

# Exogenous-Loss Awareness in Queue Management

## Toward Global Fairness

MINA GUIRGUIS  
msg@cs.bu.edu

AZER BESTAVROS  
best@cs.bu.edu

IBRAHIM MATTA  
matta@cs.bu.edu

Computer Science Department  
Boston University  
Boston, MA 02215

**Abstract**—For a given TCP flow, exogenous losses are those occurring on links other than the flow’s bottleneck link. Exogenous losses are typically viewed as introducing undesirable “noise” into TCP’s feedback control loop, leading to inefficient network utilization and potentially severe global unfairness. This has prompted much research on mechanisms for hiding such losses from end-points. In this paper, we show through analysis and simulations that low levels of exogenous losses are surprisingly beneficial in that they improve stability and convergence, without sacrificing efficiency. Based on this, we argue that exogenous loss awareness should be taken into account in any AQM design that aims to achieve global fairness. To that end, we propose an *exogenous-loss aware* Queue Management (XQM) that actively accounts for and leverages exogenous losses. We use an equation based approach to derive the quiescent loss rate for a connection based on the connection’s profile and its global fair share. In contrast to other queue management techniques, XQM ensures that a connection sees its quiescent loss rate, not only by complementing already existing exogenous losses, but also by actively *hiding* exogenous losses, if necessary, to achieve global fairness. We establish the advantages of exogenous-loss awareness using extensive simulations in which, we contrast the performance of XQM to that of a host of traditional exogenous-loss unaware AQM techniques.

**Keywords:** Active Queue Management; TCP Models; Control Theory; Transient Analysis; Performance Evaluation; Network Simulation.

## I. INTRODUCTION

One of the defining characteristics of the Internet is that it caters to an increasingly heterogeneous set of constituents. As such, a network resource (e.g., router or link) is likely to be shared by a set of competing flows with significantly different characteristics. While some may command fairly long round-trip-times as a result of traversing a satellite link, others may be subject to multiple congestions as they traverse a large number of hops with bursty cross-traffic, and worse yet, others may be subject to excessive losses as they traverse noisy wireless channels. To deal with this heterogeneity, networking research has traditionally focused on mitigating the sources of heterogeneity in a piecemeal approach. Examples of this are abound: from wireless TCP research that attempts to contain wireless losses [3], [4], to research on new rate adaptation mechanisms that are suitable for high bandwidth-delay product networks [8], [24], [14]. While dealing with such issues separately leads to simpler “specialized” solutions to different problems (e.g., wireless losses, large bandwidth-delay product flows), it is not clear if *such solutions may be working at cross purposes from one another*. In this paper we identify one such instance—namely, the impact of exogenous losses on TCP’s performance and the advantages of leveraging such losses by an AQM for purposes of improved stability, efficiency, and fairness.

**Motivation:** Packet loss (or marking) events are interpreted by end-to-end transmission control mechanisms (such as TCP) as constituting the “feedback” signal from the link to which such a mechanism must adapt its sending rate. As such, packet losses which are not incidental to that link pose a formidable challenge to an end-to-end transmission control protocol’s ability to claim its fair share of network resources and/or react effectively to changes in network resource availability. In this paper, we use the term *exogenous losses* to refer to such losses. Exogenous losses can be thought of as occurring in a manner that is independent of the source’s short-term behavior or its long-term fair share of network resources. The emergence of exogenous losses could be attributed to two radically different causes: the first is simply a consequence of traversing lossy channels (e.g., wireless first and last hops, satellite links), whereas the second is due to the bursty nature of cross-traffic on non-bottleneck links.

Exogenous losses are problematic as they constitute “noise” with which a transmission controller must reckon. Unchecked, exogenous losses could be quite harmful. By preempting a source from claiming its fair share of the available bottleneck link capacity, exogenous losses may result in an unfair allocation of the bandwidth of overloaded links, or in a decreased utilization of underutilized links. Moreover, unwarranted reactions to exogenous losses may jeopardize stability and convergence properties. Recent research efforts have started to address these issues by adding specialized functionality either *in the middle* or/and *at the end-points* of the network. For example, through the use of a TCP proxy, losses on a wireless link could be hidden from end-points [3], [4]. Alternatively, the negative impact of exogenous losses could be mitigated by enabling a source to diagnose the cause of packet losses and to react differently to different types of losses [6], [35], [13], [5], [8], or by slowing down its reaction to packet losses through the use of “smoother” control rules [37].

For the purposes of this paper, we focus our attention on the first of the above-mentioned negative implications of exogenous losses—namely their impact on *global fairness*. The bandwidth allocated to a flow is globally fair if it reflects the fair share of the capacity of the bottleneck link for that flow, in either an absolute or relative sense, e.g., w.r.t. Round Trip Time (RTT).

**Overview and Contributions:** While countering the effects of exogenous losses is a worthy goal, a more important goal is to assess the extent to which these losses actually impact the behavior of control loops. More to the point, to be able to assess the usefulness of the plethora of traffic control strategies dealing with effects of exogenous losses, we need a

rigorous methodology for the analysis of the emergent behaviors that result from the composition of end-to-end protocols (e.g., increase-decrease rules), network element behaviors (e.g., RED/AQM [15]), and new application-level functionalities. To that end, a particularly promising approach is to marshal techniques from control theory and optimization theory to the modeling and evaluation of complex network transmission control strategies, as exemplified in a number of recent efforts [30], [24], [20]. While useful, these efforts were limited by the fact that they did not explicitly model exogenous losses.

In this paper, we capture the effect of exogenous losses by extending a dynamic fluid model of the widely deployed Transmission Control Protocol (TCP) [22]. As one would expect, we show that high levels of exogenous losses lead to inefficient network utilization and potentially severe global unfairness. Surprisingly though, we also show that low levels of exogenous losses introduce convergence to fairness properties that are both beneficial and desirable! Specifically, we show that if exogenous loss levels do not adversely affect global fairness (i.e., they do not exceed the value necessary for a flow to converge to its global fair share), they tend to improve stability and convergence, without sacrificing efficiency. We elaborate on this point below.

Since TCP, by its nature, adaptively seeks available bandwidth, exogenous losses in effect impose an upper limit on achievable TCP throughput. The extent to which exogenous losses limit achievable throughput makes the crucial difference between desirable and undesirable exogenous losses. In particular, if this limit lies below a connection’s long-term fair share, then exogenous losses cripple that TCP connection. Otherwise, we show that exogenous losses enable the fast and stable convergence of TCP connections to their long-term fair shares of network resources. This is because such exogenous losses serve as early error notifications to the sources, which, similar to RED (Random Early Detection) [15], randomize packet drops across all connections. This randomness prevents an individual TCP connection from monopolizing the bottleneck resource, in addition to preventing several connections from synchronizing their sending behavior which may result in high delay variance (jitter). Thus, low levels of exogenous losses, which do not force TCP throughput to dip below its long-term fair share, can be beneficial in reaching an *efficient, stable* and *fair* allocation of resources.

This observation suggests that the common wisdom of utterly hiding all exogenous losses may indeed be counter-productive. Even if such hiding is harmless, the overhead of implementing it—for example through local error recovery over wireless access links using Snoop [4]—may not be justified.

This observation also suggests that the best place to manage exogenous losses is precisely the bottleneck link (or an AQM proxy thereof). Namely, to ensure global fairness, an AQM design must be *exogenous-loss aware* in that it must hide exogenous losses only when they exceed a certain nominal value corresponding to said globally fair allocation of the link’s available bandwidth.

Towards a constructive application of our findings, we argue that exogenous loss awareness should be taken into account in any AQM design that aims to achieve global fairness. In particular, we propose an *eXogenous-loss aware* Queue

Management (XQM) that actively accounts for and leverages exogenous losses already introduced by other processes in the network. Our version of XQM can be thought of as a per-flow implementation of REM [2]. The goal of such an approach is to work towards providing a *global* fair-share for connections through tuning the level of losses they observe. We envision the deployment of XQM to be at network boundaries—maintaining a profile for each long-lived flow (or flow aggregate) passing through it, while keeping the network core simple. This profile includes estimates of the round trip time and the current connection’s throughput. We use an equation based approach to derive the quiescent packet loss rate to impose based on the connection’s profile and its fair share. In contrast to other queue management techniques, XQM ensures that a connection sees its quiescent loss rate, not only by *complementing* already existing exogenous losses, but also by actively *hiding* exogenous losses, if necessary, to achieve global fairness.

To illustrate XQM’s leverage of exogenous losses, consider a TCP flow for which a quiescent 2% loss rate would result in a global fair share. If exogenous losses amount to 1%, then XQM would introduce additional losses to bring the total loss rate to 2%.<sup>1</sup> On the other hand, if exogenous losses amount to 4% then XQM could leverage any number of mechanisms to hide up to half of these losses to bring the total loss rate down to 2%.

We establish the advantages of exogenous-loss awareness using extensive simulations in which, we contrast the performance of XQM to that of a host of traditional exogenous-loss unaware AQM techniques.

**Paper Outline:** The rest of the paper is organized as follows. In Section II, we present a dynamic model of TCP that incorporates exogenous losses. We analytically derive a lower bound on losses that need to be hidden from TCP sources to ensure efficient operation. In Section III, we classify the effects of exogenous losses into short-term and long-term effects, and show that hiding short-term exogenous losses improves the transient behavior of TCP connections. We capitalize on this finding in Section IV, where we outline and evaluate the performance of our XQM queue management approach. Section V presents XQM’s performance evaluation compared to other AQMs in different setups. We present relevant related work in Section VI and also throughout the paper, when appropriate. We conclude in Section VII with a summary of results and follow-up research.

