)
r2e	RED with ECN	$B/9$ (same as **r2**)
r3e	RED with ECN	5 packets (same as **r3**)

TABLE I
Different queue management schemes.

C. Traffic Source Models

We consider the following four types of traffic sources in our simulations:

Type 1. *Long-lived TCP traffic sources.* Traffic sources of this type belong to FTP sessions with an infinite amount of data to transmit.

Type 2. *Short-lived Web-like TCP traffic sources.* We use the following simple source model to mimic the behavior of Web sessions: Each connection repeatedly make short file transfers. Between two consecutive transfers, there is a think-time that starts after the last byte of the first file has been acknowledged. The transfer size is kept at 30 KB, which is the average web transfer size (including inline images) [23]. The think times are drawn uniformly between 1 and 3 seconds. When restarting data flow after an idle period, the sender always uses the slow start procedure [21] to probe the network available bandwidth. Our model is by no means realistic. However, we believe that it does capture the essence of Web data transfers, that is, a significant amount of time is spent during the slow start phase.

Type 3. *Constant-Bit-Rate (CBR) UDP traffic sources.* To assess the effect of RED on relatively "smooth" real-time traffic, we consider UDP traffic sources sending at a constant bit rate of 24 Kbps. The packet size for the CBR sources is set to 210 bytes, which is the default value in ns.

Type 4. *ON/OFF UDP traffic sources.* The ON/OFF times are drawn from Pareto distributions with the "shape" parameters set to 1.5, which is the default value in ns. The mean ON time is 1 second and the mean OFF time is 2 seconds. During ON times, the sources transmit with a rate of 24 Kbps. Like for the CBR traffic sources, we use the default packet size of 210 bytes for the ON/OFF UDP sources. It has been reported by Park *et al.*[30] that WWW-related traffic tends to be self-similar in nature. Willinger *et al.*[39] show that self-similar traffic may be created by using several ON/OFF UDP sources whose ON/OFF times are drawn from heavy-tailed distributions such as the Pareto distribution. That's why we study ON/OFF traffic in our simulations.

D. TCP Configurations

For all TCP connections, we use the ns TCP/Reno and TCP/Sack1 (TCP with support for *selective acknowledgments* (SACK)) simulation code, which closely models the congestion control behavior of most of the TCP implementations in widespread use. For this study, we disable delayed acknowledgments, although we have repeated several of our experiments with delayed acknowledgments enabled and our results are qualitatively similar.

The size of TCP data packets is set to 500 bytes, which is typical for wide-area TCP connections. The size of the TCP ac-

knowledgments is set to 40 bytes. The timer granularity is set to 100 msec, which is default in `ns`. The maximum congestion window size is set to 100 KB (200 packets). As suggested by Floyd *et al.*[9], in order to explore properties of network behavior unmasked by the specific details of traffic phase effects, *we always add a random processing time at the TCP sender*, which is uniformly chosen between zero and the bottleneck service time for a TCP data packet.

E. Performance Metrics

We use the *goodput* as the primary performance metric for TCP flows. For a single TCP flow, its goodput is defined as the number of *good* bits received by the receiver (excluding unnecessary retransmissions) in unit time. For a set of TCP flows, their *aggregate goodput* is defined as the number of *good* bits received by all receivers in unit time. The *average goodput* for a set of TCP flows is defined as their aggregate goodput divided by the number of flows. Goodput can be easily translated into other metrics such as the completion time. Compared with network-centric like network link utilization , goodput is often what end users really care about.

For UDP flows, we focus on the loss rate for this study. Besides the aggregate loss rate seen by all UDP flows, we also look at the distribution of the loss rate seen by each UDP flow.

V. Simulation Results

In this section, we evaluate the end-to-end performance impact of RED with different types of heterogeneity we identify in Section I.

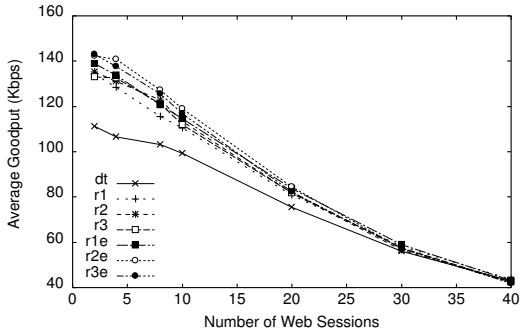
A. Mix of Short-Lived and Long-Lived TCP Connections

We conduct the following experiments to assess the effects of different queue management schemes summarized in Table I on a workload with both long-lived and short-lived TCP connections: In each experiment, a set of foreground Web sessions (Type 2 in Section IV-C) compete with a fixed number of background FTP sessions (Type 1 in Section IV-C). To avoid deterministic behavior, besides adding random processing time at the TCP senders, we also include six telnet sessions competing with the main flows. The inter-arrival times for telnet sessions are drawn from the "tcplib" distribution as implemented in `ns`. Each simulation run lasts 200 sec. All connections start randomly from within the initial two seconds. We record the average goodput for both Web sessions and FTP sessions in the final 150 sec. For each simulation configuration, we report the mean [1] of 5 runs of an experiment with different random seeds.

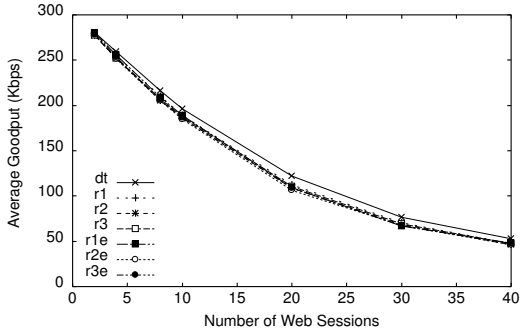
Fig. 3 summarizes the results for TCP/Reno when the number of short-lived TCP connections varies from 2 to 40 and the number of FTP sessions is kept at 5. As we can see, the Web sessions receive 10-30% higher average goodput with RED than with DT, regardless of the RED parameters used. As a result, the FTP sessions receives slightly lower goodput with RED than with DT (because the total amount of bandwidth consumed by all flows remains constant after the bottleneck link get saturated).

We originally conjectured that this is because with DT, short flows tend to get more than their share of losses. This is not

[1] The variation is in general very small compared to the mean.



(a) Average goodput of short-lived TCP connections (Web sessions).



(b) Average goodput of long-lived TCP connections (FTP sessions).

