Emulating AQM from End Hosts

Sumitha Bhandarkar [*], A. L. Narasimha Reddy, Yueping Zhang, and Dmitri Loguinov [†]
Texas A&M University, College Station, TX 77843
{sumitha,reddy}@ece.tamu.edu, {yueping,dmitri}@cs.tamu.edu

ABSTRACT

In this paper, we show that end-host based congestion prediction is more accurate than previously characterized. However, it may not be possible to entirely eliminate the uncertainties in congestion prediction. To address these uncertainties, we propose Probabilistic Early Response TCP (PERT). PERT emulates the behavior of AQM/ECN, in the congestion response function of end-hosts. We present fluid-flow analysis of PERT/RED and PERT/PI, versions of PERT that emulate router-based RED and PI controllers. Our analysis shows that PERT/RED has better stability behavior than router-based RED. We also present results from ns-2 simulations to show the practical feasibility of PERT. The scheme presented here is general and can be used for emulating other AQM algorithms.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols—*TCP Congestion Control.*

General Terms

Algorithms, design, experimentation, measurement, performance, theory

Keywords

Congestion avoidance, early congestion response, delay-based congestion response

1. INTRODUCTION

Congestion control in TCP is provided by an additive increase multiplicative decrease algorithm. In the absence of congestion, the congestion window is increased by one packet per RTT to probe for available bandwidth[1]. When congestion is perceived, the congestion window is decreased by half to relieve the congestion. The only mechanism used by current standard TCP variants to detect congestion is the loss of a packet. Hence, even after the link is saturated and the bottleneck link buffers start to fill up, the congestion window continues to *increase*, until eventually the buffers overflow, which results in packet loss.

The need for response to the onset of congestion – i.e., to have the flows back-off from active probing when the queues start to fill up – has been discussed abundantly in literature. Two rivaling approaches have been suggested for addressing the problem. One school of thought has been that the routers at the bottleneck links know when the buffers are getting full and so are in the best position to judge the onset of congestion. Once congestion is detected, they inform the end-host implicitly by either a binary signal (e.g., [2, 14, 16, 19, 25]) or a non-binary signal (e.g., [18, 32]) which informs the senders of what their sending rate should be. This is the area broadly known as *Active Queue Management* (AQM).

The other school of thought treats the network as a black box and tries to detect network congestion from end-hosts. In these schemes, the sender tries to determine the level of congestion in the network based on information contained in the round trip time or the observed throughput and responds to it in an effort to prevent packet losses. In this approach, the senders do not expect any support from routers. We refer to the work in this area as *end-host delay-based congestion avoidance.* Some of the representative schemes in this area are CARD [17], TRI-S [29], DUAL [30], Vegas [7], and CIM [21].

Both mechanisms have their advantages and disadvantages. While it is true that routers are in the best position to detect congestion in the network, deploying experimental AQM mechanisms in the core routers has not been easy. Overcoming the technical challenges related to the implementation of these schemes in routers, while retaining the speed and efficiency at which they operate, is probably an easier task compared to convincing the ISPs to replace or add these experimental routers to production networks. Comparatively, updating the TCP stack at end-hosts is a relatively easier task. But end-host delay-based schemes have been assailed by doubts about the accuracy and effectiveness of congestion prediction at end-hosts [21, 24, 26].

[*]Currently with Motorola Inc., Austin, TX.

[†]Supported by NSF grants CCR-0306246, ANI-0312461, CNS-0434940, and CNS-0519442.

[1]Aggressive probing mechanisms have been proposed for improving link utilization in high-speed networks, but as long as they use loss-based probing, the discussion here remains valid.

The aim of this paper is not to debate which approach is better. Instead, we focus on evaluating and improving congestion prediction from end-hosts. We show that such congestion prediction is more accurate than characterized by studies in [21, 26]. While it is possible to further improve the prediction ability of end-host based congestion estimators, it may not be possible to entirely eliminate the noise/uncertainty in the signal. We show that this uncertainty can be offset by choosing an appropriate response function. By emulating the probabilistic marking mechanism of router-based AQM schemes in the congestion response function of end-hosts, performance and benefits similar to router-based AQM can be obtained *without actually requiring any modifications to the routers*. We support our claim via extensive simulation on the ns-2 simulator for the emulation of RED [14] and preliminary results for the emulation of PI [16]. We study the stability of the the proposed scheme using control theoretic analysis.

The rest of the paper is organized as follows. Section 2 inspects the existing end-host based congestion prediction signals and some of the studies that cast doubt on their accuracy. We show why some of the claims made by these measurement studies may be inaccurate. We then proceed to determine (a) how to further improve the prediction efficiency of end-host based estimators and (b) based on our observations, what is the appropriate signal to be used in our proposed scheme. In Section 3, we design the congestion response function that emulates RED/ECN behavior. Section 4 presents an extensive evaluation of PERT that emulates RED/ECN using ns-2 simulations over a wide range of network conditions. In Section 5, we present a control-theoretic analysis of the stability of PERT that emulates RED/ECN. Motivated by this analysis, we have started inspecting the effectiveness of emulating PI/ECN at end-hosts and present the analysis and preliminary results of ns-2 simulations in Section 6. Finally, in Section 7, we offer a critical discussion of the issues that are still open and the directions for future work, followed by conclusions in Section 8.

2. END-HOST CONGESTION DETECTION

2.1 Overview of Related Work

In 1989, Raj Jain [17] was the first to propose enhancing TCP with delay-based congestion avoidance. The main observation is that, as the sending rate of a flow increases so does its achieved throughput, until the knee where the link is nearly saturated. During this time, the delay observed by the flow will be low. Beyond the knee point, the throughput stabilizes and the delay increases sharply. The author proposed monitoring the normalized delay gradient of a flow at the sender for determining the knee point and making the flow operate at this point.

Ever since this proposal, several different schemes have appeared. In TRI-S [29], the authors propose using the normalized throughput gradient instead of the normalized delay gradient to foresee when the bottleneck link reaches saturation. In DUAL [30], the authors compare the current sample of the RTT to the average of the minimum and maximum RTT to determine if the bottleneck link queues are more than half full and it is time to respond. In Vegas [7], the authors propose comparing the achieved throughput to the expected throughput based on the minimum observed RTT for predicting congestion. In CIM [21], the authors propose comparing the moving average of a small number of RTT samples to the moving average of a large number of RTT samples for determining if there is any congestion. In TCP-BFA [3], the authors propose monitoring the variance of the RTT for preventing the bottleneck link from filling up. In Sync-TCP [31], the authors use the trend of one-way delays combined with four different levels of window increase/decrease to improve on delay-based congestion avoidance.

Several other studies [6, 21, 24, 26] have focused on understanding whether delay-based congestion prediction can reliably work in the real Internet. Some of the issues raised by these studies are unique to end-host based mechanisms, while others have received similar attention in router based schemes. For instance, it has been pointed out that it is hard to measure queuing delays accurately when they are very small compared to the end-to-end RTT [21, 24, 26]. However, this is a limitation of not just end-host based schemes, but also router-based schemes since a large RTT compared to the queue length implies a large feedback loop. In such cases, the router queue may fill up and result in a loss before the sender receives and responds to this congestion feedback. End-host based measurements of the state of the queue essentially entails sampling the queue at certain times and the fundamental limits of such measurements have been recently highlighted [26, 27]. Problems of oversampling the queue lengths in router based RED mechanisms have been studied in [16]. It has been suggested that on highly aggregated paths, the impact of the response of a single flow may be limited [6, 21]. But if several flows respond, then the combined effect may be sufficient to relieve the congestion and reduce the queue lengths. In this paper, we focus on understanding the issues specific to end-host based schemes.

2.2 Accuracy of End-host Based Congestion Estimation

In [21], the authors measured the correlation between the observed loss rate of a flow and its round trip time samples just before each loss, for data collected using *tcpdump* on seven Internet paths. Their observation is that (a) losses are preceded by a significant increase in RTT in very few cases, and (b) responding to a wrong prediction can result in severe degradation of performance. Based on their observations the authors conclude that although the RTT samples may contain useful information, it cannot be reliably used. Another study [26] shows similar results but over a larger dataset and considers several of the different delay-based prediction metrics.