## II. MODELING TCP + EXOGENOUS LOSSES

In this section, we extend an analytical fluid model, similar to that proposed in [20], [25], [30], [38], to capture the effect of exogenous losses on closed-loop TCP control loops. We present ns-2 [12] simulations to validate our observations from the model.

### A. Model Derivation

A transmission control loop is typically viewed as consisting of two components: a *forward control path*, which governs

<sup>1</sup>Had exogenous losses been hidden through an independent mechanism elsewhere (e.g., using Snoop), an AQM would have been forced to introduce new losses. This is a perfect instance of what we mentioned earlier regarding solutions working at cross purposes from one another.

how much data the sender can inject into the network, and a *feedback path*, via which the network (*e.g.*, AQM at the bottleneck) informs the sender of congestion or available bandwidth. Such feedback is always associated with a delay known as the *feedback delay*, which is the time it takes the feedback signal to reach the sender. Notice that not every transmission protocol has both of these components: TCP [22] is an example of a closed-loop protocol where both components are present, whereas UDP [36] is an example of an open-loop protocol with no feedback component.

We consider a dynamic fluid model of  $m$  TCP connections traversing a single bottleneck of capacity  $C$ . The round trip time  $r_i(t)$  at time  $t$  for connection  $i$  is equal to the round-trip propagation delay  $D_i$  between the sender and the receiver for connection  $i$ , plus the queuing delay at the bottleneck router. Thus  $r_i(t)$  can be expressed by

$$r_i(t) = D_i + \frac{b(t)}{C} \quad (1)$$

where  $b(t)$  is the backlog buffer size at time  $t$  at the bottleneck router. We denote the propagation delay from sender  $i$  to the bottleneck by  $D_{s_i b}$ , which is a fraction  $\alpha_i$  of the total propagation delay.

$$D_{s_i b} = \alpha_i D_i \quad (2)$$

The backlog buffer  $b(t)$  evolves according to the equation:

$$\dot{b}(t) = \sum_{i=1}^m x_i(t - D_{s_i b}) - C \quad (3)$$

which is equal to the input rate  $x_i(\cdot)$  from the  $m$  connections minus the output link rate. Notice that the input rates are delayed by the propagation delay from the senders to the bottleneck  $D_{s_i b}$ .

We assume that the links between the bottleneck and the receivers are subjected to exogenous packet losses, and that all connections see the same level of exogenous losses. It follows that the total packet loss probability  $q(t)$  observed by senders would comprise the congestion-induced loss probability  $p_c(t)$  (due to buffer overflow at the bottleneck) as well as the exogenous loss probability  $p_e(t)$ . Thus, the total loss probability seen by senders is given by

$$\begin{aligned} q(t) &= 1 - (1 - p_c(t))(1 - p_e(t)) \\ &\approx \min(p_c(t) + p_e(t), 1) \end{aligned} \quad (4)$$

where the congestion loss probability  $p_c(t)$  depends on our choice of a queue management implementation at the bottleneck router.

For DropTail,  $p_c(t)$  is simply given by

$$p_c(t) = \begin{cases} 0 & b(t) < B \\ 1 & b(t) = B \end{cases} \quad (5)$$

where  $B$  is the maximum buffer size.<sup>2</sup>

<sup>2</sup>We assume that when operating in a certain regime at time  $t$ , *e.g.*, when  $b(t) < B$ , the probability that the queue is full is small enough that the queue length is practically less than  $B$  over all sample paths. This assumption is validated by the ns-2 simulations presented later in this section.

For RED [15], the congestion loss probability  $p_c(t)$  is given by<sup>3</sup>

$$p_c(t) = \begin{cases} 0 & v(t) \leq B_{min} \\ \sigma(v(t) - \varsigma) & B_{min} < v(t) < B_{max} \\ 1 & v(t) \geq B_{max} \end{cases} \quad (6)$$

where  $\sigma$  and  $\varsigma$  are the RED parameters given by  $\frac{P_{max}}{B_{max} - B_{min}}$  and  $B_{min}$ , respectively, and  $v(t)$  is the average queue size, which evolves according to the equation:

$$\dot{v}(t) = -\beta C(v(t) - b(t)), \quad 0 < \beta < 1 \quad (7)$$

Notice that in the above relationship, we multiply  $\beta$  by  $C$  since RED updates the average queue length at every packet arrival, whereas our model is a fluid model [20], [30].

The throughput of TCP,  $x_i(t)$  is given by

$$x_i(t) = \frac{w_i(t)}{r_i(t)} \quad (8)$$

where  $w_i(t)$  is the size of the TCP congestion window for sender  $i$ .

According to the TCP Additive-Increase Multiplicative-Decrease (AIMD) rule, the dynamics of TCP throughput for each of the  $m$  connections can be described by the following differential equations:

$$\begin{aligned} \dot{x}_i(t) &= \frac{x_i(t - r_i(t))}{r_i^2(t)x_i(t)}(1 - q(t - D_{b_{s_i}}(t))) - \\ &\quad \frac{x_i(t)x_i(t - r_i(t))}{2}(q(t - D_{b_{s_i}}(t))) \\ i &= 1, 2, \dots, m \end{aligned} \quad (9)$$

The first term represents the additive increase rule, whereas the second term represents the multiplicative decrease rule. Both sides are multiplied by the rate of the acknowledgments coming back due to the last window of packets  $x_i(t - r_i(t))$ . In the above equations, the time delay from the bottleneck to sender  $i$ , passing through the receiver  $i$ , is given by

$$D_{b_{s_i}}(t) = r_i(t) - D_{s_i b} \quad (10)$$

## B. Model Assumptions

The fluid model we presented above makes the following assumptions, some of which we have already mentioned.

- (1) Congestion signals are observed by all connections, hence, all react to it.
- (2) The level of exogenous losses experienced by all connections is identical.
- (3) All losses (exogenous and congestion) are observed after the same feedback delay  $D_{b_{s_i}}(t)$ .
- (4) The effect of slow start and timeouts is ignored in the TCP equations, focusing only on the AIMD mechanism.

As we will discuss later, some of the above assumptions do indeed hold in special settings (*e.g.*, when exogenous losses are due to wireless first/last-hop losses). However, in general, the

<sup>3</sup>For simplicity, we follow the same assumptions of other studies by ignoring the uniformization of packet drops [15]. This assumption is relaxed in our ns-2 simulations.

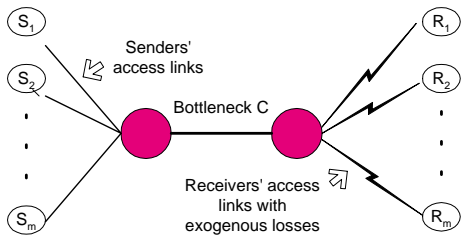


Fig. 1. Dumbbell topology used in numerical evaluation and simulations.

above assumptions may not hold and will need to be relaxed—which we could do in simulation experiments, but not in analysis. For example, since exogenous loss levels and patterns depend on the traversed links, assumption (2) may not hold. Also, since exogenous losses may be produced on any point along a path (let alone at the congestion point), assumption (3) may not hold. Be that as it may, the analytical model captures the essential dynamics necessary to gain valuable insights, which could be confirmed using more empirical means.

### C. Model Application

Low *et al.* [30] studied the dynamics of TCP over RED queues through linearization around equilibrium points.<sup>4</sup> While useful, linearization fails to track the system trajectories across different regions dictated by the non-linear equations.

We refer the reader to [19], where we show the linearization of the system (TCP + exogenous losses) modeled above. In particular, we show how such a system switches between an open loop control, when exogenous losses are high, and a closed loop control, otherwise. This switching between operating regions prevents us from using traditional transient control analysis. Thus, in the remainder of this section, we solve the above set of non-linear equations numerically for a careful and continuous tracking of the model's behavior through different operating regions.

### D. Impact on Efficiency

Figure 1 depicts the topology under consideration. We set the total number of competing connections to 20; we set the capacity  $C$  to  $2,000 \frac{\text{pkts}}{\text{sec}}$ ; and we chose the propagation delay of all connections uniformly at random between 80 and 120 msec. Each connection's fair share of the link is around  $100 \frac{\text{pkts}}{\text{sec}}$ . The total buffer size at the bottleneck is chosen to be 250 packets. RED's minimum and maximum buffer thresholds are set to 50 and 120 packets, respectively. The weight parameter  $\beta$  was set to 0.00001 and  $P_{max}$  was set to 0.1. We also chose  $\alpha_i$  in equation (2) uniformly at random in the interval [0.25-0.5]. Loss modules, generating exogenous losses, are attached to every access link between the bottleneck and each receiver. During the time period  $[0, 20)$  we introduce 0% exogenous losses, during  $[20, 40)$  the rate of exogenous losses is increased to 1% and finally during  $[40, 60]$ , exogenous losses are increased further to 5%.

Figure 2 shows the throughput and the queue size obtained using our numerical solution under both DropTail (top row) and RED (bottom row) for an overall period of 60 seconds,

<sup>4</sup>Linearization assumes (and hence requires) that the system always stays within a certain operating regime.

during which exogenous loss rates are set to 0%, 1%, and 5% over consecutive 20 second periods.

In the first 20 seconds, *i.e.*, under zero exogenous losses, TCP throughput oscillates between low and high sending rates for DropTail. While RED sustains these oscillations only until the average queue size reaches its steady state value (around time 10). In the next 20 seconds, when the level of exogenous losses increases to 1%, TCP throughput converges to its fair share under both DropTail and RED. Notice how the queue size converges to a steady state (non-zero) value, hence the system is well utilized. In the last 20 seconds, exogenous losses (now increased to 5%) result in the convergence of each  $x_i(t)$ , albeit to a value lower than the fair share and the queue size drops to zero, hence the system is under utilized. This observation suggests that low levels of exogenous losses (e.g., 1%) do not degrade the throughput of TCP. But clearly, when exogenous loss rates are increased significantly (e.g., 5%), TCP's throughput suffers and the system becomes under utilized (e.g., below the fair share of  $100 \frac{\text{pkts}}{\text{sec}}$ ).