Fig. 3. Effect of RED on the mix of short-lived and long-lived TCP connections. The number of short-lived TCP connections varies from 2 to 40 while the number of competing FTP sessions is kept at 5. 6 telnet sessions are used to avoid deterministic behavior. TCP/Reno is used for all TCP connections.

the case with RED, which distributes losses uniformly across all flows. Consequently, Web sessions receives higher goodput with RED than with DT. However, this turns out to be *not the case*. With a large number of TCP flows, the statistical multiplexing level is very high. Consequently, there is little difference in the loss rates experienced by the Web sessions and the FTP sessions even with DT. This can be illustrated by Fig. 4, which compares the packet drops vs. the throughput received by each individual flow in a simulation run. (The same type of diagram is also used by Floyd *et al.*in [9]). As we can see, there is not much difference between **dt** and **r1**. (The results for the other RED configurations are very similar.)

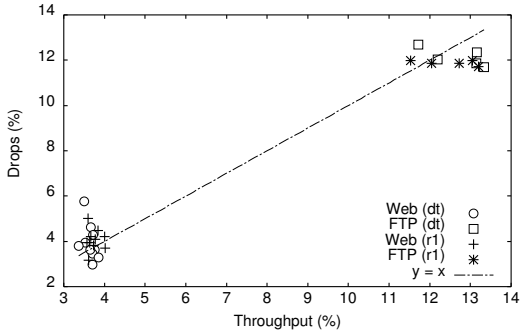


Fig. 4. Packet drops vs. throughput for each flow in one simulation run. 10 Web sessions compete with 5 FTP sessions. 6 telnet sessions are used to avoid deterministic behavior. The queue management scheme is either **dt** or **r1** as defined in Table I. TCP/Reno is used for all TCP connections.

The actual reason is as follows. RED reduces the average queue length, and consequently, increases the packet loss rate. The increase in packet loss rate has a greater impact on the FTP sessions, which typically have more outstanding packets in a roundtrip time than the Web sessions. (Note that as the num-

ber of outstanding packets increases, so does the probability of *getting at least one loss in one roundtrip time*.) Consequently, the FTP sessions receive lower throughput, which in turn creates more bandwidth available to the Web sessions. Similar effects can be achieved by reducing the buffer size at a Drop-Tail gateway. As illustrated in Fig. 5, as we vary the ratio of the buffer size over the bandwidth-delay product (br) from 1.4 down to 0.2, the average goodput received by the Web sessions experiences a 10-30% increase.

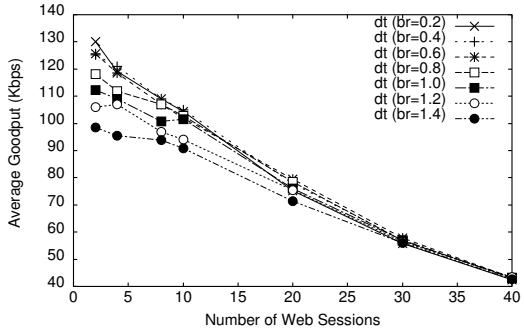


Fig. 5. The average goodput of Web sessions. 10 Web sessions compete with 5 FTP sessions. 6 telnet sessions are used to avoid deterministic behavior. The bottleneck router uses the Drop-Tail FIFO queue management. The ratio of the buffer size over the bandwidth-delay product (br) varies from 0.2 to 1.4. TCP/Reno is used for all TCP connections.

Fig. 6 summarizes the results for the same experiment in Fig. 3 except that TCP/Sack1 is used for all connections instead of TCP/Reno. For brevity, we only show the average goodput for the Web sessions. The performance for the FTP sessions can be easily inferred because the total amount of bandwidth consumed by all flows is almost constant after the bottleneck link get saturated. The most interesting part of Fig. 6 is that with **dt**, the Web sessions receive 10% lower goodput when all flows use TCP/Sack1 instead of TCP/Reno. In contrast, SACK makes little difference in terms of the average goodput with other queue management schemes.

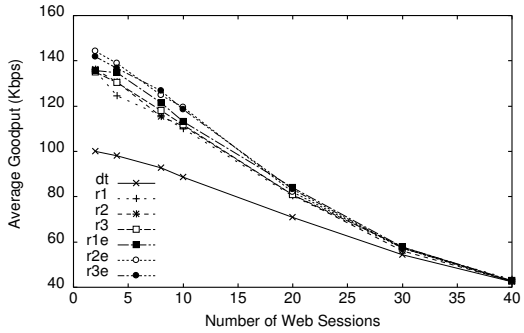


Fig. 6. Average goodput of the Web sessions. The simulation settings are the same as in Fig. 3 except that TCP/Sack1 instead of TCP/Reno is used for all TCP connections.

We believe such discrepancy can be explained as follows. Compared with the Web sessions, the FTP sessions typically have larger congestion windows, and consequently, higher probability of getting multiple losses in a roundtrip time. When packet losses are bursty (as is the case with DT), TCP/Reno often needs to use timeout to recover from multiple losses in a roundtrip time. With SACK, this is no longer necessary. Therefore, with DT, FTP sessions benefit more from SACK than the Web sessions. As a result, FTP sessions get higher average goodput, which in turn reduces the amount of bandwidth available to Web sessions. In contrast, with RED, losses tend to be less bursty than with DT. In this case, even TCP/Reno can often recover from multiple packet losses in the same roundtrip time. Consequently, the benefit of SACK is much smaller.

To summarize, short-lived TCP flows tend to receive higher goodput with RED than with DT when competing with long-lived TCP flows. This is desirable because in the real Internet, short data transfers typically belong to interactive Web browsing sessions, which is delay-sensitive. Meanwhile, with DT, SACK tends to benefit long-lived flows more than short-lived flows. In contrast, with RED, SACK has very little performance impact.

B. Mix of TCP and UDP Traffic

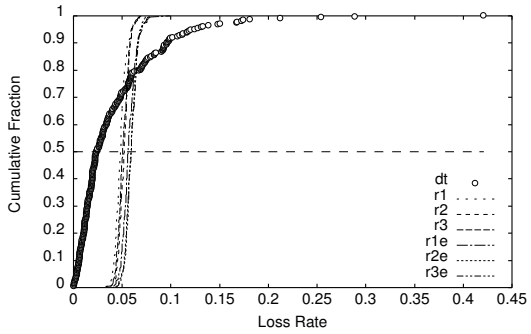
In this section, we study the performance impact of RED in an environment with both TCP and UDP traffic. We focus on the loss rate for the UDP traffic.