In order to understand these claims better, consider a very simple scheme that directly monitors the observed round trip delay. Figure 1 shows a simple state diagram of the different states and transitions for such a scheme. State A represents the "low delay" or "low congestion" state. The flow transitions from state A to B when higher delay is observed, or the bottleneck queue is starting to get filled. If no action is taken, the flow transitions to state C or the "packet loss" state. In response to packet loss, when the flow reduces its congestion window, the bottleneck queue starts to empty out and the flow will eventually return to State A.

Next consider the other possible state transitions. If the flow does not use an accurate method for determining the states, or if the flow shares the bottleneck link with a lot of cross traffic, then the duration of state B may be too short

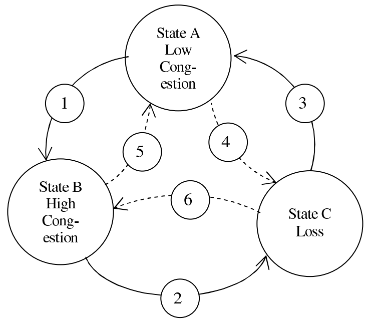

Figure 1: State diagram showing the transition between different congestion states in standard TCP flows.

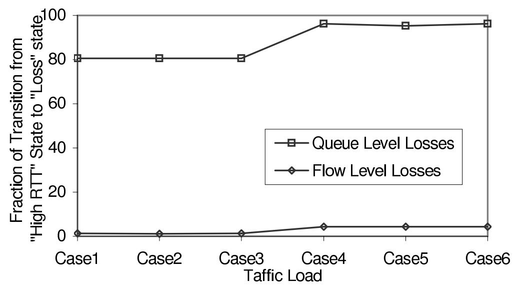

Figure 2: Comparison of the fraction of transitions from "High RTT" to "Loss" when the losses are measured within a flow and at the bottleneck queue.

and the flow may not be able to detect it at all. As a result, before the flow can detect that the delay has increased, it may observe a packet loss. This is the transition from state A directly into state C. Alternately, the transitions are governed by how the protocol defines state A and state B and what criteria is used for the transition from state A to state B. If the protocol is very aggressive, then it may determine the onset of congestion too early and make the transition from state A to state B, only to find out later that it was a false alarm and return from state B to state A. The transition from state C back into state B occurs if the flow does not respond enough after a packet loss.

The transitions marked "4" and "5" are both harmful to a proactive scheme. Transition "4" (state A to C) may mean that short term congestion has occurred and the flow cannot predict it and hence is incapable of avoiding the resulting loss or that the protocol is simply not aggressive enough in predicting congestion. This transition indicates the presence of "false negatives". Transition "5" (state B to A), on the other hand, which we will refer to as "false positive", will mean that the protocol is too aggressive or unreliable in predicting congestion, and as a result may unnecessarily reduce its sending rate and face performance degradation.

The claim made in [21, 26] is that for the congestion predictors used in existing schemes, transition "5" happens more often than transition "2" hence limiting the effectiveness of congestion prediction. The authors arrive at this conclusion by looking at a large dataset of *tcpdump* data of standard TCP flows on the Internet. They then run the algorithms used by different congestion predictors to determine states A and B. The limitation of these studies, however, is the way in which state C is defined. Owing to the methodology used for collecting data, state C (i.e., losses), is observed *within a single flow*. When several different flows share a bottleneck link, the flow under observation may not necessarily be the one to face losses when the bottleneck link is full. Hence observing high RTT that is not followed by a loss (at a single flow) does not necessarily mean that there is no congestion. It just means that the observed flow does not suffer a packet loss (possibly because other flows have responded to losses and reduced the congestion). In order to evaluate the usefulness of a congestion prediction metric, we should consider the correlation between the round trip time and losses *at the bottleneck link*.

In order to illustrate this, we present here the results of a very simple ns-2 simulation. The topology consists of two routers connected by a link of capacity 100 Mbps and delay 20 ms. Several nodes are connected to both routers with links of capacity 500 Mbps and varying delay, resulting in different flows having different RTTs. The capacity of the queue at the routers is set to 750 packets. We use six test case loads denoted as `case1` through `case6` corresponding to the combinations of (a) either 50 or 100 long term flows (in both directions), and (b) 100, 500 or 1000 web sessions. One of the long-term flows is tagged and referred as "observed" flow. For this flow, we collect the RTT samples for every packet over a period of 1000 seconds. The end-to-end delay of this flow is 60 ms. Using a simple threshold of 65 ms to indicate a state of high-RTT, we measure the fraction of transitions from the high-RTT state into the loss state, with losses measured at both flow level and queue level. Figure 2 shows the comparison for the six different cases mentioned above.

While this study is not exhaustive, it clearly indicates that the correlation between the RTT and losses observed at the queue is significantly higher than that observed by a single flow. In other words, delay-based indicators may be a lot more effective in predicting congestion than what is reported in [21, 26].

2.3 Different Congestion Predictors

In order to understand the reliability of congestion predictors used in different end-host based schemes, we applied the algorithms used by these schemes for predicting congestion on the data obtained for the six traffic cases above and evaluated the efficiency of predicting losses (at the bottleneck link queue), and the fraction of false positives and false negatives. We measure the efficiency of loss prediction by the fraction (number of "2" transitions)/("2" transitions + "5" transitions). We measure the fraction of false positives as (number of "5" transitions)/("2" transitions + "5" transitions). Finally, we measure the rate of false negatives as (number of "4" transitions)/("2" transition + "4" transitions). Figure 3 shows the results.

From the figure notice that among the existing schemes for end-host congestion prediction, Vegas has the highest prediction efficiency (which implies the lowest rate of false positives) and the lowest percentage of false negatives. Note that these results are not meant to be an exhaustive evalua-

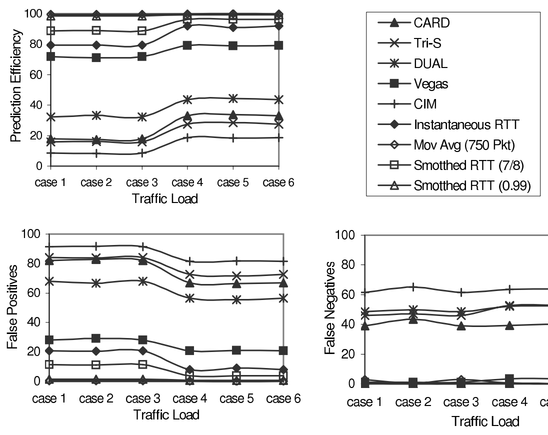

Figure 3: Prediction efficiency, false positives and false negatives for different predictors.

tion of the merits of the different metrics. Rather, it is meant to guide us in our choice for a predictor. From the results, we notice that while the accuracy of congestion prediction may be higher than characterized by earlier measurement studies, it leaves room for further improvement.

2.4 Improving Congestion Prediction

Note that Vegas, CARD, TRI-S, and DUAL obtain RTT samples once per RTT which could result in under-sampling. In order to evaluate the impact of higher sampling, we computed the instantaneous RTT upon the receipt of each acknowledgment[2] and used a simple fixed threshold for determining that the flow is in the high congestion state. Taking RTT samples on each packet addresses some of the concerns raised about end-host measurements [27] and reduces sampling errors. Surprisingly, as shown in the graph, the prediction efficiency of this signal was higher than that of Vegas for the six test cases of the traffic that we have considered.

While the instantaneous RTT signal is more aggressive in predicting losses, it can be quite noisy due to the fluctuations in the instantaneous queuing delays observed by the packets, and hence false positives are quite high. In order to eliminate the noise, we smoothed the RTT signal. Since the aim of the prediction signal is to track changes in the bottleneck queue, we used a moving average of 750 packets (the size of the buffer) for smoothing the signal. This signal was very effective in predicting losses while avoiding false positives and false negatives. However, it is difficult for a flow to estimate the size of buffers at the bottleneck link. So we investigated the effectiveness of using Exponentially Weighted Moving Average (EWMA) of the instantaneous RTT signal, similar to that used by TCP for determining the retransmission timeout, with a weight of 7/8 for the history sample. As seen from the graph, while this reduces the noise (false positives) compared to the instantaneous RTT signal, it still is not as good as the moving average signal. We repeated the EWMA with a higher weight of 0.99 and the signal was able to obtain high prediction efficiency with low false positives and low false negatives.