A transmission control loop is said to be *efficient* if the TCP throughput for that loop matches the bottleneck link capacity. Thus, at steady state, the following two equations should be satisfied for an efficient network utilization. These equations are obtained by setting the derivatives to zero in equations (3) and (9).

$$\sum_{i=1}^m \hat{x}_i = C \quad (11)$$

$$\hat{x}_i = \frac{1}{\hat{r}_i} \sqrt{2\left(\frac{1}{\hat{q}_i} - 1\right)} \quad (12)$$

Clearly, the steady-state TCP throughput  $\hat{x}_i$  is inversely proportional to the square root of the total loss probability  $\hat{q}_i$ , which in turn is directly affected by the exogenous loss rate  $\hat{p}_e$ .<sup>5</sup> For a steady state behavior,  $\hat{q}_i$  must be larger than zero. Having no drops removes the upper limit on the rate/window and this, in theory, will cause it to grow indefinitely.

As the steady-state value of  $\hat{p}_e$  increases, the sending rate would start to decrease, approaching zero. This could prevent TCP throughput  $\sum_{i=1}^m \hat{x}_i$  from reaching  $C$ , *i.e.* equation (11) cannot be satisfied. The value of  $\sum_{i=1}^m \hat{x}_i$  being less than  $C$  means that the system is under utilized. Hence TCP is forced to operate with no buffering at the bottleneck, and no congestion signals going back to senders. When this happens, the TCP transmission control loop is actually broken—it operates as an *open-loop control system* with no feedback from routers.

When exogenous losses are not present, nothing hinders the increase of TCP throughput so as to match its bandwidth share.<sup>6</sup> Once the connection hits its bandwidth share, packets start to accumulate until  $b(t)$  reaches  $B$  under DropTail, or the average queue size starts building up until it exceeds  $B_{min}$  under RED. At that time, congestion signals are generated and

<sup>5</sup>Observe that equation (12) resembles the so-called TCP-friendly equation [18], except that in our model,  $\hat{q}_i$  is not necessarily a Bernoulli probability, but depends on queue management parameters.

<sup>6</sup>The connection is still limited by its round-trip time, but eventually will hit its bandwidth share. We assume that connections are not limited by the advertised receiver's window.

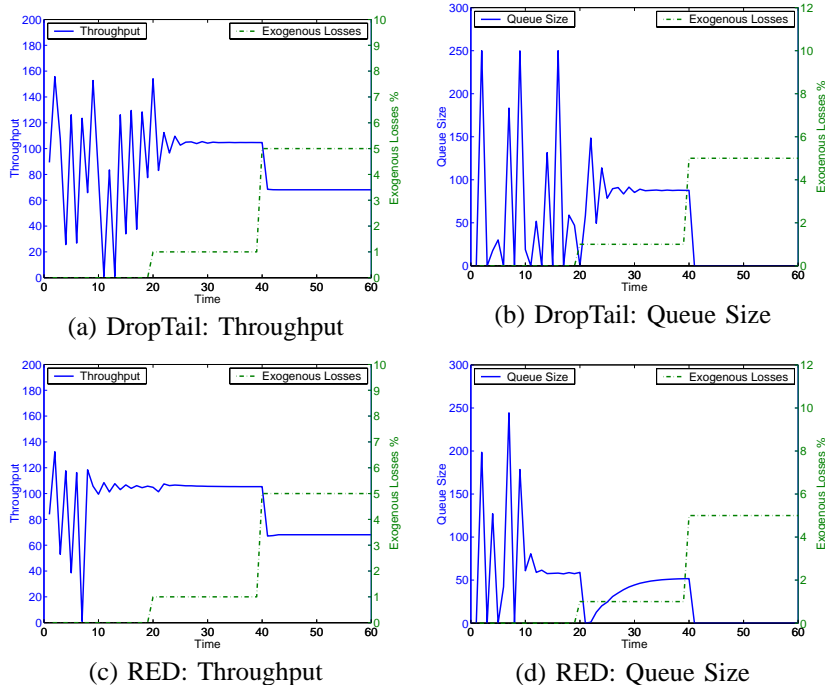


Fig. 2. Impact of exogenous losses (0%, 1%, and 5%) on efficiency of DropTail (top) and RED (bottom) as reflected by throughput (left) and buffering (right).

the sender would back off and this cycle repeats (cf. equation (9)).

The presence of exogenous losses imposes an upper limit on TCP’s throughput and it is crucial *where* this upper limit lies. If this upper limit is close to the connection’s long-term fair share, then these exogenous losses turn out to improve the connection’s convergence to its fair share. This is exactly what happens in the time period [20, 40] in Figure 2. Without such exogenous losses [0, 20), the connection’s throughput shows large oscillations under DropTail.

Consider the same setup described above, except that all connections have an identical propagation delay of 100 msec. If the goal is to allocate an equal share of the bandwidth to each TCP connection, then using equation (11), each connection’s (long-term) fair share,  $\hat{x}_i$ , would equal  $100 \frac{pkts}{sec}$ . Equation (12) can be solved for  $\hat{q}_i$  for a given round-trip time  $\hat{r}_i$ , which depends on the steady state buffer occupancy. Indeed, Figure 3(a) shows that when exogenous losses are in the range of 1% to 2%, the throughput converges to the fair share value. Below or above these values, the fair share is not matched due to either oscillations (left) or under utilization (right), respectively. Notice that exogenous losses around 2% represent the case where the steady state round-trip time is equal to the propagation delay with zero buffer occupancy.

To validate the above observations, we conducted a simple ns-2 simulation on a simple dumbbell topology similar to the one in Figure 1. The bottleneck link capacity is set to 16Mb and its propagation delay is set to 1 msec. A total of 20 TCP connections are created between the senders and the receivers. Senders and receivers connect to the routers through access links with propagation delay chosen uniformly at random between 1 and 4 msec. The receivers’ access links are associated with error modules that would represent the effect of exogenous

losses. At time 0, we start with no exogenous losses, at time 100 we set the exogenous losses to 2% across all receivers’ access links, at time 150, exogenous losses are increased to 7% and the experiment ends at time 200. We use DropTail at the bottleneck link and we ignore the first 50 seconds of the simulation experiment. Figures 3(b) and 3(c) show the effect of exogenous losses on fairness and on queue size, respectively. Fairness is computed across all connections every 1 second interval, using Chiu and Jain’s Fairness Index [9], which is given by:

$$f(x_1, x_2, x_3, \dots, x_m) = \frac{(\sum_{i=1}^m x_i)^2}{m \times \sum_{i=1}^m x_i^2} \quad (13)$$

Notice how the fairness index improved significantly when exogenous losses increased to 2% since such value in effect helped the connections converge to their fair share, and also helped the buffer size converge. Increasing exogenous losses to 7% leads to a deterioration in the fairness index and leads to under utilization of the network since the queue size gets closer to 0.

Many protocols have been developed for hiding *all* exogenous losses from the sender [4], [3]. For example, in Snoop [4] the connection between the server and the client is intercepted by the proxy in the middle. The proxy buffers data packets to allow link-layer retransmission when duplicate acknowledgments, indicating packets lost over the wireless proxy-client link, arrive at the proxy. Snoop does not allow such duplicate acknowledgments to pass back to the sender to prevent it from doing fast retransmit and recovery (*i.e.*, halving its sending rate). While such protocols attempt to improve efficiency by removing the upper limit imposed on throughput by exogenous losses, they could be hindering the convergence to fairness! Furthermore, hiding further packet losses from connections that are already getting their fair share would not

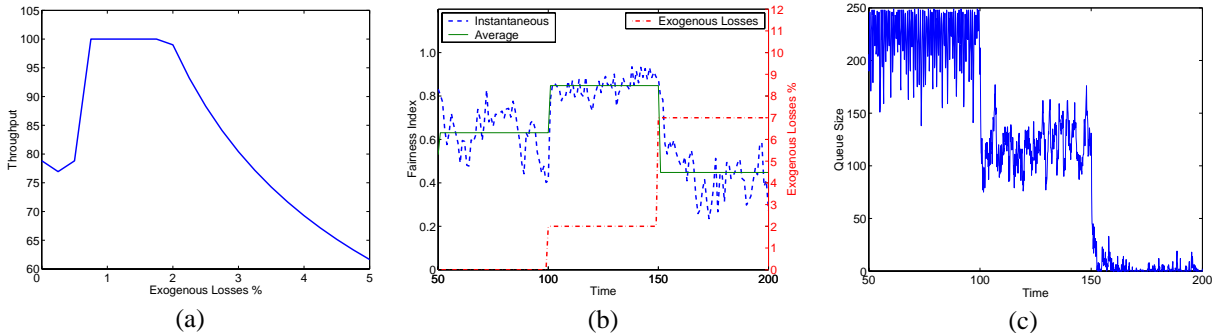


Fig. 3. Model-predicted effect of exogenous losses on efficiency (left) and validation via simulations of effect on fairness (middle) and queue size (right).

be beneficial, but would only add the overhead of complete hiding (e.g., the cost of buffering and local retransmission at the Snoop proxy)—not to mention the fact that another process (in this case an AQM) would have to reintroduce packet losses.

Ideally, we would like to always report a value of  $\hat{q}_i$  to sender  $i$  that corresponds to its fair share, since this would mean that the network is utilized efficiently while, at the same time, connections have a fair chance to compete. In the next section, we address the challenges behind active management of exogenous losses inside the network to achieve this goal.