Recent work at INRIA [24] has shown that RED can significantly increase the loss rate for "smooth" CBR UDP traffic. However, [24] only look at the average loss rate aggregated over all UDP flows. We are more interested in the distribution for the loss rate seen by each individual UDP flow. We conduct the following experiment with a set of CBR UDP flows (Type 3 in Section IV-C) competing with FTP sessions (Type 1 in Section IV-C). Each run of the experiment lasts 200 sec. We record the loss rate in the final 150 sec for each UDP flow. Then we plot the cumulative distribution for all such loss rates recorded for 15 simulation runs.

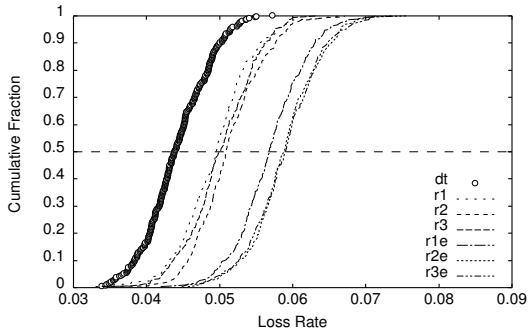
Fig. 7 summarizes the results for 20 CBR UDP flows competing with 10 FTP sessions. (We have also conduct the same experiment with different number of UDP and FTP sessions. Our results are qualitatively similar.) We consider both fixed and random inter-arrival times for the CBR traffic. We make two observations:

- The use of SACK or ECN for TCP flows can significantly increase the loss rate for UDP traffic. The use of RED without ECN can also increase the loss rate for UDP traffic, but the relative impact is always smaller or comparable to the use of SACK with DT. This is because with RED/ECN/SACK, TCP flows are less likely to have timeouts, and consequently, become more aggressive.
- When the CBR traffic has fixed inter-arrival times, there is a much bigger tail ($> 15\%$) at the high loss rate region (loss rate > 0.1) with DT than with RED, which means a significant number of UDP flows suffer from high loss rate. Currently we are still investigating the exact cause for this. But reducing the tail at the high-loss region is clearly beneficial for applications.

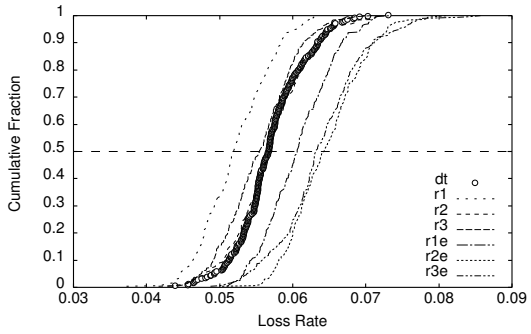
Fig. 8 summarizes the results for an experiment with the same simulation settings as in Fig. 7 except that ON/OFF UDP traffic replaces the CBR UDP traffic. Again, we find that the use of RED with ECN and/or SACK can significantly increase the loss rate for UDP traffic compared to the case when TCP/Reno and DT are used. This is because TCP becomes more aggressive with SACK and/or ECN. On the other hand, the use of RED without ECN actually reduces the loss rate for UDP traffic. This is because RED can reduces the bias against bursty traffic by distributing losses uniformly across all connections. (Note that



(a) CBR traffic with fixed inter-arrival times. FTP uses TCP/Reno.



(b) CBR traffic with random inter-arrival times. FTP uses TCP/Reno



(c) CBR traffic with random inter-arrival times. FTP uses TCP/Sack1.

Fig. 7. Cumulative distribution for the loss rate experienced by CBR UDP flows. 20 CBR UDP flows (Type 3 in Section IV-C) compete with 10 FTP sessions (Type 1 in Section IV-C). The CDF is obtained from 15 simulation runs. In (a) the inter-arrival-times remain constant. In (b) and (c), the inter-arrival times are uniformly distributed between $0.5 \cdot T$ and $1.5 \cdot T$, where T is the inter-arrival time in (a).

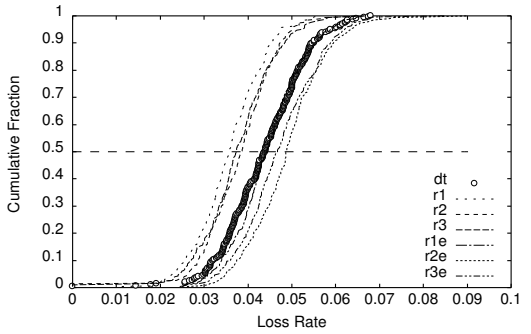
the ON/OFF UDP traffic is more bursty than TCP traffic in that it does not respond to network congestion.)

To summarize, the use of SACK and/or RED with ECN for TCP flows always significantly increases the loss rate for UDP traffic. The use of RED without ECN tends to reduce the loss rate for bursty UDP traffic, as well as increase the loss rate for smooth UDP traffic (The increase is less significant compared to the case when SACK is used with DT). Moreover, RED can reduce the tail at the high loss rate region for CBR traffic with fixed inter-arrival times, which is beneficial.

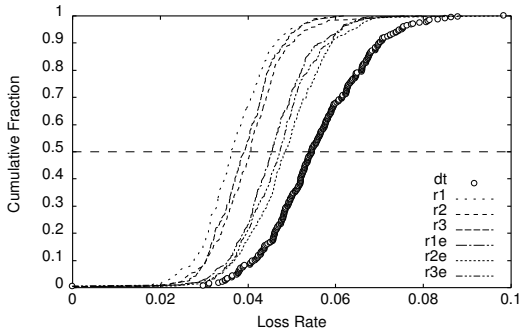
C. Mix of ECN-Capable and Non-ECN-Capable Traffic

To assess the effect of RED on mixes of both ECN-capable and non-ECN-capable traffic, we consider a set of $2 \cdot n$ TCP flows competing for the same bottleneck link. Half of the flows are ECN-capable, while the other half are not. We then report the ratio between the average goodput for ECN-capable flows over the average goodput for non-ECN-capable flows.

Fig. 9 shows the results when all flows are long-lived FTP



(a) TCP/Reno is used for FTP sessions



(b) TCP/Sack1 is used for FTP sessions

Fig. 8. Cumulative distribution for the loss rate experienced by ON/OFF UDP flows. The simulation settings are the same as in Fig. 7 except that ON/OFF UDP flows (Type 4 in Section IV-C) are used instead of the CBR traffic.