[2]Current versions of the Linux operating systems (2.4.x and above) do this for RTO calculation anyway [28].

We use the smoothed RTT signal with the weight of 0.99 for the history sample as the congestion predictor in our scheme.[3] In order to differentiate the smoothed RTT signal that we use from that used by TCP for timeout calculations, we refer to our signal as $srtt_{0.99}$. Even though the weights used for smoothing are different, note that this signal achieves similar behavior to that in RED routers where average queue lengths are monitored to infer congestion, while accommodating bursty traffic by allowing fluctuations in the instantaneous queue length.

With the $srtt_{0.99}$ signal, while the false positives are low, they are still non-zero – for the six test cases we considered, the false positives were in the range of $0.7 - 1.5\%$. This number may vary for other traffic cases. The throughput of a TCP flow is proportional to $1/\sqrt{p}$ (p here will be the probability of response). Responding to even a small fraction of false positives may result in severely degraded performance as shown in [21]. In this paper, we take a two-step approach: (a) we improve the accuracy of prediction by using more frequent samples and history information, and (b) we accept that end-host prediction cannot be perfect and devise mechanisms to counter/mitigate this inaccuracy. In the next section, we discuss the design of the response function.

3. RESPONSE TO CONGESTION PREDICTION AT END-HOSTS

One possible mechanism for reducing the impact of false positives would be to keep the amount of response small. In case of a false positive, due to the small amount of response, the flow does not lose much throughput. This is the approach used in Vegas. Vegas uses additive decrease (by one packet) for early congestion response. However, note that this trades off the fairness properties of TCP in favor of maintaining high link utilization, since it has been shown in [8] that Additive Increase/Additive Decrease (AIAD) does not result in fair allocation between competing flows. In the steady-state, if congestion avoidance is successful, then state C is not visited often and hence the flow spends most of its time transitioning between states A and B. Using AIAD for these transitions will result in compromising the fairness properties of the protocol.

Additionally, since the response is small, the buildup of the bottleneck queue may not be cleared out quickly. Hence, compared to a flow starting earlier, a flow that starts later may have a different idea of the minimum RTT on the path (in this case over-estimate it). This variable is used extensively in most end-host based schemes (including Vegas) to estimate what component of the RTT is due to the propagation delay and what component is due to queuing delay. Not clearing out the bottleneck queue completely can hence result in offering an unfair advantage to flows starting later, and result in these flows getting more than their fair share of bandwidth.

An alternate mechanism for reducing the impact of false positives is to respond probabilistically. When the probability of false positives is high, the probability of response to an early congestion signal should be low and vice versa. Due to the probabilistic nature of the response, the protocol can retain multiplicative decrease for early response.

[3]It may be possible to modify the weight for the history sample based on the ack arrival rate. We are investigating this as part of our future work.

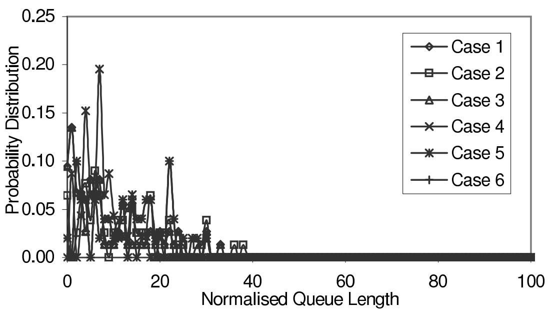

Figure 4: Probability distribution function of normalized queue length when false positives occur.

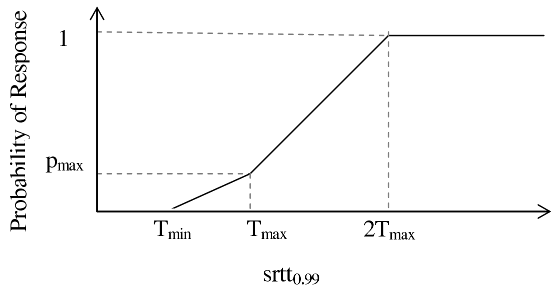

Figure 5: Probabilistic response curve used by PERT.

So, next we study the relationship between the false positives and the queue length at the bottleneck router. Figure 4 shows the probability distribution of the normalized queue length when false positives are detected by the prediction signal $srtt_{0.99}$ for the six cases of traffic loads discussed earlier. From the figure, notice that false positives are more likely to occur when the queue length is smaller. Note that the traffic load for the six different cases consists of a mix of long term flows in both forward and reverse directions with different RTTs and also web traffic. For these cases of traffic load, the false positives occur mostly when the queue length is less than 50% of the total queue size. While we may not be able to generalize the results for all possible types of traffic mix, it provides us an important insight: the uncertainties in congestion prediction are more likely to occur at lower queue lengths than at higher queue lengths, at least when $srtt_{0.99}$ is the congestion predictor.

Since the aim is to reduce the impact of false positives by designing an appropriate response function, we argue that when the queue size is small the response of a proactive scheme should be small and when the queue size is large, the response should be large. Note that this is conceptually similar to the probabilistic response function used in the AQM mechanism RED. Hence, for the response function, we emulate the probabilistic response function of RED. Due to the probabilistic nature of early response we call our scheme *Probabilistic Early Response TCP* (PERT).

The probabilistic response of PERT is designed to be similar to that of "gentle" RED. Figure 5 shows the probability of response against the congestion detection signal $srtt_{0.99}$. Similar to RED, we define two thresholds T_{min} and T_{max} and the maximum probability of response p_{max}. When the value of $srtt_{0.99}$ is below T_{min}, the probability of response is 0. As the value of $srtt_{0.99}$ increases beyond T_{min}, the probability of reducing the window in response to each ACK linearly increases until it reaches the value p_{max} at T_{max}. Between T_{max} and $2T_{max}$, the probability increases from p_{max} to 1. Beyond $2T_{max}$, the probability remains constant at 1.[4] For the parameters T_{min}, T_{max} and p_{max} we use fixed values of ($P+5$ ms, $P+10$ ms and 0.05) respectively, where P is the propagation delay estimated by the minimum RTT observed by the flow. It is possible to choose these values adaptively based on network conditions similar to the mechanisms suggested in [12] – we are looking into this as part of our future work.

[4]Different response functions can be chosen. "Gentle" RED is used here as a representative function.

When the RTT is measured on every ACK, potentially each ACK could indicate congestion. Since the impact of response may not be seen until after an RTT, we limit the early response to once per RTT.

For the early response, we use multiplicative decrease. When the proactive congestion response is successful, the queue lengths are expected to be maintained low. As a result, it is not necessary to respond with a 50% window reduction in case of early response. In [1], the authors suggest that router buffers are commonly set to the Bandwidth-Delay Product of the link since the TCP flow reduces its window by 50%. If TCP flows were to use a factor f reduction during loss, then the relationship between the buffer size B and the window reduction factor f can be re-written as

$$B > \frac{f}{1-f} \times BDP, \qquad (1)$$

where BDP is the Bandwidth-Delay Product of the link. Based on this, for a conservative value that the queue length does not exceed half of its capacity and that the capacity of the buffer is set to one BDP according to the rule of thumb, we choose the window decrease factor to be 35%. If a packet loss occurs, then the response will be similar to that of standard TCP variants and the fast retransmit/recovery algorithms are triggered.

Note that the choice of the amount of response is a trade off. Since the flows respond before the bottleneck queue is full, a large multiplicative decrease can result in lower link utilization. On the other hand, decreasing the amount of response could result in the bottleneck link buffers not getting entirely cleared, leading to unfairness among flows starting at different times, as discussed earlier.

While it has been a natural choice based on our observations to emulate the probabilistic marking of RED, it is possible to replace the probabilistic response curve with the algorithms used in other AQM schemes as well. In the next section, we present an extensive evaluation of PERT that emulates RED/ECN using ns-2 simulations. This is followed by a control theoretic analysis of the stability of PERT emulating RED. Based on our analysis we are motivated to evaluate the emulation of PI/ECN in PERT, for which we provide analysis and preliminary simulation results. As part of our future work, we will continue to investigate the possibility of emulating other AQM mechanisms as well.