### III. ACTIVE TUNING OF EXOGENOUS LOSSES

As discussed in the previous section, for given bottleneck link and RTT characteristics, there exists a desirable value for the loss rate that would promote both convergence and efficiency of a TCP control loop. We use the term *quiescent loss rate* to refer to this desirable value. For example, a quiescent loss rate of 2% yielded both efficiency and convergence for the experimental setting used in Figure 3(a). In this section we examine the advantages and disadvantages of various active approaches for relaying such quiescent loss rates to senders.

**Exogenous Loss Unaware Signaling (Local Fairness):** The traditional approach ignores the presence of exogenous losses and imposes a loss rate value that would improve some local metric (e.g., stability of the buffer backlog). An example of such an approach is RED (or other variants thereof, e.g. [34]), whereby the dropping or marking of packets is conditioned on the local queue occupancy in order to stabilize the queue. Therefore, a natural question to ask is whether tuning the average drop rate of such queue management techniques over a long time scale (to match the quiescent loss rate) would yield the desirable efficient convergence to fair share. The results in Figure 3(d) suggest that this would be the case, but only over long time scales. As evident from the results in Figure 2(d), RED exhibits transient inefficiencies when faced with variability in exogenous loss rates over short time scales. Specifically, at time 20, TCP’s queue size drops to zero, before converging again to the new steady-state value around 50. This transient anomaly does not occur under DropTail.

The undesirable transient behavior exhibited under RED is due to RED’s unawareness of exogenous losses, which is exacerbated by the lag time necessary for the average queue size (seen by RED) to reflect the “real” conditions. To elaborate on this, consider the case when RED’s average queue length is above  $B_{min}$ . Now consider a situation whereby

TCP flows react to a sudden increase in the exogenous loss rate (by backing off), which in turn would cause the RED queue to drain. Since RED uses the *average* queue length as an indicator of congestion, it would take RED some time to realize this new “drained” state. As a result, RED would keep on generating congestion signals (by dropping or marking packets) according to the stale higher value of its average queue length—causing further degradation in efficiency.<sup>7</sup> Obviously, under such conditions, the congestion-minded design of RED is challenged by exogenous losses, since it is no longer true that the sender reduces its rate only in response to congestion signals! As soon as the average queue size catches up with the new value below  $B_{min}$ , RED ceases to send its feedback signal. At that point, TCP is in fact operating as an open loop control system and starts to increase its sending rate.

The inability of RED (as a representative of exogenous-loss unaware AQMs) to cope with exogenous losses is further complicated by issues of heterogeneity in flow characteristics (e.g., the possibly wide range of exogenous loss rates across flows, and the variability in round trip times). Clearly, no AQM would be able to address issues of global fairness without some accounting of flow characteristics. Indeed, if RED were to achieve global fairness, it would require more than parameter tuning, namely awareness of the presence of these losses and invoking the right control rules.

**Exogenous Loss Aware Signaling (Global Fairness):** The above discussion suggests that, towards global fairness, it is crucial for an AQM to take into account the presence of exogenous losses. In particular, FRED [28] implicitly makes few steps towards this goal by protecting fragile flows—flows with small window sizes. It is important to note that this is more of a side effect than a “by design” feature since FRED protects flows with excessively small window sizes, independent of whether flow fragility is due to exogenous losses, or simply a reflection of that flow’s fair share. In contrast, XQM as will be presented later in this paper, makes the decision of when to introduce losses, when to hide losses, and when not to interfere, based on the level of exogenous losses present. In effect, XQM utilizes such external losses, toward its own feedback signal. Thus, it provides the minimum interference and only when needed.

Assuming that the exogenous loss rate for a flow can be relayed to (measured/estimated by) an XQM, then it is possible

<sup>7</sup>This phenomenon was noted in [21], prompting the need for decoupling the queue size from the loss probability.

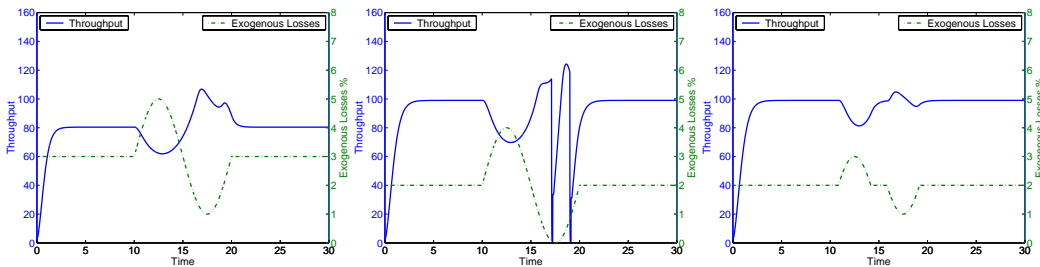


Fig. 4. Effect of short-term cyclic (sinusoidal) exogenous losses on efficiency (left); effect of hiding exogenous losses over long time scales (middle); and effect of hiding exogenous losses over long and short time scales (right).

to ensure that the sender will only see the quiescent end-to-end loss rate by having the XQM adjust its own control rules accordingly. Namely, if exogenous losses are below the quiescent rate, then it is possible to “introduce” losses to promote efficient convergence to a fair share. This could be done through randomly dropping or marking packets. If exogenous losses are higher than the quiescent rate then it would be necessary to “hide” such losses from the sender. This could be done in many ways, including link layer retransmission, forward error correction techniques, or replication over multiple paths (i.e. dispersity routing [33]).

In practice, requiring that an XQM be able to obtain an accurate estimate of exogenous loss rates is hard to achieve. Thus, alternately, an XQM could dynamically adjust its behavior (i.e., figure out the exact levels of losses to be introduced or hidden) in response to each connection’s performance, without having to explicitly know the exogenous loss rate for such connections.

In the above discussion, we have assumed that exogenous loss rates are static. In a real setting, this is likely not to be the case. Thus, it is important to assess the impact of such variability, both over long and short time scales.

With the same setup described in the previous section, Figure 4(left) shows the effect of short-term fluctuations (captured by a sinusoidal cyclic behavior) of  $\pm 2\%$  during the interval [10,20], which is superimposed on an average, persistent (i.e., long-term) level of exogenous losses of 3% during the interval [0,30]. Clearly, the higher-than-desirable long-term exogenous losses result in lower efficiency during the intervals [0,10] and [20,30], with throughput of  $80 \frac{pkts}{sec}$  (as opposed to the  $100 \frac{pkts}{sec}$  fair-share). Given such a mix of long-term and short-term effects, how could an agent massage the exogenous losses observable by senders?

*Long-Term Adjustment:* As a first approximation, such an agent may attempt to bring the long-term average of observable exogenous losses to the quiescent rate. We refer to such an approach as *long-term adjustment* of exogenous losses, whereby an agent would hide losses in the amount of  $p_h(t)$ , allowing senders to observe loss rates of  $p_r(t)$  that are equal to the difference between the real losses  $p_e(t)$  and  $p_h(t)$ .

$$p_r(t) = \max(0, p_e(t) - p_h(t)) \quad (14)$$

The above equation effectively shifts  $p_e(t)$  down by a value that is equal to  $p_h(t)$  in order to match a desirable quiescent loss rate.

A long-term reduction<sup>8</sup> of exogenous loss rates will result in larger TCP congestion window sizes (i.e., higher throughput). Thus, by appropriately setting the value of  $p_h(t)$ , we are able to bring connections operating in an inefficient region to an efficient one.

Figure 4(center) shows the results of applying such a policy, where the average level of exogenous losses over a long time scale (e.g., calculated over the interval [0,30]) is brought down from 3% to 2%. As expected, the resulting throughput is decidedly better during the intervals [0,10] and [20,30], but the short-term variability in exogenous losses during the interval [10,20] results in wide oscillations that are clearly undesirable.

*Short-Term Compensation:* To tackle variations in exogenous loss rates over shorter time-scales, we may extend our policy to allow for a *dead-band controller* which filters out short-term changes in exogenous losses (by hiding and/or introducing losses) that are within a prescribed range (e.g.,  $\pm 1\%$ ) around the long-term average. We refer to this as *short-term compensation*.

Short-term compensation causes the sender to see smoother loss patterns (not affected by the real dynamics of exogenous losses). Let the losses  $p_e(t)$  be evolving around a certain average denoted by  $p_a(t)$ . Under short-term compensation, the rate of exogenous losses  $p_r(t)$  reported back to senders is given by:

$$p_r(t) = \begin{cases} p_e(t) - p_h(t) & \text{if } p_e(t) > p_a(t) + p_h(t) \\ p_e(t) + p_h(t) & \text{if } p_e(t) < p_a(t) - p_h(t) \\ p_a(t) & \text{otherwise} \end{cases} \quad (15)$$

The equation above reports to senders an average exogenous loss rate of  $p_a(t)$ , unless short-term variability in exogenous loss rates is beyond what can be hidden or introduced, in which case remnants of this variability are observable by senders.

Figure 4(right) shows the results from applying short-term compensation as well as long-term adjustment policies. Clearly this two-pronged strategy results in significant smoothing of achievable throughput during times of short-term variability in exogenous loss rates, while keeping the long-term average around the quiescent value for efficient convergence to a connection’s fair share.

#### IV. EXOGENOUS-LOSS AWARE QUEUE MANAGEMENT

In this section, we discuss and evaluate our proposed eXogenous-loss aware Queue Management (XQM) approach.

<sup>8</sup>The case in which exogenous losses are below the quiescent rate is treated similarly using a long-term increase in the losses introduced by XQM.