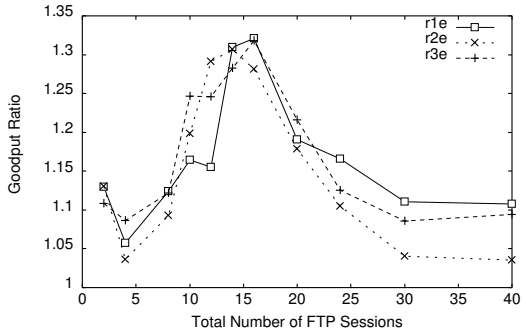
sessions (Type 1 in Section IV-C). n varies from 1 to 20 (thus the total number of competing flows varies from 2 to 40). It is evident from the figure that ECN-capable sources consistently out-performs non-ECN-capable sources by 5-30%. Meanwhile, as the congestion level increases, the goodput ratio initially increases and later decreases. The peak occurs when on average each flow has roughly 3-4 outstanding packets during each round trip. This is reasonable because when the average congestion window size (which is the number of outstanding packet during each RTT) decreases, the relative performance benefit of avoiding an early drop at the RED gateway increases. But when the average congestion window drops below 3, *fast retransmission* breaks down even with a single loss. Consequently, the timeout penalty becomes the dominant factor, which limit the relative performance benefit of ECN.

Fig. 10 summarizes the results for the Web sessions (Type 2 in Section IV-C). As we can see, the shape of the curve is very similar to Fig. 9 (a), except that the peak goodput ratio is much smaller. This is because the short-lived flows spend a significant amount of time probing the network available bandwidth during slow start, which limit the relative benefit of ECN.

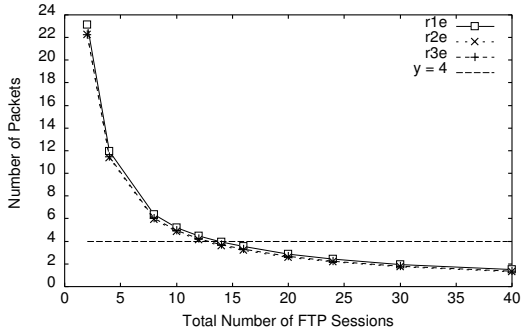
To summarize, ECN-capable TCP flows almost always have higher goodput than non-ECN-capable TCP flows. For bulk transfers, the relative performance benefit is greatest when on average each flow has around 3-4 outstanding packets per roundtrip time. For short-lived flows, the relative benefit of ECN is much smaller.

D. Effect of Different RTT's

In this section, we explore the effect of different queue management schemes on flows with different RTT's. We start with the simplist scenario in which there are only two different RTT



(a) The goodput ratio between two groups



(b) Average number of outstanding packets in an RTT for each flow.

Fig. 9. Effect of RED on an ensemble of n ECN-capable and n non-ECN-capable FTP sessions. n varies from 1 to 20. TCP/Reno is used for all connections. (The results for TCP/Sack1 is very similar.)

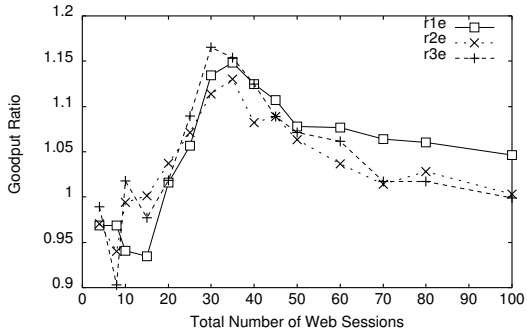


Fig. 10. The ratio between the average goodput of n ECN-capable and n non-ECN-capable Web sessions. n varies from 1 to 50. TCP/Reno is used for all connections. (The results for TCP/Sack1 is very similar.)

values and all flows are long-lived FTP sessions that can reach their steady state (in Section V-D.1). We then consider more complicated scenarios where either there are more that two different RTT values (in Section V-D.2) or all flows are short-lived Web sessions, which can hardly ever reach their steady state (in Section V-D.3). **We use TCP/Reno for all simulations in this section.**

D.1 Bulk Transfers with 2 Different RTT's

We divide all TCP flows into two equal-sized groups G_1 and G_2. All flows within the same group G_i have the same two-way propagation delay PD_i (i = 1, 2). We vary PD_1 while keeping PD_2 fixed. This can be achieved by properly adjusting the one-way propagation delay on the high-bandwidth (10 Mbps) links in Fig. 2. For each simulation run, we record the average queuing delay (Q) experienced by all packets. Then we can estimate the average roundtrip time for flows in group G_i, as $RTT_i = PD_i + Q$. We also record $Goodput_i$, the average goodput for flows in group G_i. The goal is to study how $(Goodput_2/Goodput_1)$ changes with respect to (RTT_1/RTT_2).

Fig. 11 shows the results 6 competing FTP sessions with the Drop-Tail FIFO queue management. The bottleneck link has T1 capacity and 50 msec one-way propagation delay. PD_2 is kept at 102 msec. (We do not consider scenarios in which the RTT ratio is ≥ 5, which is unlikely to happen in the real Internet.) As we can see, the goodput ratio always satisfies Equation (1). (Extensive simulations with many different settings also give the same result.)

$$0.5 \times \left(\frac{RTT1}{RTT2}\right)^2 \leq \frac{Goodput_2}{Goodput_1} \leq \left(\frac{RTT1}{RTT2}\right)^2 \qquad (1)$$

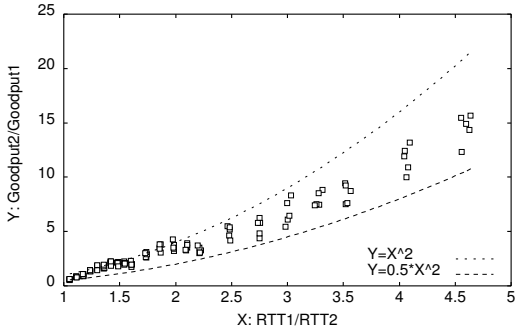


Fig. 11. The effect of Drop-Tail queue (**dt** in Table I) on 6 FTP sessions with 2 different RTT's. The bottleneck link has T1 capacity and 50 msec one-way propagation delay. PD_1 varies while PD_2 is kept at 102 msec.

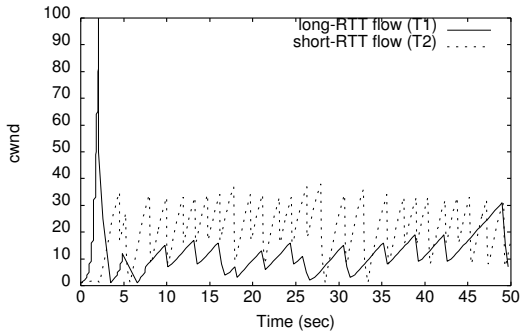


Fig. 12. The congestion window evolution of two TCP flows T_1 and T_2 with Drop-Tail FIFO queue management. $RTT_1 > RTT_2$.