4. EXPERIMENTAL EVALUATION

We have conducted extensive ns-2 simulations to evaluate PERT. We attempt to make our evaluation realistic by simulating a wide range of network parameters. We first present the results for a single bottleneck topology with bottleneck link bandwidth in the range of [1 Mbps, 1 Gbps], RTT in the range of [10 ms, 1 s], the number of long-term background flows in the range of [1, 1000] and the number of web sessions in the range of [10, 1000]. We then evaluate the impact of multiple bottleneck links and flows of different RTTs. All simulations are run for 400 seconds and reported results are measured during the stable period between 100 and 300 seconds to show the steady state behavior. Next, we evaluate the dynamic behavior due to transient changes in traffic load where sudden changes in available bandwidth are caused by the arrival or departure of flows. For all experiments, the bottleneck buffer size is set to the bandwidth-delay product, with the minimum number of packets being equal to at least twice the number of flows. When multiple flows share a link, their start times are chosen randomly in the range (0, 50) seconds to illustrate the impact of flows starting at different times on the fairness as discussed in the previous section. We present all results in comparison to (a) SACK with Droptail queues; (b) ECN-enabled SACK with RED queues; and (c) TCP-Vegas. For experiments with TCP-Vegas and PERT, the bottleneck link routers use the default Droptail buffer management.

4.1 Impact of Bottleneck Link Bandwidth

In this experiment, the bottleneck link bandwidth is varied from 1 Mbps to 1 Gps. The end-to-end RTT of the flows is set to 60 ms and the number of flows is varied such that the link is efficiently utilized even at large bandwidth. Figure 6 shows the average bottleneck link queue length, the bottleneck link drop rate, the link utilization and the Jain fairness index [8] of the competing flows. From the graph, notice that PERT's average queue length is similar to (and in some cases better than) that of SACK/RED-ECN. As expected, the average queue length with SACK/Droptail remains high for most experiments. A surprising result is that TCP-Vegas has a higher average queue length than SACK/Droptail in some cases. Higher drop rates observed with SACK/Droptail relieve the congestion while Vegas' maintenance of 1-3 packets in the buffer (based on the α and β parameters) could lead to high queue lengths, while avoiding packet losses. All the proactive mechanisms (SACK/RED-ECN, PERT and TCP-Vegas) maintain zero losses in most cases. The link utilization of PERT is lower than SACK in the cases where the bottleneck link bandwidth is small resulting in short buffers, but in the other cases, it is similar. TCP-Vegas maintains a high link utilization, which comes at the cost of fairness among competing flows. The fairness among PERT flows is similar to that of standard TCP, with the Jain fairness index being close to 1.

4.2 Impact of Round Trip Delays

In this experiment, the bottleneck link bandwidth is 150 Mbps and the number of flows is 50. The end-to-end delay is varied in the range of 10 ms to 1 second. Figure 7 shows the results. From the figure, we see that the average bottleneck link queue length and the drop rate are similar for both PERT and SACK/RED-ECN. The latter has a lower link utilization, which is slightly better than in PERT since

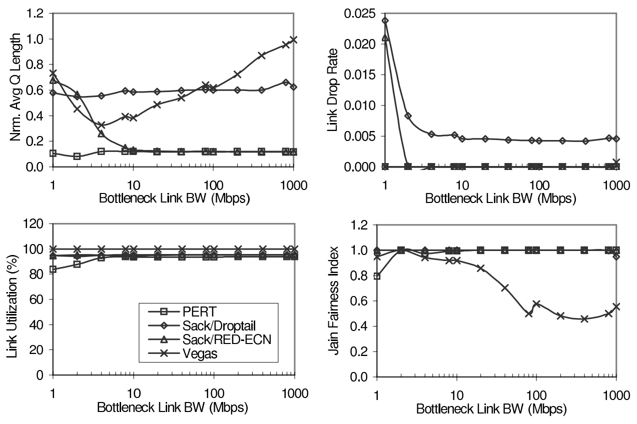

Figure 6: Impact of bottleneck link bandwidth. Note the logarithmic scale of the x-axis.

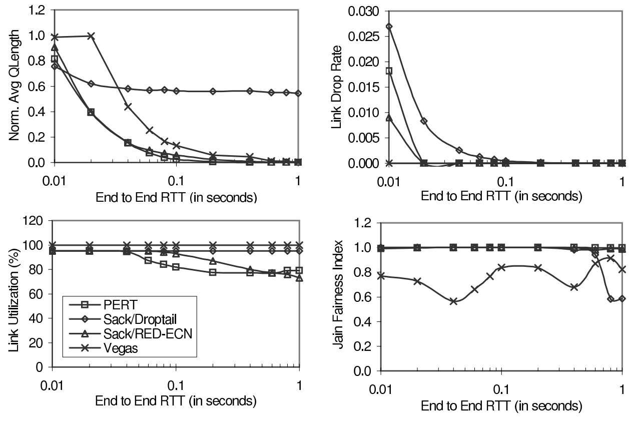

Figure 7: Impact of end-to-end RTT. Note the logarithmic scale of the x-axis.

we have used the adaptive RED version for the routers that tunes the parameters according to network conditions and PERT uses fixed thresholds. The Jain fairness index remains high indicating that the throughput is shared in a fair manner among the 50 flows.

4.3 Impact of Varying the Number of Long-term Flows

In this experiment, the bottleneck link bandwidth is set to 500 Mbps and the number of long-term flows is varied from 1 to 1000. The end-to-end delay is 60 ms. Figure 8 shows the results. Again the average bottleneck link queue length and the bottleneck link drop rate of PERT are similar to that of SACK/RED-ECN. The Jain Index remains high even when the number of flows is large. Link utilization of SACK/RED-ECN is slightly higher than that of PERT. Vegas tries to maintain a fixed number of packets in the queue and as a result, as the number of flows increases, so does the average queue length and the drop rates. The link utilization of Vegas remains high, while the Jain fairness index remains low.

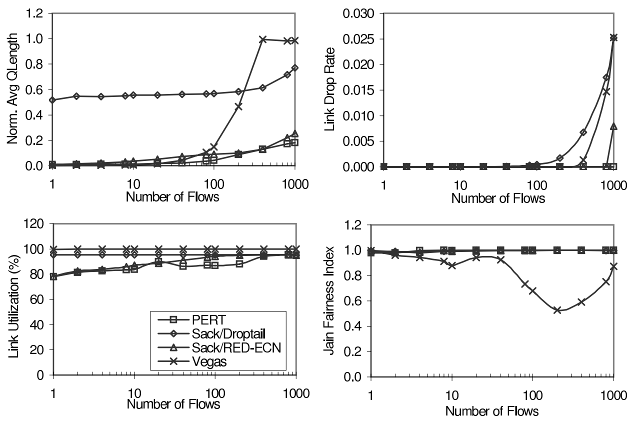

Figure 8: Impact of varying the number of long-term flows. Note the logarithmic scale of the x-axis.

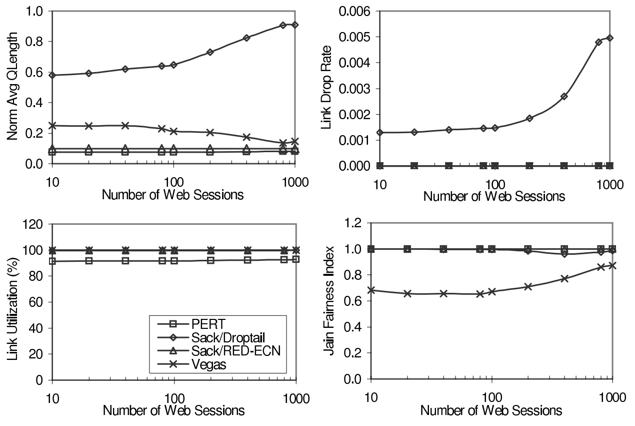

Figure 9: Impact of varying the number of web sessions. Note the logarithmic scale of the x-axis.