XQM’s main goal is to tune the exogenous losses that are already present in the network to improve fairness, without sacrificing efficiency. Thus it provides each flow<sup>9</sup> with its fair share of resources. XQM maintains a profile for each long-lived flow passing through it.<sup>10</sup> This profile includes the round trip time estimate  $\hat{r}_i$ , the current flow’s throughput  $x_i(t, MP)$ , measured as the number of packets sent over the past *Measurement Period* (MP), the flow’s steady state quiescent loss rate  $\hat{q}_i$  and the current imposed loss rate  $q_i(t)$ . Round-trip times could be estimated from the middle by XQM using “measurement-in-the-middle” techniques (e.g., [23]). From equation (12), the quiescent loss rate for each flow is given by:

$$\hat{q}_i = \frac{2}{(\hat{x}_i \times \hat{r}_i)^2 + 2} \quad (16)$$

On a packet arrival, XQM identifies the flow this packet belongs to and drops the packet with probability  $p_i(t)$  that is a function of the current quiescent loss rate:

$$p_i(t) = \frac{q_i(t)}{1 - k \times q_i(t)} \quad (17)$$

where  $k$  is the number of packets queued since the last packet drop. This insures that XQM spreads losses uniformly over time, as discussed in RED’s design[15]. Every *Control Period* (CP), XQM updates the current imposed loss rate  $q_i(t)$  for each active flow. Next, we propose three different schemes for implementing such an update.

#### XQM ON-OFF: Throughput Matching using On-Off Control

In this implementation, a flow is classified as being active or inactive based on the measured throughput. If the measured throughput is above the flow’s fair share, XQM marks the flow as active and starts imposing losses to match this flow’s established quiescent loss rate. If the flow becomes inactive, XQM ceases to impose losses. This simple on-off controller can be expressed as:

$$q_i(t + CP) = \begin{cases} 0 & \text{if } x_i(t, MP) \leq \hat{x}_i \\ \hat{q}_i & \text{otherwise} \end{cases} \quad (18)$$

We henceforth refer to this implementation as XQM ON-OFF. While simple, XQM ON-OFF may lead to oscillations in throughput, since it alternates between two regions, a region with no losses and a region with some losses. As such, XQM ON-OFF does not smooth out the control signal, thus we look at other controllers.

#### XQM PI-T: Throughput Matching using PI Control

In this implementation, an error signal is obtained by comparing the current throughput of the flow with its target fair share. XQM imposes losses at the tune of  $q_i(t + CP)$  for connection  $i$  where:

$$q_i(t + CP) = q_i(t) + \delta \times (x_i(t, MP) - \hat{x}_i) \quad (19)$$

<sup>9</sup>For the purposes of this discussion, we don’t insist on a strict definition of a flow. One can think of a 4-tuple definition (source IP, destination IP, source port #, destination port #), a 2-tuple definition (source IP, destination IP) or simply aggregates of these.

<sup>10</sup>We assume that XQM will allocate some fixed amount of its link capacity to short flows so they are not subject to XQM control rules.

where the controller is invoked every CP and the throughput is measured over the last MP. We henceforth refer to this implementation as XQM PI-T. The value of  $q_i(t + CP)$  is set to 0 when the above equation results in a negative value. It is set to  $\hat{q}_i$  if the above equation results in a value greater than  $\hat{q}_i$ . In other words, we never impose losses in excess of  $\hat{q}_i$ , and we use the fact that  $q_i(t + CP)$  is below zero to trigger a mechanism for hiding excessive exogenous losses (as we will detail later).  $q_i(0)$  is chosen to be the quiescent loss rate  $\hat{q}_i$  for better initial control.

#### XQM PI-TB: Throughput and Buffer Matching using PI Control

It is important to maintain the buffer size at a target level since this will guarantee efficiency and at the same time reduces jitter. Throughput is also improved when this target buffer size is low, since this would mean reducing the round-trip time. The implementations above do not allow for target buffer matching. To do so, we modify equation (19) as follows:

$$q_i(t + CP) = q_i(t) + \delta(x_i(t, MP) - \hat{x}_i) + \phi(b(t) - \hat{b}) \quad (20)$$

where  $b(t)$  is the current buffer size and  $\hat{b}$  is the target buffer size. Again,  $q_i(t + CP)$  is set to 0 or to  $\hat{q}_i$  if it is below zero or greater than  $\hat{q}_i$ , respectively. We henceforth refer to this implementation as XQM PI-TB. The four parameters CP, MP,  $\delta$  and  $\phi$  play a very important role in the general behavior of XQM. We summarize our experience with these parameters next.

First we focus on the interplay between the measurement and control periods. Since, the control period (CP) directly affects the frequency with which the controller is invoked, we choose a small value (around 10 msec). Having a larger CP will cause XQM to be less responsive, while having a smaller CP would only add to the overhead of invoking the controller. For a correct estimate of the connections’ throughput, the measurement period (MP) should be a multiple of the congestion epochs. That is, it should be long enough to capture multiple packet drops. Having a shorter MP, will cause errors in throughput estimation due to window fluctuation. However, having a longer MP, will limit XQM’s ability to capture short-term behaviors. In other words, a long MP would result in a smooth estimate for an otherwise jittery signal—in effect making the average metric over the MP more or less approach its exponentially weighted moving average (EWMA) updated every CP.

We now turn our attention to  $\delta$  and  $\phi$ , the weights in equation (20). These weights play an important role in the decision process. They specify the tradeoff between efficiency and fairness. In particular, a higher value of  $\phi$  will tend to improve efficiency, while a higher value of  $\delta$  will tend to improve fairness. There are four possible cases, which we consider next.

The first two of these cases are straightforward; they correspond to situations in which the two constituent controllers in equation (20) are in agreement as to whether the loss rate imposed on a flow is to go up or down. Namely, these two cases occur when the bandwidth (for a flow) and the buffer size are *both* below their prescribed values, or *both* above their prescribed values. Clearly, for the former, XQM will decrease the loss rate imposed on the flow, and for the latter, XQM will increase the loss rate imposed on the flow.

The other two cases correspond to situations in which the two constituent controllers in equation (20) are at odds with one another regarding whether the loss rate imposed on a flow is to go up or down. For example, what if a connection’s throughput is less than the targeted throughput, but the buffer size is larger than the targeted buffer size? In our experiments (some of which we will present in the next section), we found that having a value of  $\phi$  relatively larger than  $\delta$  is helpful in cases when some connections are unable to get their fair share of the throughput (e.g., they are source limited). Under such conditions, we allow other connections to grab the available bandwidth by giving a higher “weight” to buffer-size matching. On the other hand, in our experiments, we found that having a value of  $\phi$  that is much greater than  $\delta$  tends to hurt connections that are already below their fair share (in an attempt to whip the buffer into matching its prescribed value). By tuning the values of  $\delta$  and  $\phi$ , XQM is able to expose the tradeoffs between efficiency and fairness.

To summarize, XQM has two key design features. The first is that XQM decouples the measurement period from the control period. This decoupling allows XQM to improve fairness (over longer time scales) without sacrificing efficiency (over shorter time scales). This is achieved by exercising control over short time scales based on throughputs measured over longer time scales. The second key feature of XQM is that it exposes the tradeoffs between efficiency (over shorter time scales) and fairness (over longer time scales). The selection of the characteristic time scales for measurement and for control, as well as the adjustment of the tradeoff between efficiency and fairness are *both* possible to manage dynamically based on the traffic profile. This dynamic tuning of XQM’s operation (based on traffic profiling) is the subject of on-going research.

## V. SIMULATION RESULTS

In this section, we present results from extensive ns-2 [12] simulation experiments we conducted to assess the advantages of exogenous loss awareness in general, and XQM’s performance when compared to a host of traditional exogenous-loss unaware AQMs in particular.

### A. Effect of Losses Due to Cross-traffic on Non-Bottleneck Links

Ideally, for a TCP connection to reach its fair share, it should get its loss signal from the bottleneck link only. In practice, due to network dynamics, a TCP connection could experience packet losses on (multiple) non-bottleneck links. The more congested/bursty path segments a connection traverses, the noisier the feedback signal (due to exogenous losses on non-bottleneck links). In this section, we show how XQM in effect “adopts” packet losses on other links as its own—in effect using them towards the total packet losses it needs to impose on the flow. Figure 5 illustrates the actions taken by XQM as a function of exogenous loss levels. XQM would decrease the losses it imposes as exogenous loss rates increase toward the quiescent loss rate, moreover it would increase the value it hides as exogenous loss rates increase above the quiescent loss rate. However, if the exogenous losses are close to the quiescent loss rate, XQM will not interfere.

Figure 6 depicts the topology under consideration. We have three links AB, BC and CD of capacity 100 Mbps, 150

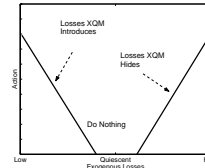


Fig. 5. XQM loss/hiding behavior versus observed exogenous loss rates.

Mbps and 150 Mbps, respectively.<sup>11</sup> All links have a one-way propagation delay of 1 msec. A total of 10 FTP connections, with unlimited data to send, traverse the topology from A to D. We refer to these as the AD flows, with IDs from 1 to 10. In addition, three groups of 10 FTP connections each, representing cross-traffic with unlimited data to send, traverse exactly one of the links in the topology. We refer to these as the AB, BC and CD flows. Sources as well as receivers of these FTP flows connect to the routers A, B, C and D through access links.

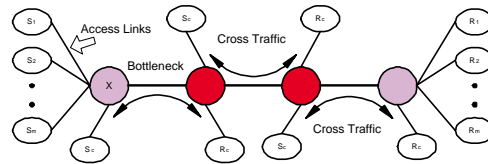


Fig. 6. The three-link topology used in ns-2 simulation experiments.

In the topology of Figure 6, the first link (AB) uses the AQM under consideration—namely the various XQM implementations, RED [15], FRED [28], REM [2] and PI[21]. The second and the third links (BC and CD) use RED as an AQM.