To explain (1), we need to thoroughly understand the synchronization effect of Drop-Tail gateways. For brevity, let us only consider two competing TCP flows. All our analysis still applies when there are more than two competing TCP flows. Fig. 12 illustrates the evolution of the congestion window size ($cwnd$) for two competing TCP flows T_1 and T_2 (with $RTT_1 > RTT_2$) under Drop-Tail FIFO queue management. As we can see, the synchronization effect on T_1 and T_2 is more complicated than the frequently mentioned "global synchronization" for TCP connections with the same RTT [36]. For T_2, whenever the bottleneck buffer becomes full, it will get a loss within one roundtrip time (RTT_2) when it increases its congestion window. It takes another roundtrip for T_2 to detect the loss by 3 duplicated ACK's and reduce its sending rate by halving its $cwnd$. It takes yet a third roundtrip for such rate reduction to actually take effect. Therefore, on average it takes roughly $2.5 \times RTT_2$ for the rate of T_2's packets arriving at the bottleneck router to decrease. When T_1 increases its $cwnd$, the extra packet sent by T_1 will

get dropped only if it arrives at the bottleneck queue before the arrival rate of packets sent by T_2 decreases. With the random processing time at the sender, this extra packet can arrive at the bottleneck queue at any time within RTT_1 after the queue becomes full. Therefore, we can estimate the drop probability for this extra packet as $\min(1, 2.5 \times RTT_2/RTT_1)$, which is within $[0.5, 1]$ when $RTT_1/RTT_2 \in [1, 5]$. Or equivalently,

$$\frac{\text{\#drops seen by } T_1 \text{ in unit time}}{\text{\#drops seen by } T_2 \text{ in unit time}} \in [0.5, 1] \quad (2)$$

Assuming that at steady state, the $cwnd$ of connection T_i grows from W_i to $2 \times W_i$ during each epoch, then T_i sees a drop every $W_i \times RTT_i$. Consequently, (2) becomes

$$\frac{W_2 \times RTT_2}{W_1 \times RTT_1} \in [0.5, 1] \quad (3)$$

Meanwhile, the goodput for T_i can be approximated as:

$$Goodput_i = \frac{3 \times W_i}{2 \times RTT_i} \quad i = 1, 2. \quad (4)$$

From (3) and (4), we immediately get (1). □

Now we considered the effect of RED queue management. It is well-known that $Goodput_i$ is roughly inversely proportional to $RTT_i\sqrt{p_i}$, where p_i is the packet loss rate for flow i [29]. With RED, different flows roughly experience the same packet loss rate under steady state. Consequently, we have:

$$\frac{Goodput_2}{Goodput_1} \approx \frac{RTT_1}{RTT_2} \quad (5)$$

The validity of (5) can be best illustrated by Fig. 13.

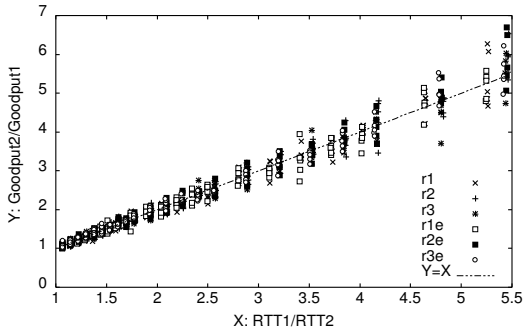


Fig. 13. The effect of RED gateway on FTP sessions with 2 different RTT's. The simulation settings are the same as in Fig. 11 except that the bottleneck router uses RED queue management.

D.2 Bulk Transfers with 8 Different RTT's

In this section, we consider 8 competing FTP sessions F_i (i=1,2,...8) all with different RTT's. The two-way propagation delay for F_1 is kept at 102 msec. The propagation delays for all the other FTP sessions are uniformly chosen between 102 msec and 300 msec during each simulation run. The configurations for the bottleneck link and the buffer size are the same as before. For each run of the experiment, we record goodput ratios $Goodput_1/Goodput_i$ and the RTT ratios RTT_i/RTT_1 (i = 2,3,...,8). We then make a scatter plot of all the data points $(RTT_i/RTT_1, Goodput_1/Goodput_i)$ obtained from 15 runs for each queue management scheme. The results are summarized in Fig. 14.

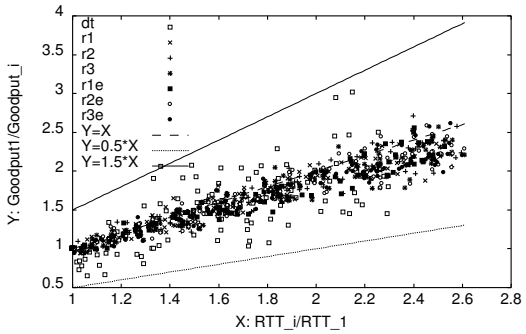


Fig. 14. Comparison between the effect of RED gateway and the dropping tail gateway on 8 TCP flows with all different RTT's.

As we can see, with **dt**, the bias against long-RTT flows is smaller that the case with only two different RTT's. The throughput ratios are centered around line $Y = X$ and are well-bounded by two lines $Y = 1.5X$ and $Y = 0.5X$. In comparison, with RED, the throughput ratios are clustered much closer to line $Y = X$. This suggests that RED is more fair that DT. To quantify the fairness of different queue management schemes for flows with different RTT's, we use the *normalized fairness ratio* [1], which is defined as follows.

$$f = \frac{(\sum_{i=1}^{n} Goodput_i \times RTT_i)^2}{n \sum_{i=1}^{n} (Goodput_i \times RTT_i)^2}$$

Fig. 15 shows the normalized fairness ratio for different queue management policy. As we can see that DT is significantly less fair compared to RED.

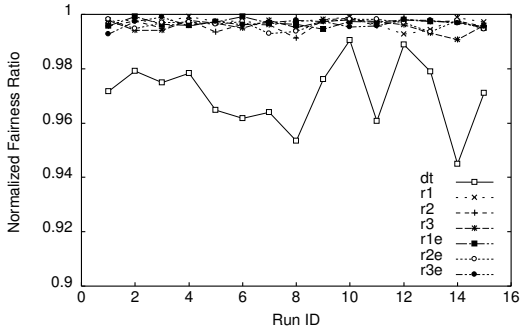


Fig. 15. Normalized fairness ratio using different queue management policies.