4.4 Impact of Web Traffic

We now consider simulations with web traffic to understand the impact of bursty flows. The bottleneck link bandwidth is set to 150 Mbps, the end-to-end delay is 60 ms and the number of long term flows is 50. The number of web sessions that share the bottleneck link is increased from 10 to 1000. For the web traffic, the parameters are chosen based on the guidelines in [11]. As seen from Figure 9, as the load offered by the web traffic increases, the average link queue length remains low and as a result no packet losses are observed for PERT, similar to SACK/RED-ECN. The link utilization of PERT is slightly lower than that of SACK/RED-ECN. The Jain fairness index of the long-term flows remains high.

4.5 Impact of Different RTTs

In the following experiment, a bottleneck link with 150 Mbps capacity is shared by 10 flows with end-to-end delays being $(12, 24, 36, \ldots, 120)$ ms. We run 100 web sessions in the background. Table 1 shows the normalized average queue length (Q), bottleneck link droprate (p), bottleneck

	Q	p	U	F
PERT	0.28	3.98E-06	93.81	0.86
Sack/Droptail	0.42	7.18E-04	93.77	0.44
Sack/RED-ECN	0.41	4.95E-04	93.90	0.51
Vegas	0.07	0	99.99	0.98

Table 1: Normalized average queue length (Q), bottleneck link droprate (p), bottleneck link utilization (U) and Jain fairness index (F) when flows with different RTTs share the bottleneck link.

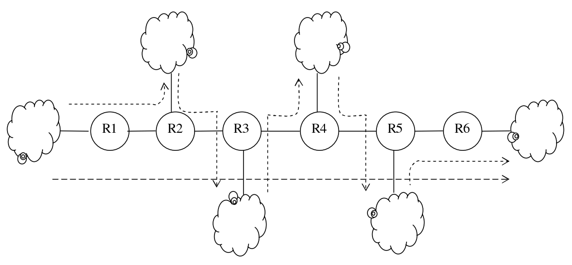

Figure 10: Topology used for understanding the impact of multiple bottleneck links.

link utilization (U) and Jain fairness index (F) for the different schemes. From the table we see that PERT and Vegas may reduce the RTT unfairness inherent to TCP. The link utilization of PERT is similar to that of SACK with either Droptail or RED/ECN, while the average queue length and drop rate are lower.

4.6 Multiple Bottlenecks

The congestion signal in PERT measures the end-to-end RTT, and hence conveys information about the combined queue lengths along the path. Router based AQM schemes, on the other hand, estimate the queue length locally, then use ECN to convey the information end-to-end. In this experiment we investigate the impact of multiple bottleneck links. The topology, shown in Figure 10, consists of six routers labeled R1 to R6. The links between routers have a capacity of 150 Mbps and a delay of 5 ms. Each router is connected to a cloud of 20 nodes with a link of capacity 1 Gbps and delay 5 ms. The nodes in each cloud send data to the nodes in the cloud connected to the adjacent router. Also, all the nodes in the cloud connected to router R1 also send data to the nodes in the cloud connected to R6.

Figure 11 shows the average queue length, drop rate and utilization of the link between each pair of routers as well as the Jain fairness index of all flows between each pair of routers. From the figure we see that PERT maintains low queue length and zero drop rates across all bottleneck link queues. Its link utilization is similar to that of SACK/RED-ECN and the fairness among flows passing through the common set of routers is maintained.

4.7 Dynamic Protocol Behavior

In this experiment, we study the dynamic protocol behavior of the different schemes. Unlike previous experiments which focused on the steady-state behavior, the results of the first few seconds of simulation time are not discarded to illustrate the impact of transient dynamics. These experi-

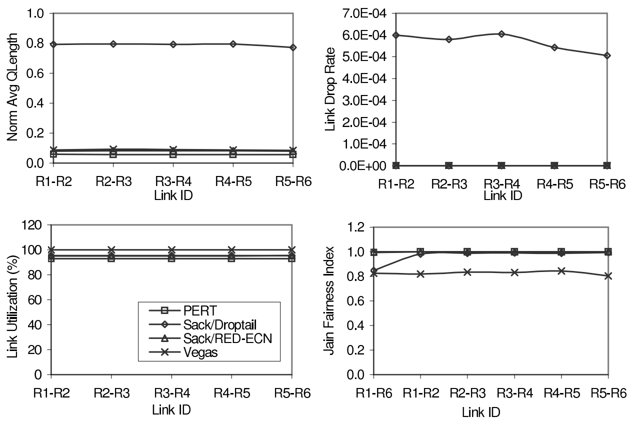

Figure 11: Impact of multiple bottleneck links.

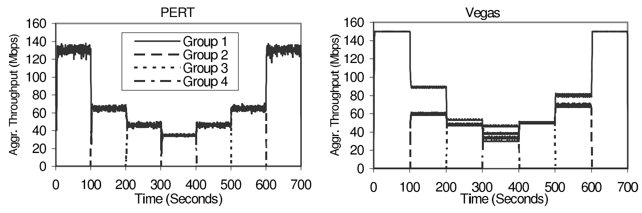

Figure 12: Response to sudden changes in responsive traffic.

ments investigate the responsiveness of the different schemes to sudden changes in traffic.

In this experiment, 25 PERT flows are started at time 0 seconds. Starting at 100 seconds, for the next 300 seconds 25 new flows are added at 100 second intervals, causing severe contention for available bandwidth. Starting at 400 seconds, 25 flows leave the network at 100 second intervals creating a sudden availability of bandwidth. We repeat the experiment with SACK/Droptail, SACK/RED-ECN and Vegas. Figure 12 shows the aggregate throughput of the set of flows that start together. From the figure it is clear that the PERT flows respond quickly to dynamic changes in network bandwidth. Vegas exhibits previously observed unfairness among competing flows. The results for SACK/Droptail and SACK/RED-ECN are similar to that of PERT and are available in [4].

We have conducted additional experiments, where dynamic changes in traffic were caused by non-responsive traffic. The results are similar to those above. We have not included the results here due to the lack of space. They are available in [4].

5. MODELING AND STABILITY OF PERT

5.1 Model

Our modeling of PERT is composed of three parts: window adjustment, RED emulation, and queuing behavior. We start with the window dynamics. Similar to [23], we consider a single-link scenario and assume the forward propagation delay from the source to the router is negligible and thus the round-trip time $R(t)$ measured by the end-user at time t is composed of backward propagation delay T_p and queuing delay $T_q(t - R(t))$, i.e.,

$$R(t) = T_p + T_q\big(t - R(t)\big). \tag{2}$$

Denoting by C the link's capacity and by $q(t)$ the queue size at time t, queuing delay $T_q(t)$ can be approximated by $q(t - R(t))/C$. Note that delay $R(t)$ in the last expression is because the queuing delay perceived by the user at time t is actually experienced by the router $R(t)$ time units earlier. To compare the stability of PERT to router-based RED with standard TCP, we set window decrease factor β to 0.5 and note that results for $\beta = 0.35$ can be similarly obtained following the procedure below. Then, window dynamics of a PERT end-flow is written as:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)W\big(t - R(t)\big)}{2R\big(t - R(t)\big)}p(t), \tag{3}$$

where, at time t, $W(t)$ is the congestion window size, $R(t)$ is the RTT, and $p(t)$ is the packet dropping probability. Note that loss rate $p(t)$ in the last equation is an instantaneous value as opposed to its delayed counterpart $p(t-R(t))$ in the TCP/RED model obtained in [23]. This is because a PERT user makes its dropping decision at the end-host instead of the router.

To formulate PERT's emulation of the RED mechanism, assume that the propagation delay T_p is known to the end-flow (this can be approximated by the base RTT). Then, upon each packet arrival, the user can estimate the queuing delay by $T_q(t) = R(t) - T_p$ and generate the packet drop probability $p(t)$ as following:

$$p(t) = \frac{T_q(t) - T_{min}}{T_{max} - T_{min}} p_{max}, \tag{4}$$

where T_{min} and T_{max} are the maximum and minimum thresholds of queuing delays and p_{max} is a constant.

Another component of RED emulation is the estimation of round-trip time $R(t)$, which is updated per-packet using a low-pass filter (LPF) with weight α, i.e.,

$$R(t) = \alpha R(t-1) + (1-\alpha)\hat{R}(t), \tag{5}$$

where $\hat{R}(t)$ is the instantaneous RTT measured at time t and weight $\alpha = 0.99$. Following the technique used in [23], this LPF can be approximated by the following differential equation:

$$\dot{R}(t) = \frac{\ln\alpha}{\delta}(R(t) - \hat{R}(t)), \tag{6}$$

where δ is the sampling interval.