Unless otherwise stated, in our experiments, we set RED’s minimum and maximum buffer thresholds to 50 and 120 packets, respectively. The weight parameter  $\beta$  was set to 0.0001 and  $P_{max}$  was set to 0.1. The buffer size is chosen to be 250 packets at each link. All packets are 1,000 bytes in size. Also, unless otherwise stated, in our experiments, we set XQM’s parameters CP, MP,  $\delta$  and  $\phi$  to be 0.01 msec, 10 seconds, 0.00001 and 0.00004, respectively.

**Experiment 1:** We start with a simple case where all connections have the same round-trip time. To do so, we adjust the propagation delay on the access links so that all connections have the same round-trip time, taking into account the queuing delay at the links BC and CD for connections AD. We present the results across the first link (the bottleneck).

Figure 7 compares the performance of the different schemes—each shown in plots on separate rows. Plots in the first (leftmost) column represent the average throughput achieved by each connection, which is computed over the interval [20-100]. Plots in the second column represent the instantaneous throughput computed every one second interval. We only present two connections, one that belongs to the set AD and another one that belongs to the set AB. Plots in the third column represent the average losses seen by each flow over the interval [20-100]. Connections AD see losses that are also on other links, i.e., “Exogenous Losses”. Finally, plots in the last (rightmost) column represent the instantaneous queue size. In

<sup>11</sup>Such topology can also be viewed as an overlay link where the first link interfaces the whole network.

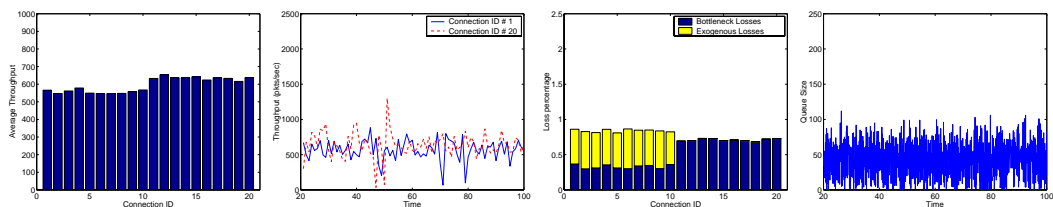
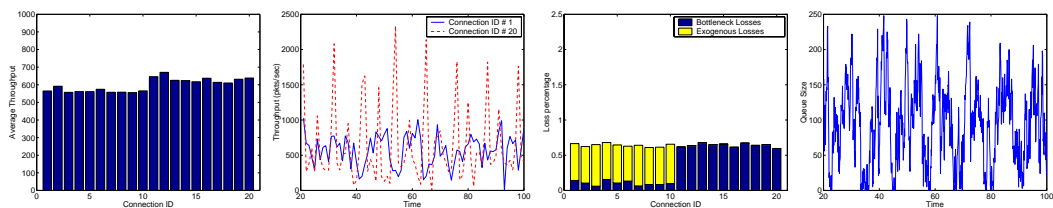
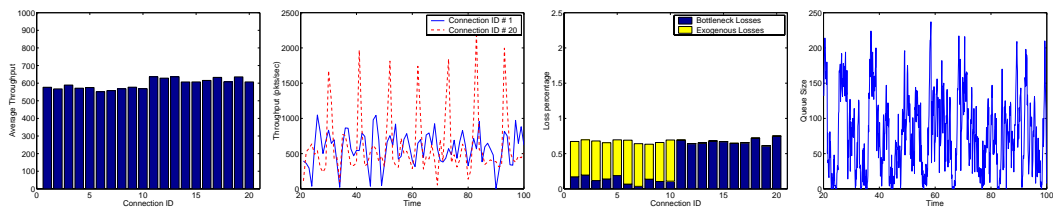
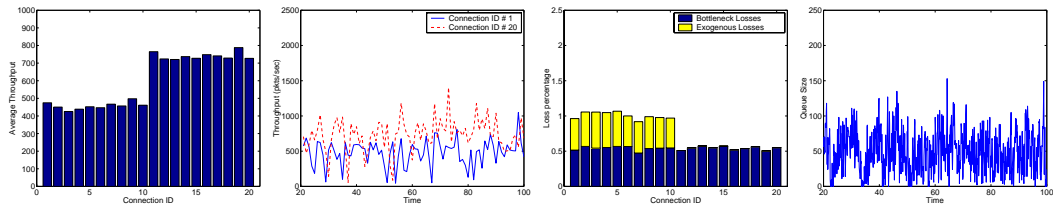
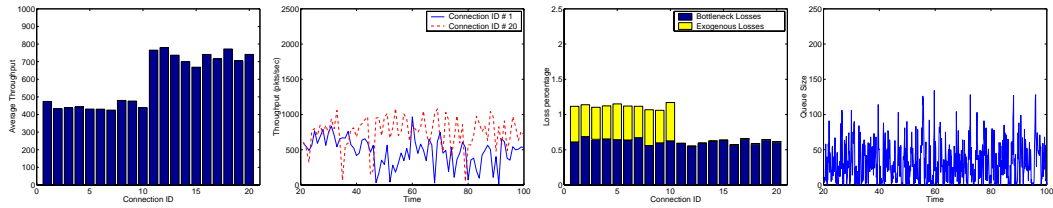
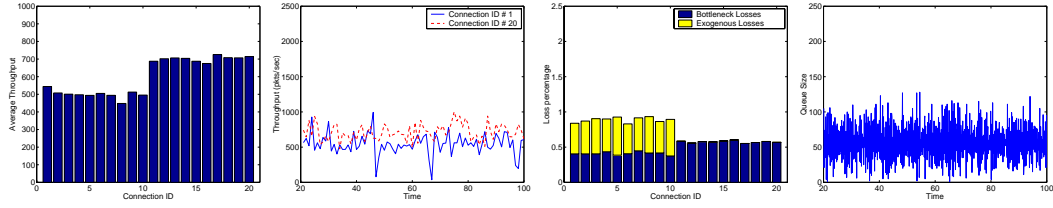
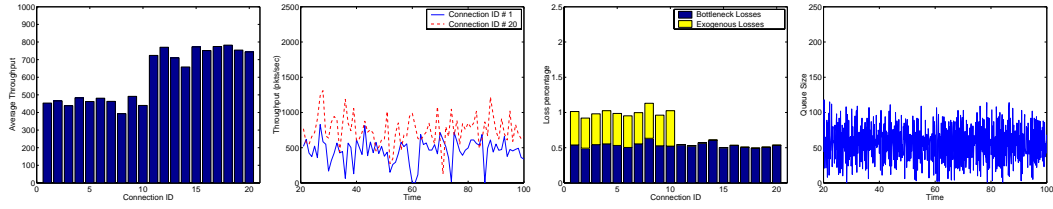


Fig. 7. Comparative performance of various AQM approaches, showing long-term throughput for all flows (left), instantaneous throughput for two exemplary flows, average loss rates seen by all flows, and instantaneous buffer size (rightmost).

all plots, we ignore the first 20 seconds (for simulation warm-up purposes) and we calculate all metrics starting from time 20.

Despite the fact that all connections have the same round-trip time, connections that traverse multiple congested links (i.e. connections 1 to 10) end up with less throughput. Since RED, REM and PI apply the same loss rate across all connections, they don't compensate for any exogenous losses on other links. FRED, on the other hand, compensates a little bit as evident by the values of the loss rate for FRED. FRED applies lower loss rate values to connections AD and higher values to AB. Still, AD connections can't reach their fair share. On the other hand, all of the XQM designs we considered (XQM ON-OFF, XQM PI-T, and XQM PI-TB) apply just enough losses so that the total loss rate seen by any connection is the same—hence the better fairness delivered by XQM.

Focusing on the first two XQM designs (namely, XQM ON-OFF and XQM PI-T), we observe that they both suffer from a fairly unstable (bursty) queue size and instantaneous throughput. However, once this is taken into account (as evident in the performance of XQM PI-TB), XQM maintains the buffer size at the target level of 50 packets.

**Experiment 2:** In the previous experiment, we fixed the propagation delay so that all connections experience the same round trip time, and thus giving us an opportunity to observe/study the effect of exogenous losses on multiple (non-bottleneck) hops on connections AD. Now, we repeat the same experiment, except that all the access links (for connections AD as well as for cross traffic connections AB, BC and CD) have a propagation delay that is uniformly distributed between 5 and 10 msec. Now, connections AD have a longer round-trip time compared to AB, since they traverse more links. So two questions arise, how much worse would connections AD fare? and how effective is XQM in dealing with such scenarios?

Increasing the round trip time for connections AD would only make the situation worse (i.e., if they can't get their fair share with a shorter round trip time, they will certainly not get their fair share with a longer one, due to the bias against connections with longer round-trip times). Indeed, this is the case under RED, FRED, PI and REM; connections AD can't get their fair share. Due to space limitations, we only present the performance under RED (as a baseline) and FRED (which had the best performance among all other exogenous loss unaware AQMs). For XQM, we only present the performance of XQM PI-TB.<sup>12</sup>

In our experiments we found that REM and PI behave quite similarly to RED, imposing the same loss rate across all connections. FRED imposes the minimum losses, however it still taxes AD connections, putting them at a disadvantage with respect to reaching their global fair share of bandwidth. XQM, on the other hand, doesn't drop any packets from AD connections. The plots in rows (a), (b) and (c) of Figure 8 show the performance of RED, FRED, and XQM PI-TB, respectively (also, the Fairness Index is noted). Since connections AD are limited by exogenous losses, they fail to grab their allocation. That is why the buffer size in row (c) is below the target

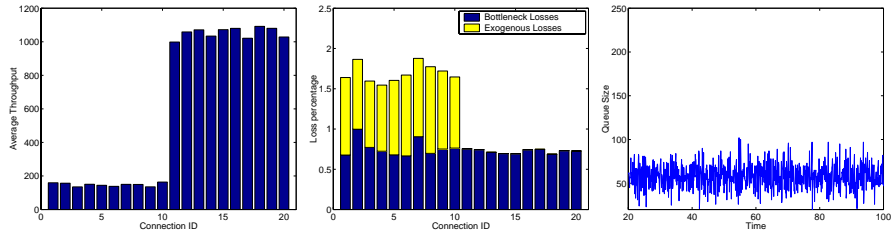
value of 50. Notice that as far as XQM is concerned (for now), it is not imposing any losses on connections AD. At this time, it would make sense to allow other connections to get the available bandwidth. That is to say, we need to give a higher weight to efficiency over fairness, as we discussed in the previous section. Indeed, row (d) shows the performance of a design in which we give more weight to buffer matching ( $\phi$  is increased to 0.0001). This, in effect, allows connections AB to grab the available bandwidth without hurting connections AD. Increasing  $\phi$  further, causes connections AD to start losing packets.