D.3 Web Sessions with 2 Different RTT's

Fig. 16 shows the results for 30 Web sessions (Type 2 in Section IV-C) with 2 different RTT's. As we can see, no matter what queue management scheme is used, $Goodput_2/Goodput_1 \approx RTT_1/RTT_2$. This is mainly because for short data transfers, the required roundtrip times is dominated by the slow start procedure. Unless two flows have significantly different loss rates, they tend to require similar number of roundtrip times. Consequently, the goodput ratio is inversely proportional to the RTT ratio.

To summarize, long-RTT bulk transfers in general tend to have higher goodput with RED than with DT. For Web sessions, RED makes very little difference.

E. Effect of Two-way Traffic

In this section, we evaluate the effects of different queue management schemes when there is data traffic in both the forward

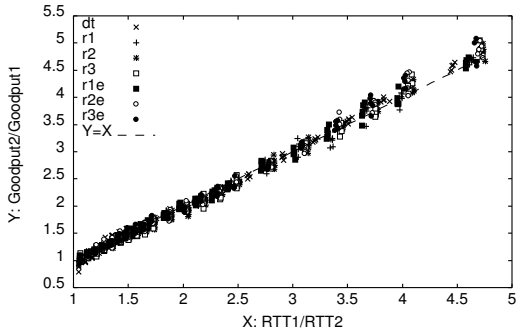


Fig. 16. Effect of different queue management schemes on 30 Web sessions with 2 different RTT's.

and the reverse path. We consider three cases: 1. the data traffic on both directions belongs to FTP sessions (in Section V-E.1); 2. the data traffic in the forward path belongs to Web traffic but the data traffic in the reverse path belongs FTP sessions (in Section V-E.2); and 3. the data traffic on both directions belongs to Web sessions (in Section V-E.3).

E.1 Bulk Transfers in Both Directions

We first consider the scenario where the forward and the reverse paths have the same congestion level. This can be achieved by keeping the same number of FTP sessions in both directions.

Fig. 17 shows the average goodput for FTP sessions (using TCP/Reno) in the forward path. The goodput for FTP sessions in the reverse path is very similar due to the symmetric topology and the symmetric congestion level. As we can see, the average goodput is slightly higher with DT than with RED when the number of competing FTP sessions is small. This is not surprising, because RED reduces the bottleneck queue length by increases the loss rate. When the number of flows is small, this leads to link under-utilization. As the number of FTP sessions increase, the bottleneck link becomes highly utilized. Consequently, there is little difference in the average goodput no matter which queue management scheme is used. We also conduct the same experiment with TCP/Sack1. The results are similar.

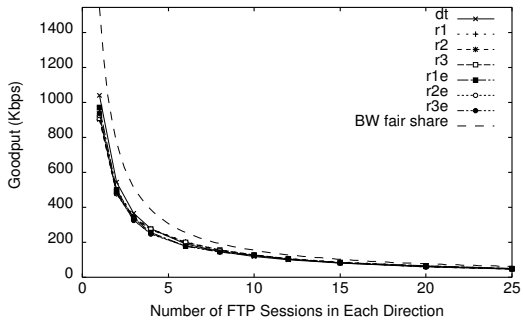
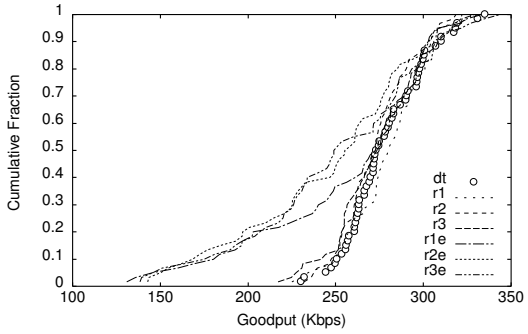
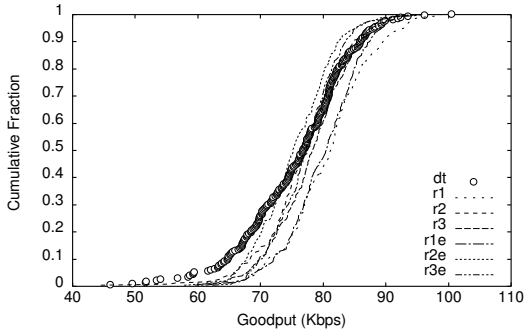


Fig. 17. Average goodput for FTP sessions in the forward direction. All flows use the TCP/Reno flavor without support for SACK.

Besides the average goodput, we are also interested in the distribution for the goodput received by each individual FTP session. For this purpose, we keep the number of FTP sessions in each direction to be either 4 or 16. For each run of the experiment, we record the goodput received by each individual FTP session. The we compute the cumulative distribution for the goodput received by each flow in 15 simulation runs. The results are summarized in Fig. 18. As we can see, when the number of competing flows is small, the average goodput with DT is higher than RED plus ECN, but comparable to RED without ECN. When the congestion level is high, DT gives worse performance than RED.



(a) 4 FTP sessions in each direction.



(b) 16 FTP sessions in each direction.

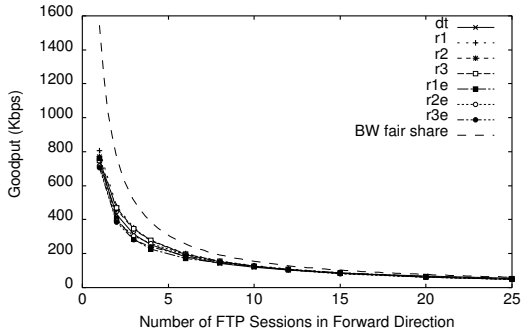
Fig. 18. Cumulative distribution for the goodput received by each individual FTP session in the forward direction. TCP/Reno is used by all FTP sessions. The distribution is obtained from 15 simulation runs.

Now we consider the scenario in which the forward and the reverse paths have different congestion levels. More specifically, we vary the number of FTP sessions in the forward direction while keeping the number of FTP sessions in the reverse direction to be either 5 or 10.