We next model the queuing dynamics, which can be described by the following differential equation of queue size:

$$\dot{q}(t) = \frac{W(t)}{R(t)}N(t) - C,$$

where $N(t)$ is the number of flows accessing the router at time t and term $W(t)N(t)/R(t)$ can be interpreted as the combined incoming rate $y(t)$. Since $T_q(t) = q(t - R(t))/C$, we re-write the last equation in terms of queuing delay $T_q(t)$:

$$\dot{T}_q(t) = \frac{W(t - R(t))N(t - R(t))}{R(t - R(t))C} - 1. \tag{7}$$

Then, equations (2)-(7) comprise the complete system model of PERT. It is easy to see that this model is very similar to the TCP/RED model derived in [23]. Note that, there are also some differences between the system models of PERT and TCP/RED. The first one is that, in PERT, loss rate p in (3) is not delayed by $R(t)$ and variables for computing queuing delay T_q in (7) are retarded by one RTT. As we have mentioned earlier, this difference is because PERT monitors queuing dynamics at the end-user, while RED does this inside the router. The second difference is that PERT uses queuing delay $T_q(t)$ to determine the loss probability, while RED uses queue length $q(t)$. We next study stability of PERT (2)-(7).

5.2 Stability

Similar to [23], we assume that the number of flows N and RTT R are constant in the steady state. Then, the system becomes:

$$\begin{aligned} f(W, W_R, T_q, p) &= \frac{1}{R} - \frac{W(t)W_R(t)}{2R}p(t), \\ g(W, T_q) &= \frac{N}{RC}W(t) - 1, \end{aligned} \tag{8}$$

where $W_R(t) = W(t-R)$. In the steady state, we have $W(t) = W_R(t) = W^*$, $p(t) = p^*$, and $T_q(t) = T_q^*$. Applying this result and equating the left-hand sides of (8) to zero, we can derive the equilibrium point (W^*, p^*, T_q^*) as follows:

$$W^* = \frac{RC}{N} \text{ and } p^* = \frac{2N^2}{R^2C^2}. \tag{9}$$

The following theorem states a sufficient condition for PERT (8) to be locally stable in its stationary point. Due to limited space, we omitted proofs of all theorems presented in this section and refer interested readers to [5] for more detail.

Theorem 1. *Let L_{PERT} and K be defined as follows:*

$$L_{PERT} = \frac{p_{max}}{T_{max} - T_{min}}, \quad K = \frac{\ln\alpha}{\delta}, \tag{10}$$

and assume bounds R^+ and N^- satisfy the following condition:

$$\frac{L_{PERT}R^{+3}C^2}{(2N^-)^2} \le \sqrt{\frac{w_g^2}{K^2} + 1}, \tag{11}$$

where:

$$w_g = 0.1 \min\Big(\frac{2N^-}{R^{+2}C}, \frac{1}{R^+}\Big). \tag{12}$$

Then, PERT modeled by (8) is locally stable for all $N \ge N^-$ and $R^* \le R^+$, where $R^* = T_p + T_q^*$ is the stationary RTT.

Note that, in addition to conditions given in Theorem 1, the equilibrium condition $p^* \le p_{max}$ is also necessary for the system to be stable. To examine validity of this condition, we infer from (9) that $p^* = 2/(W^*)^2$. Thus, condition $p^* \le p_{max}$ holds if and only if $p_{max} \ge 2/(W^*)^2$. For a stationary window size $W^* = 10$ packets, we have $p_{max} = 2\%$, which is reasonable in practice.

We next investigate whether there exists a closed-form solution of condition (11)-(12). Recalling $K = \ln\alpha/\delta$, we convert (11) into the following condition on δ:

$$\delta \ge -\frac{\ln\alpha}{4(N^-)^2 w_g}\sqrt{L_{PERT}^2 R^{+6}C^4 - 16(N^-)^4}. \tag{13}$$

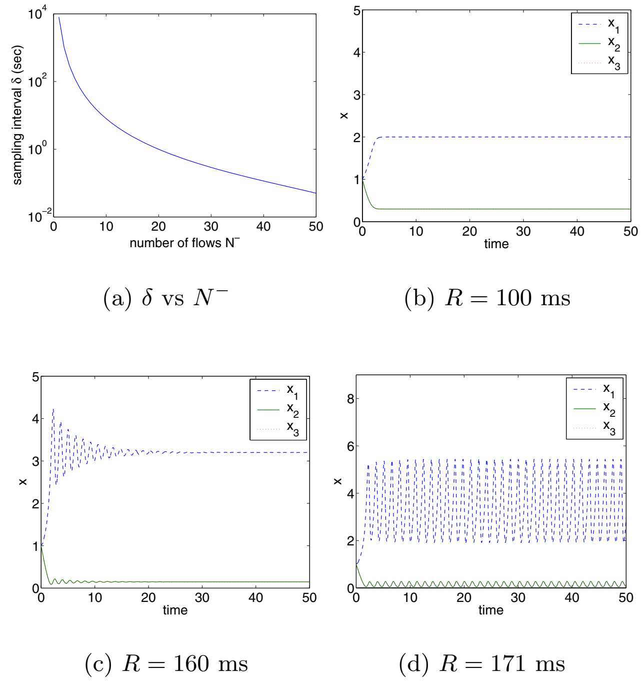

Figure 13: (a) Sampling interval δ as a function of the minimum number of flows N^-; (b)-(d) the fluid model of PERT (2)-(7) under different delay R.

This equation can be used as the guideline for choosing sampling interval δ given C, R^+, and N^-.

Moreover, we observe that given R^+ and C, δ scales inversely proportional to N. To better see this, consider the following simulation, where RTT $R = 200$ ms, packet size $s = 1250$ bytes, $p_{max} = 0.1$, $T_{max} = 100$ ms, $T_{min} = 50$ ms, and $\alpha = 0.99$. We set $C = 10$ Mbps (which corresponds to 1000 pkts/s) and range N^- from 1 to 50 to see the minimum requirement of δ. As seen from Figure 13(a), the minimum δ monotonically decreases and reaches 0.1 seconds as N^- goes to 40. We note that N^- is the lower bound of N. Thus, once a stable δ is picked given certain N^-, R^+, and C, stability of the system is not affected by N or R as long as $N \ge N^-$ and $R \le R^+$.

5.3 Simulations

We next examine Theorem 1 using Matlab simulations. Bringing in notations $R(t) = R$, $x_1(t) = W(t)$, $x_2(t) = \hat{R}_q(t)$, and $x_3(t) = R_q(t)$, the fluid model of PERT (2)-(7) can be transformed into the following delay differential equations (DDEs):

$$\begin{aligned} \dot{x}_1(t) &= \frac{1}{R} - \frac{L_{PERT}x_1(t)x_1(t-R)(x_3(t-R) - T_{min})}{2R}, \\ \dot{x}_2(t) &= \frac{N}{RC}x_1(t) - 1, \\ \dot{x}_3(t) &= Kx_3(t) - Kx_2(t), \end{aligned} \tag{14}$$

where K and L_{PERT} are defined in (10).

Set link capacity $C = 100$ pkt/s (or 1 Mbps with packet size 1250 bytes), $\delta = 0.1$ ms, $N = N^- = 5$, $p_{max} = 0.1$, $T_{max} = 100$ ms, $T_{min} = 50$ ms, and $\alpha = 0.99$. We keep

$R = R^+$ and test stability of (14) under different values of delay R. For all simulations, we set the initial point to be $(1, 1, 1)$ and the unit of $x_1(t)$ is packets and that of $x_2(t)$ and $x_3(t)$ are both seconds. Start with $R = 100$ ms, which satisfies the stability condition in Theorem 1. As illustrated in Figure 13(b), the system is stable with monotonic trajectories. We then increase R to 160 ms, which is closer to the stability boundary, but still satisfies the stability condition. As shown in Figure 13(c), the system is stable and converges to its equilibrium (9) after decaying oscillations. Finally, we increase R to 171 ms, which is exactly on the stability boundary. As seen from Figure 13(d), the system is unstable and exhibits persistent oscillations, whose amplitude increases with the value of R. Note that the stability boundary derived from Theorem 1 is not exact. Even when R is within the stability region but close enough to the boundary, the system becomes unstable.