**Experiment 3:** The previous experiment illustrated that for a particular connection to get its fair-share, it may not be enough for XQM to simply “*not introduce additional losses*”. In particular, when exogenous loss rates are fairly high, some losses should be hidden from the sender. We propose (and present results of) a technique that can be deployed at the edge routers, which marks packets so core routers would not drop these packets. The plots in rows (e) of Figure 8 show two different ways of implementing exogenous loss hiding in XQM. In the first, once  $q(t)$  reaches zero (implying that the XQM need not introduce any additional losses), we trigger hiding as well whereby we mark all packets so they won't get dropped at down-stream routers. Once the  $q(t)$  goes above zero, hiding is stopped. A drawback of this scheme is the potential for oscillations due to the alternation in control rules. Nonetheless, this scheme is able to improve the fairness as well as maintaining the queue size at the target level. The plots in row (f) of Figure 8 provide a remedy for this, whereby the XQM smoothly tunes the level of hiding (i.e., incrementally increasing it) when  $q(t)$  is negative. This technique is not susceptible to the negative impacts from a sudden alteration in control rules.

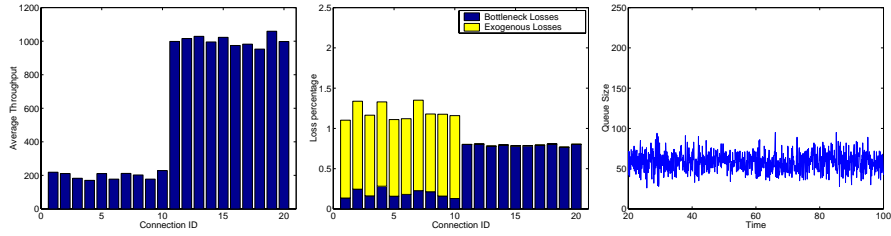
### B. Weighted Fair Sharing using XQM

Through adjustments of the quiescent loss rate for each connection, XQM can easily achieve any weighted allocations of throughput. In order to demonstrate this, we show how XQM can provide absolute fairness among connections with different round trip times. We compare XQM PI-TB to DropTail, RED, FRED, REM and PI. In this experiment, we consider a bottleneck link with 16Mb capacity and a 10ms propagation delay shared by 20 TCP connections with unlimited data to send. We vary the propagation delay on the access links to have different round trip times. The propagation delay for the shortest connection is 24 msec, and for the longest connection it is 176 msec. Each connection is a step of 8 msec increase in the propagation delay. The buffer size is chosen to be 250 packets. All packets are 1000 bytes in size. RED's minimum and maximum buffer thresholds are set to 50 and 120 packets, respectively. The weight parameter  $\beta$  was set to 0.0001 and  $P_{max}$  was set to 0.1. FRED's parameters are the same as RED's. In REM and PI, we used the default parameters the authors recommended in their ns implementations. Figure 9 compares the performance of the different schemes—each shown in plots on separate rows. Plots in the first (leftmost) column represent the average throughput achieved by each connection, which is computed over the interval [20-100]. Plots in the second column represent the instantaneous throughput computed every one second interval. We only present two connections, one with the shortest RTT (ID 1) and another

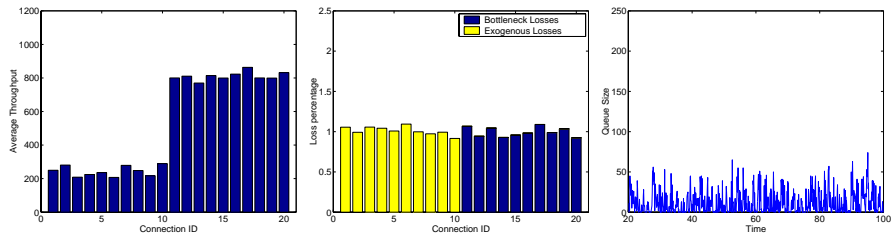
<sup>12</sup>For the remainder of this paper, we use XQM and XQM PI-TB interchangeably.



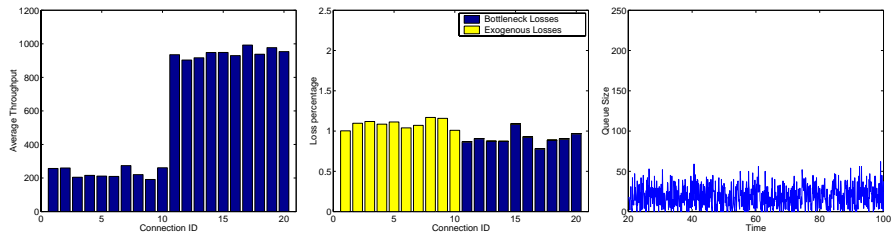
(a) RED [Fairness Index = 0.63]



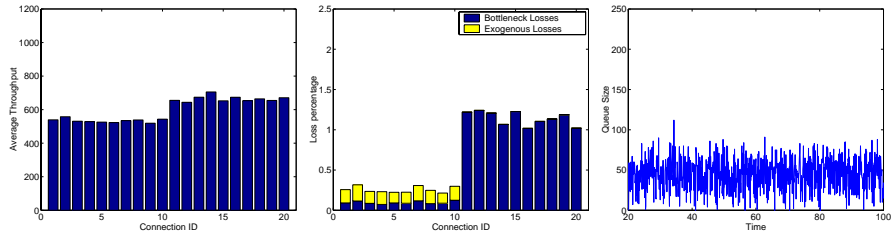
(b) FRED [Fairness Index = 0.69]



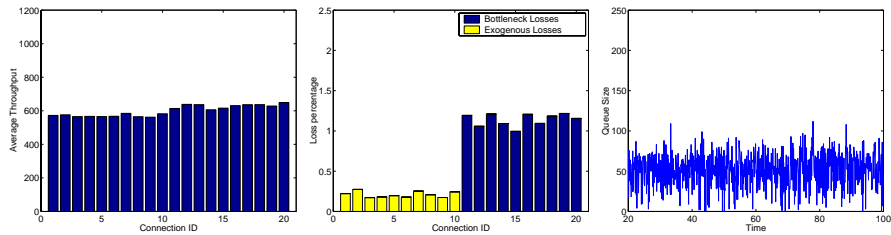
(c) XQM PI-TB [Fairness Index = 0.77]



(d) XQM PI-TB (with  $\phi$  increased to 0.0001) [Fairness Index = 0.73]

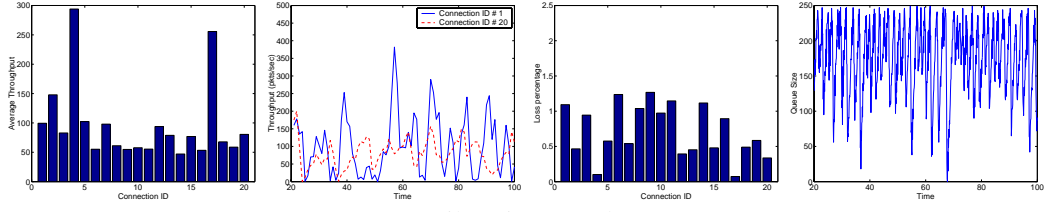


(e) XQM PI-TB (with 100% hiding enabled when  $q(t) \leq 0$  and disabled otherwise) [Fairness Index = 0.98]

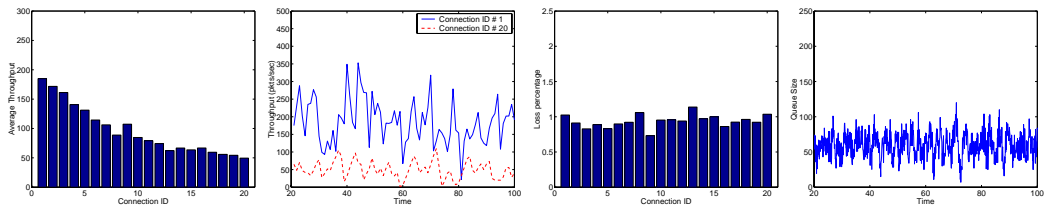


(f) XQM PI-TB (with hiding enabled smoothly as a function of  $q(t)$ , when  $q(t) \leq 0$ ) [Fairness Index = 0.99]

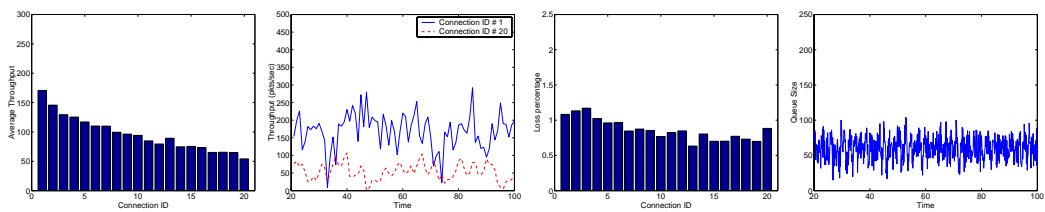
Fig. 8. Comparative performance of various AQMs, showing average throughput (left), overall loss rate (middle), and instantaneous buffer size (right).