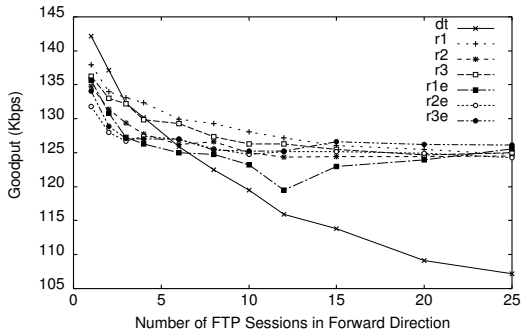
The results are summarized in Fig. 19. As illustrated in Fig. 19(a), the goodput received by FTP sessions in the forward path is qualitatively similar to the results given in Fig. 17. In particular, there is very little difference in terms of average goodput with different queue management schemes. This is not surprising because the congestion level at the reverse direction largely remains constant. As the forward path becomes saturated, the average goodput only depends on the number of competing flows. Compared to the forward path, the reverse path is more interesting. As we can see in Fig. 19(b), with DT, the average goodput received by FTP sessions in the reverse path continuously decreases as the congestion level in the forward path increases. This is not the case with RED, where the goodput initially decreases but soon stabilizes at a level much higher than the average goodput with DT.

E.2 Web Sessions in Forward Direction, Bulk Transfers in Reverse Direction

Fig. 20 summarizes the results when the number of Web sessions in the forward path varies from 1 to 60 and the number of FTP sessions in the reverse path is kept at 5. As we can see, different queue management scheme makes very little difference in terms of average goodput for the Web sessions on the forward.

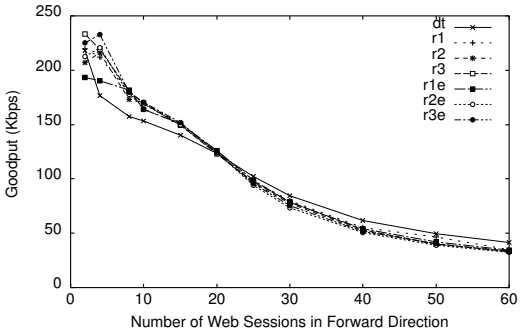


(a) Average goodput for FTP sessions in the forward path.

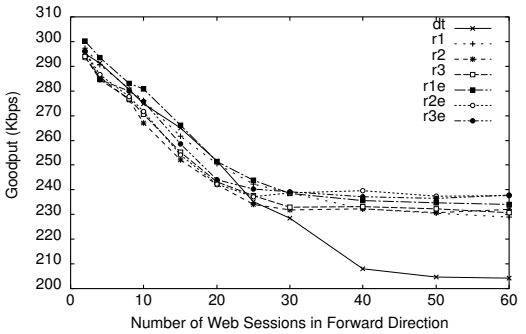


(b) Average goodput for FTP sessions in the reverse path.

Fig. 19. Performance evaluation for the case when the number of FTP sessions in the forward direction varies from 1 to 25 and the number of FTP sessions in the reverse path is kept at 10. All flows use TCP/Reno.



(a) Average goodput for Web sessions in the forward path.



(b) Average goodput for FTP sessions in the reverse path.

Fig. 20. Performance evaluation for the case when the number of Web sessions in the forward path varies from 1 to 60 and the number of FTP sessions in the reverse path is kept at 5. TCP/Reno is used for all flows.

For the FTP sessions in the reverse path, with RED, the average goodput decreases but soon stabilizes as the congestion level increases in the forward path. This is not the case with DT, where the average goodput continues to decrease.

E.3 Web Sessions in Both Directions

Now we consider the case when the data traffic in both directions are Web sessions.

Fig. 21 shows the average goodput for the Web sessions in the forward path when the congestion levels in both directions are similar. Again, there is very little difference in goodput with different queue management policies.

Fig. 22 shows the results for the case when the forward path and the reverse path have different number of Web sessions. The number of Web sessions in the forward path varies from 1 to 60 while the number of Web sessions in the reverse path is kept at 10. As we can see, the results is qualitatively similar to Fig. 20.

We can summarize the results with two-way traffic as follows.

- When the congestion level in the ACK path is high, TCP tends to get higher goodput with RED than with DT.
- When the congestion level in the ACK path is low but the congestion level in the data path is sufficiently high, TCP tends to get comparable goodput with RED and with DT.
- When the congestion levels in both the ACK path and the data path are low, TCP tends to get slightly lower but still comparable goodput with RED than with DT.

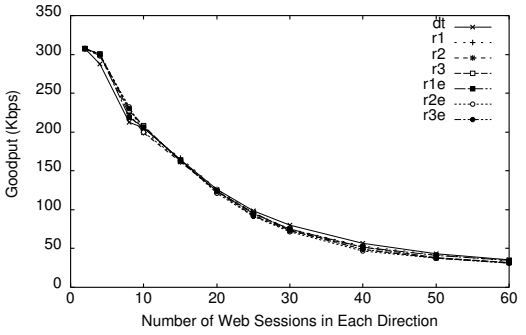
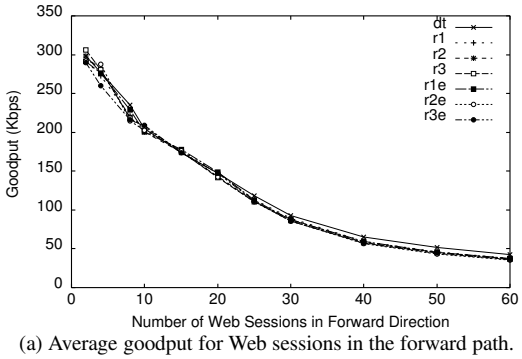


Fig. 21. Average goodput for the Web sessions in the forward direction. The number of Web sessions in both directions varies from 1 to 60. All flows use TCP/Reno.

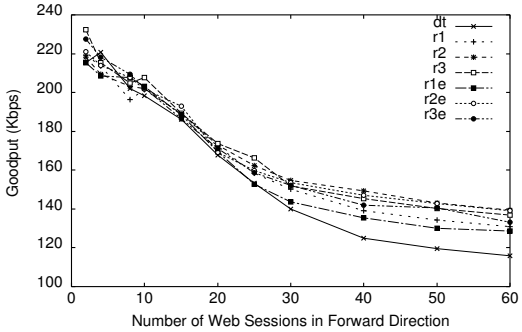
- When competing with long-lived TCP flows, short-lived flows tend to get higher goodput with RED than with DT. This is desirable because short-lived flows typically belong to interactive Web browsing sessions.
- When competing with TCP traffic, bursty UDP traffic tends to get lower loss rate with RED than with DT. Meanwhile, smooth UDP traffic tends to get higher loss rate with RED than with DT, but such increase is often less significant compared to the case when competing with TCP/Sack traffic under DT queue management.
- When ECN-enabled traffic compete with non-ECN-enabled traffic, ECN-enabled traffic can receive up to 30% higher goodput.
- For TCP flows with different RTT's, RED in general tends to reduce the bias against long-RTT bulk transfers. For short data transfers with different RTT's, RED neither helps nor hurts.
- When the ACK path is congested, TCP gets higher goodput with RED than with DT.