This discrepancy is due to the approximations $W(t) = W(t - R(t))$ in the proof. To justify this reasoning, we rewrite (14) using $W(t) = W(t - R(t))$ and repeat the above simulations. The simulation results demonstrate that stability of the system is guaranteed under condition (11) and becomes unstable when R is increased to 175 ms (which exceeds the stability boundary 171 ms). Lack of necessity of the stability condition is also pointed out in the context of TCP/RED in [15]. However, Theorem 1 still serves as a general guideline for choosing PERT parameters.

5.4 Discussion

We remark on two differences between Theorem 1 and [23, Proposition 1]. The first one is that in the left-hand side of (11), we have C^2 instead of C^3 as in [23, (8)]. This is because PERT uses queuing delay $T_q(t)$ to determine loss probability, while RED uses queue size $q(t)$. Applying $L_{PERT} = L_{RED}C$ in (11), it is evident that the resulting stability condition is identical to that of RED.

The second difference is that sampling interval δ in RED is approximately fixed to $1/C$, while in PERT δ of user i is the inter-packet arrival time $1/x_i(t)$, which is approximately N/C. As a consequence, for fixed C, the more flows are accessing the bottleneck link, the slower the sampling action of each flow. Reflected in the stability condition, this also results in less constraint on C, R^+, and N^- than in RED. Thus, both differences actually increase the stability region of PERT. On the other hand, when the frequency of sampling action becomes extremely low, RTT estimation at the end user may not accurately capture the queuing dynamics inside the router. However, we argue that this problem does not occur in practice, since the ISPs will increase link capacity as the number of accessing flows becomes large to decrease the packet loss rate and prevent end-flows from starving. Thus, it can be expected that Internet flows would normally have a sufficient number of packets within one RTT of each flow such that the end-user has enough samples for RTT estimation.

In addition, it is common to assume that link capacity C scales linearly with the number of flows N [10], i.e., $C/N = \sigma$, where σ is constant. Then, assuming $W^* \geq 2$, $N = N^-$, and $R = R^+$, condition (11)-(12) translates into:

$$L_{PERT}\sigma^2 R^+ \leq 4\sqrt{\frac{0.04}{\sigma^2 K^2 R^{+4}} + 1}, \tag{15}$$

which is independent of C and N^- and is only a function of R^+. This property does not hold for RED due to term C^3 instead of C^2 in (11). As a consequence, sampling interval δ of RED depends on (and is thus unscalable to) C even if C/N is kept constant.

We should also note that, similar to RED, sampling interval δ in PERT also depends on the link capacity C and packet size s. Specifically, δ becomes small when C increases or s decreases and becomes large otherwise. As pointed out in [23], this adaptive nature of δ is harmful since small δ results in close tracking of the instantaneous queue length and thus leads to large oscillations. On the other hand, large δ results in increased rise time and extended initial overshoot of the queue length. Thus, it is suggested that δ be set to a fixed value that is independent of C, s, and N. As mentioned above, this requirement can be easily satisfied when ratio C/N is kept constant. A detailed discussion of the impact of δ on system performance is available in [23, Section 3.4].

6. EMULATING PI

It has been identified that RED has several drawbacks, such as a lack of significant performance improvement for pure web traffic [9] and mixtures of FTP, UDP and HTTP traffic [22], difficulties in tuning RED parameters [9, 13, 23], and the tradeoff between stability and responsiveness of the system [23]. As pointed out in [16], the major cause of the last drawback in the above list is the averaging mechanism of the LPF. Thus, the authors suggest using the instantaneous queue length and applying a PI controller to it. Simulation results in [16] demonstrate that PI offers better performance than RED in terms of stability and stationary queue size. Motivated by this fact, we next seek to emulate PI inside PERT.

The transfer function of PI is:

$$C_{PI}(s) = \frac{\Delta P(s)}{\Delta T_q(s)} = K\frac{1 + s/m}{s}, \tag{16}$$

where $\Delta P(s)$ and $\Delta T_q(s)$ are respectively the Laplace transforms of $\delta p(t) = p(t) - p^*$ and $\delta T_q(t) = T_q(t) - T_q^*$ and K and m are constants to be determined next. Assuming $p^* = 0$, we re-write PI (16) in the time-domain as follows:

$$\delta p(t) = p(t) = K\Big(\delta T_q(t) + \frac{1}{m}\int \delta T_q(t)dt\Big). \tag{17}$$

To implement the continuous PI (17) in PERT, we discretize it using the bilinear transform $s = 2(z-1)/\delta(z+1)$, where δ is the sampling interval, and convert the s-domain transfer function given in (16) into the following counterpart in the z-domain:

$$C_{PI}(z) = \frac{\Delta P(z)}{\Delta T_q(z)} = \frac{\gamma z - \beta}{z - 1}, \tag{18}$$

where $\gamma = K/m + K\delta/2$ and $\beta = K/m - K\delta/2$. Then, the time-domain version of the PI controller becomes:

$$p(t) = \beta(T_q(t) - T_q^*) - \gamma(T_q(t-1) - T_q^*) + p(t-1), \tag{19}$$

Thus, in the new version of PERT, the only change is to replace RED emulation (4)-(5) with the PI controller (19).

Stability condition of PERT/PI defined by (8) and (19) is given below.

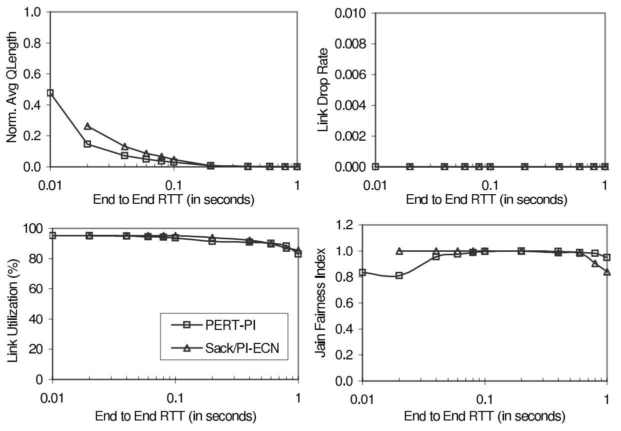

Figure 14: Emulating PI at end-hosts. Note the logarithmic scale of the x-axis.

THEOREM 2. *Assume:*

$$W^* \gg 2, \tag{20}$$

where W^* is the stationary window size. Then, PERT/PI defined by (8) *and* (19) *with*

$$m = \frac{2N^-}{R^{+2}C} \quad and \quad K = m\left|\frac{jR^*m+1}{\frac{R^{+3}C^2}{(2N^-)^2}}\right|, \tag{21}$$

is locally stable for all flow numbers $N \geq N^-$ and all stationary RTT $R^* \leq R^+$.

Similar to Theorem 1, since PERT uses queuing delay to determine the loss probability, in parameter K (21), we have term C^2 instead of C^3 as in the original stability condition of TCP/PI [16, Proposition 2]. In addition, it is clear that this controller is easy to implement and does not require a complicated tuning process of parameters such as T_{min}, T_{max}, p_{max} and α as in RED.

6.1 Preliminary Results

In this section, we present preliminary results of ns-2 simulations for the emulation of PI [16] in PERT. Similar to the emulation of RED in PERT, we use the $srtt_{0.99}$ signal for measuring the delay. However, the PI algorithm is used for predicting congestion. The parameters for PERT-PI are chosen by multiplying the parameters used in router PI by the link capacity. The target delay is set at 3 ms. The experiment is similar to that in Section 4.2. Figure 14 shows the results.

As seen from the figure, the link utilization and average queue length of PERT-PI is similar to router-based PI with ECN support. Also, the PERT-PI emulation from end hosts is very effective in avoiding packet drops. The Jain fairness index at low RTTs is slightly worse, but at high RTTs, the Jain fairness index is slightly better compared to the router-based implementation. Note that these results are preliminary and have been included here to illustrate that PERT is a general scheme and can be used for emulating different AQM mechanisms. We are currently conducting more exhaustive simulations to evaluate PERT/PI in more diverse network conditions and examining the emulation of other AQM schemes from end-hosts.

7. DISCUSSION

While the results presented in Section 4 and 6 are highly encouraging, we have identified some open issues, which require further study. Here, we discuss the open issues and some of the possible solutions for addressing them.

Impact of Reverse Traffic: Similar to earlier schemes using delay-based congestion avoidance, we use the round trip time to predict the build-up of bottleneck queues. However, since round trip time is the sum of delays in both forward and reverse direction, congestion in the reverse path can trigger an early response. Whether congestion response should be only for forward path congestion, like the current versions of TCP, or for bidirectional congestion is debatable since reverse path congestion can cause loss of acks possibly resulting in timeouts. In this paper, we do not address this issue. It must be noted, however, that if responding to reverse path congestion is not acceptable, then PERT can be used with one-way delays to achieve similar benefits. Methods for computing one-way delays have been studied in [20, 31].

Co-existence with Non-Proactive Flows: Another issue with end-host solutions is that of co-existence with non-proactive flows. This problem can be addressed relatively easily in router-based schemes, by dropping the packets instead of marking them when the queue length exceeds a certain threshold forcing the non-proactive schemes to also back off before the bottleneck link queue fills up. With an end-host based scheme, each flow only has control over its own congestion window and hence emulating something similar is not possible.

We are currently investigating mechanisms for making the pro-activeness adaptive based on the flow's perception of how effective early response has been. A number of possibilities exist: increasing the time for next response progressively if queue lengths persist, making the probability of response a function of the time since the last response, limiting the probabilistic early response to once when the probability exceeds some threshold (say 0.75) etc. Alternately, the increase function can be made more aggressive than that in TCP in the absence of congestion to compensate for the loss in throughput in the presence of congestion. We will inspect these options as part of our continued work in this area.

Other Key Differences between Router-Based AQM and End-host Emulation: In the case of router-based AQM, end-hosts receive congestion information only when the marking algorithm used by the AQM scheme marks the packets of its flow. With end-host AQM, potentially all the end-hosts will be able to identify the onset of congestion at the same time. This can lead to more options in designing the response function.

Even though the long term average probability of response is similar for the two approaches, distributions of probabilities at individual flows could be different. These differences may lead to a slightly different choice of parameters at end-hosts than at routers.

Some of the algorithms used in AQM schemes need an estimation of the round trip time of the flow to determine the feedback [18, 32] to the end-host. The routers use an average value for this purpose. With an end-host based emulation, the flow knows its RTT, providing options for different new designs.

End-host based emulation uses end-to-end delay as a congestion measure and hence can ensure that delays have a tighter bound end-to-end, compared to the router-based AQM where each router can offer bounds only locally.

8. CONCLUSIONS

In this paper, we showed that congestion prediction at end hosts is more accurate than characterized by previous studies based on flow-level measurements. We showed that while it was necessary to further improve the accuracy of end-host delay-based congestion predictors, the impact of any inaccuracies could be mitigated by the choice of an appropriate response function. We have presented here a scheme called PERT, which emulates the behavior of AQM in the congestion response function. PERT is shown to offer benefits similar to using router-based schemes with ECN marking, but without requiring router support. Our results, based on a wide array of network conditions, indicate that PERT can efficiently maintain low queue lengths and almost zero losses, while retaining a high degree of fairness. PERT's link utilization is similar to that of router-based schemes. The proposed scheme is flexible in the sense that other AQM schemes can be potentially emulated at the end-host.

9. REFERENCES

[1] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *Proc. ACM SIGCOMM*, pages 281–292, August/September 2004.

[2] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: Active queue management. *IEEE Network*, 15(3):48–53, May/June 2001.

[3] A. A. Awadallah and C. Rai. TCP-BFA: Buffer fill avoidance. In *Proc. IFIP High Performance Networking Conference*, pages 575–594, September 1998.

[4] S. Bhandarkar. *Congestion Control Algorithms of TCP in Emerging Networks*. PhD thesis, Texas A&M University, August 2006.

[5] S. Bhandarkar, A. L. N. Reddy, Y. Zhang, and D. Loguinov. Emulating AQM from end hosts. Technical Report TAMU-ECE-2007-03, Texas A&M University, June 2007.

[6] S. Biaz and N. Vaidya. Is the round-trip time correlated with the number of packets in flight? In *Proc. USENIX/ACM IMC*, pages 273–278, October 2003.

[7] L. Brakmo, S. O'Malley, and L. Peterson. TCP vegas: New techniques for congestion detection and avoidance. In *Proc. ACM SIGCOMM*, pages 24–35, August 1994.

[8] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, June 1989.

[9] M. Christiansen, K. Jeffay, D. Ott, and F. Smith. Tuning RED for web traffic. In *Proc. ACM SIGCOMM*, pages 139–150, August 2000.

[10] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer sizing for congested Internet links. In *Proc. IEEE INFOCOM*, pages 1072–1083, March 2005.

[11] 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 *Proc. ACM SIGCOMM*, pages 301–313, September 1999.

[12] W. Feng, D. Kandlur, D. Saha, and K. Shin. A self-configuring RED gateway. In *Proc. IEEE INFOCOM*, pages 1320–1328, March 1999.

[13] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. Technical report, ICIR, August 2001.

[14] S. Floyd and V. Jacobson. Random early detection gateways for congestion control. *IEEE/ACM Transactions on Networking*, 1(4):397–412, August 1993.

[15] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong. A control theoretical analysis of RED. In *Proc. IEEE INFOCOM*, pages 1510–1519, April 2001.

[16] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proc. IEEE INFOCOM*, pages 1726–1734, April 2001.

[17] R. Jain. A delay based approach for congestion avoidance in interconnected heterogeneous computer networks. *ACM Computer Communication Review*, 19(5):56–71, October 1989.

[18] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *Proc. ACM SIGCOMM*, pages 89–102, August 2002.

[19] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue algorithm for active queue management. In *Proc. ACM SIGCOMM*, pages 123–134, August 2001.

[20] A. Kuzmanovic and E. W. Knightly. TCP-LP: A distributed algorithm for low priority data transfer. In *Proc. IEEE INFOCOM*, pages 1691–1701, April 2003.

[21] J. Martin, A. Nilsson, and I. Rhee. Delay-based congestion avoidance for TCP. *IEEE/ACM Transactions on Networking*, 11(3):356–369, June 2003.

[22] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED. In *Proc. IEEE/IFIP IWQoS*, pages 260–262, June 1999.

[23] V. Misra, W.-B. Gong, and D. Towsley. A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proc. ACM SIGCOMM*, pages 151–160, August 2000.

[24] R. S. Prasad, M. Jain, and C. Dovrolis. On the effectiveness of delay-based congestion avoidance. In *Proc. PFLDNet*, pages 3–4, February 2004.

[25] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, Internet Engineering Task Force, September 2001.

[26] S. Rewaskar, J. Kaur, and D. Smith. Why don't delay-based congestion estimators work in the real-world? Technical Report TR06-001, Department of Computer Science, UNC Chapel Hill, July 2005.

[27] M. Roughan. Fundamental bounds on the accuracy of network performance measurements. In *Proc. ACM SIGMETRICS*, pages 253–264, June 2005.

[28] P. Sarolahti and A. Kuznetsov. Congestion control in linux TCP. In *Proc. USENIX*, pages 49–62, June 2002.

[29] Z. Wang and J. Crowcroft. A new congestion control scheme: Slow start and search (Tri-S). *ACM Computer Communication Review*, 21(1):32–43, January 1991.

[30] Z. Wang and J. Crowcroft. Eliminating periodic packet losses in 4.3–Tahoe BSD TCP congestion control. *ACM Computer Communication Review*, 22(2):9–16, April 1992.

[31] M. C. Weigle, K. Jeffay, and F. D. Smith. Delay-based early congestion detection and adaptation in TCP: Impact on web performance. *Computer Communications*, 28(8):837–850, May 2005.

[32] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One more bit is enough. In *Proc. ACM SIGCOMM*, pages 37–48, August 2005.