VI. Conclusions and Future Work

In this paper, we use extensive simulations to explore the interaction between RED and five types of heterogeneity, as well as the impact of such interaction on the user-perceived end-to-end performance. Our results show that:



(a) Average goodput for Web sessions in the forward path.



(b) Average goodput for Web sessions in the reverse path.

Fig. 22. Average goodput for the Web sessions in the reverse path. The number of Web sessions in the forward path varies from 1 to 60, while the number of Web sessions in the reverse path is kept at 10. TCP/Reno is used for all Web sessions.

Therefore, we conclude that overall RED improves performance at least for the types of heterogeneity we have considered.

As for future work, we are interested in considering other types of heterogeneity such as the presence of random loss and multiple congested links. We would also like to understand the aggregate effect of different types of heterogeneity.

VII. Acknowledgments

Thanks to Sally Floyd for helpful discussions.

References

[1] A. Aagarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing," *Proc. IEEE INFOCOM '2000*, Mar. 2000.

[2] F. Anjum and L. Tassiulas, "Balanced-RED: An Algorithm to Achieve Fairness in the Internet," *Proc. IEEE INFOCOM '99*, Mar. 1999.

[3] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Apr. 1998.

[4] 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, Dec. 1997.

[5] K. Claffy, H. Braun, and G. Polyzos, "A Parameterizable Methodology for Internet Traffic Flow Profiling," *IEEE Journal on Selected Areas in Communications (JSAC)*, 13(8):1481-1494, Oct. 1995.

[6] M. Christiansen, K. Jeffay, D. Ott, and F. Smith, "Tuning RED for Web Traffic," *Proc. SIGCOMM '2000*, Aug. 2000.

[7] O. Elloumi and H. Afifi, "RED Algorithm in ATM Networks," ftp://ftp.rennes.enst-bretagne.fr/pub/reseau/afifi/red-atm.ps , Technical Report, Jun. 1997.

[8] V. Firoiu, and M. Borden, "A Study of Active Queue Management for Congestion Control," *Proc. IEEE INFOCOM '2000*, Mar. 2000.

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

[10] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, 1(4):397-413, Aug. 1993.

[11] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A Self-Configuring RED Gateway," *Proc. IEEE INFOCOM '99*, Mar. 1999.

[12] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: A New Class of Active Queue Management Algorithms," University of Michigan Technical Report CSE-TR-387-99, Apr. 1999.

[13] S. Floyd, ECN Web Page, http://www.aciri.org/floyd/ecn.html .

[14] S. Floyd, RED Web Page, http://www.aciri.org/floyd/red.html .

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

[16] S. Floyd, "RED: Discussions of Setting Parameters," http://www.aciri.org/floyd/REDparameters.txt , Nov. 1997.

[17] S. Floyd, "Recommendation on Using the "gentle_" Variant of RED," http://www.aciri.org/floyd/red/gentle.html, Mar. 2000.

[18] S. Floyd, "A Report on Some Recent Developments in TCP Congestion Control," http://www.aciri.org/floyd/papers/TCPreport.ps , in submission, Jun. 2000.

[19] A. Feldmann, J. Rexford, and R. Caceres, "Efficient Policies for Carrying Web Traffic Over Flow-Switched Networks," *IEEE/ACM Transactions on Networking*, pp. 673-685, Dec. 1998.

[20] S. Gribble, and E. Brewer, "System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace," *Proc. USITS '97*, Dec. 1997.

[21] V. Jacobson and M. Karels, "Congestion Avoidance and Control," *Proc. SIGCOMM '88*, Aug. 1988.

[22] D. Lin and R. Morris, "Dynamics of Random Early Detection," *Proc. SIGCOMM '97*, Sep. 1997.

[23] B. Mah, "An Empirical Model of HTTP Network Traffic," *Proc. IEEE INFOCOM '97*, Apr. 1997.

[24] M. May, T. Bonald, and J. Bolot, "Analytic Evaluation of RED Performance," *Proc. IEEE INFOCOM '2000*, Mar. 2000.

[25] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons Not to Deploy RED," *Proc. IWQoS '99*, Mar. 1999.

[26] V. Misra, W. Gong, and D. Towsley, "A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," *Proc. SIGCOMM '2000*, Aug. 2000.

[27] UCB/LBNL/VINT Network Simulator - `ns` (version 2), 1997. http://www.isi.edu/nsnam/ns/

[28] T. Ott, T. Lakshman, and L. Wong, "SRED: Stabilized RED," *Proc. IEEE INFOCOM '99*, Mar. 1999.

[29] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and Its Empirical Validation." *Proc. SIGCOMM '98*, Sep. 1998.

[30] K. Park, G. Kim, and M. Crovella, "On the Relationship between File Sizes, Transport Protocols and Self-Similar Network Traffic," *Proc. ICNP '96*, Oct. 1996.

[31] V. Padmanabhan and J. Mogul, "Improving HTTP Latency," *Proc. 2nd International World Wide Web Conference*, Oct. 1994.

[32] V. Rosolen, O. Bonaventure and G. Leduc, "Impact of Cell Discard Strategies on TCP/IP in ATM UBR Networks," *Proc. 6th Workshop on Performance Modelling and Evaluation of ATM Networks (IFIP ATM '98)*, Jul. 1998.

[33] V. Rosolen, O. Bonaventure and G. Leduc, "A RED Discard Strategy for ATM Networks and Its Performance Evaluation with TCP/IP Traffic," *ACM Computer Communication Review*, Jul. 1999.

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

[35] K. Ramakrishnan and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks," *ACM Transaction on Computer Systems*, 8(2):158-181, May 1990.

[36] S. Shenker, L. Zhang, and D. Clark, "Some Observations on the Dynamics of a Congestion Control Algorithm," *ACM Computer Communication Review*, 20(5):30-39, Oct. 1990.

[37] K. Thompson, G. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network*, 11(6):10-23, Nov. 1997.

[38] C. Villamizar and C. Song, "High Performance TCP in ANSNET," *ACM Computer Communications Review*, 24(5):45-60, Oct. 1994.

[39] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *Proc. SIGCOMM '95*, Aug. 1995.

[40] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy, "Organization-Based Analysis of Web-Object Sharing and Caching," *Proc. USITS '99*, Oct. 1999.

[41] L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a congestion control Algorithm: The Effects of Two-Way Traffic," *Proc. SIGCOMM '91*, Sep. 1991.