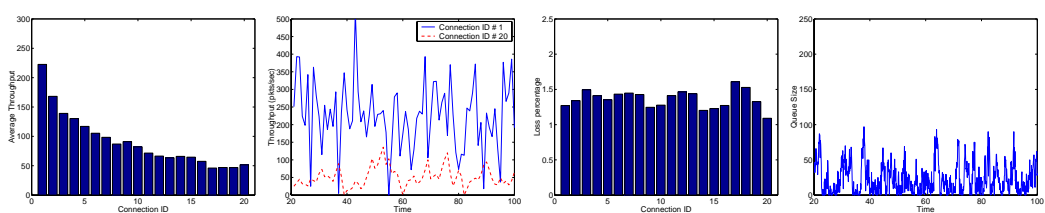
(a) DropTail [Fairness Index = 0.69]



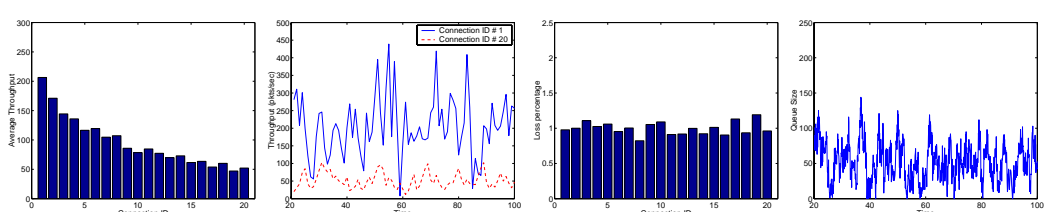
(b) RED [Fairness Index = 0.84]



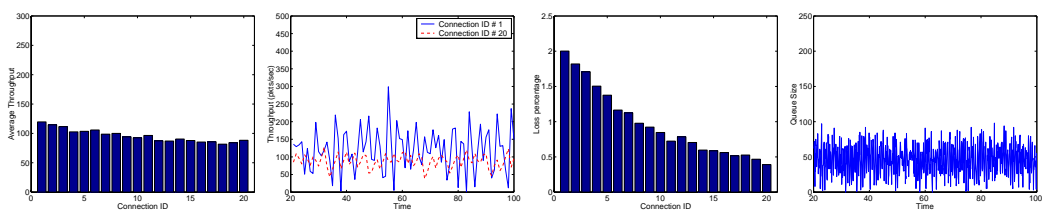
(c) FRED [Fairness Index = 0.91]



(d) REM [Fairness Index = 0.80]



(e) PI [Fairness Index = 0.84]



(e) XQM PI-TB [Fairness Index = 0.98]

Fig. 9. Comparative performance of various AQM approaches, showing long-term throughput for all flows (left), instantaneous throughput for two flows; the flow with the shortest RTT and the flow with the longest RTT, average loss rates seen by all flows, and instantaneous buffer size (rightmost).

one with the longest RTT (ID 20). Plots in the third column represent the average losses seen by each flow over the interval [20-100]. Finally, plots in the last (rightmost) column represent the instantaneous queue size. In all plots, we ignore the first 20 seconds (for simulation warm-up purposes) and we calculate all metrics starting from time 20. It suffices to say that all techniques were efficient and that all of them maintained the queue at the target level (except for DropTail). However, RED, FRED, PI and REM show bias against connections with long RTT as evident by the leftmost column in Figure 9. XQM, on the other hand, was able to improve the fairness index significantly. Table I lists the fairness index for each choice of queue management. One can see that XQM provides the highest fairness index, while remaining efficient.

AQM Scheme	Fairness Index
DropTail	0.69
RED	0.84
FRED	0.91
REM	0.80
PI	0.84
XQM ON-OFF	0.99
XQM PI-T	0.99
XQM PI-TB	0.98

TABLE I

Fairness Index for various AQMs (over 20 flows with different RTTs).

### C. Practical XQM Design Considerations and Implications

While maintaining per long-lived flow information could be feasible in some scenarios, in general, we need more practical (i.e., scalable) implementation techniques. One technique that fits well in the XQM framework is aggregation. Flows can be aggregated into classes of similar characteristics based on their round-trip time, for example. XQM, in effect, will impose a class-based quiescent loss rate rather than an individual quiescent loss rate for each connection. Such loss rate can be calculated using the average round-trip time over all connections in the class. This, in turn, will introduce some errors in quiescent loss rate calculations, due to error in estimating the round-trip time, or due to miss-classification of flows into RTT-equivalent classes. In [23], the authors developed a *running* RTT estimation technique that computes an RTT estimate every window of packets. They found out that 90% of their estimates had less than 10% error. This was compared to the triple handshake technique (that uses only one RTT estimate during the beginning of the connection), which had 90% of its estimates with relative error that is less than 30%.

To assess the susceptibility of XQM to the inaccuracies in classification and/or in RTT measurement for each class, we repeated the experiment above, where we have 20 connections of different RTTs ranging from 24 msec to 176 msec, *but now introducing errors in the RTT estimates*. Figure 10 shows the sensitivity of XQM PI-TB to error in RTT estimation (and hence error in calculating the quiescent loss rate). We allow for relative errors that are up to 50%. One can see that despite such inaccuracies, the fairness index is still high (above 0.95). In all these experiments, the target buffer was matched, so efficiency was not compromised.

Another possible extension for XQM functionality is managing UDP connections. This can be done in a similar way as discussed except that we remove the upper bound on the loss rate imposed. I.e. we don't calculate a quiescent loss rate, but we still measure the achievable throughput. This in turn,

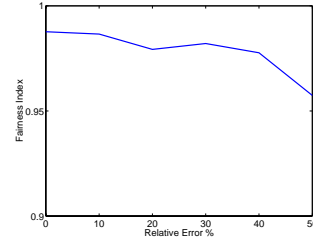


Fig. 10. Impact of misclassification of flows and/or error in estimating RTT “in the middle” on XQM’s fairness index.

will give each UDP flow its fair (weighted) share of the link capacity.

## VI. RELATED WORK

The work we present in this paper relates to a fairly large body of networking literature, targeting the simple goal of improving efficiency and fairness of transmission control loops. We exemplify the various flavors of this body of work below.

**Control-Theoretic Modeling and Analysis:** Marshaling techniques from control and optimization theory has been a fruitful direction as evidenced by a number of results, exemplified by the works in [7], [20], [21], [31], [25], [26], [17], [29], [27]. In that respect, we single out the works in [30], [20], which investigated the stability regions for TCP over RED using a dynamic fluid model for TCP. Kelly *et al.* [25] model TCP/AQM as an optimization problem, where the maximization of the aggregate resource utility is sought. These techniques, however, did not explicitly model exogenous losses. Rather, they focused mostly on congestion control.

**Active Queue Management Schemes:** The design of most AQM schemes (e.g., [15], [21], [27], [16]) have focused on the management of network congestion, with queue/buffer stabilization as the primary goal. Other schemes—notably FRED [28]—took a more active approach to protect flows that are particularly vulnerable (if additional losses are imposed) due to excessive shrinkage in buffer occupancy. As a byproduct of this special protection of vulnerable flows, FRED protects flows that are subject to excessive exogenous losses from further damage as they go through it. However, it is important to note that FRED’s protection of such flows is triggered by inadequate throughput (as opposed to an explicit accounting and management of exogenous losses) to protect them from excessively poor performance (e.g., due to the incidence of timeouts). On the other hand, schemes like fair queueing [10] and GPS [1], with per-flow state can easily provide *local* fairness, while our scheme is working towards *global* fairness. Clearly, the presence of exogenous losses negatively impacts the performance of all these AQMs, since they are unaware of (and not equipped to counteract the effects of) such losses.

**Explicit Treatment of Exogenous Losses:** Dealing with exogenous losses explicitly was addressed in projects that considered the impact of wireless communication (and wireless drops in particular) on TCP. A number of studies proposed breaking the transmission control loop into two segments [3], thus “hiding” the exogenous, wireless losses from the “wired” segment of the connection (not to mention “breaking” the end-to-end semantics of TCP). Other schemes (e.g. Snoop [4]) attempt to hide all wireless losses using local retransmission

at the wireless access point. Another set of studies opted to “hide” exogenous losses by assigning the task of dealing with such losses to end hosts, whereby the sender is empowered with diagnostic functionality that enables it to infer the reason for a packet loss and to react accordingly. Examples of this line of work are given in [6], [35], [13], [5]. More recently, and in order to avoid the severe implications of “overacting” to non-congestion-induced (e.g., exogenous) losses in high-speed, long-latency networks, the work in [8] suggests transmission control rules that use additional predictors (e.g., queuing delays) to moderate the reaction of senders to such losses. For both of these approaches (dealing with exogenous losses in the middle or at end points), exogenous losses are regarded as noise that must be completely eradicated (or hidden) from senders. None of these techniques advocate that some level of exogenous losses is harmless—let alone beneficial to boosting fairness and stability. And, clearly, none of these techniques leverages exogenous losses in the communication of the feedback signal from the bottleneck link.

## VII. CONCLUSION

In this paper, we captured the effect of exogenous packet losses by extending a dynamic fluid model for TCP. As one would expect, we showed that high levels of exogenous losses lead to inefficiencies. Surprisingly though, low levels of exogenous losses that don’t force TCP below its fair share, improve fairness between flows. We argue that exogenous loss awareness should be taken into account in the design of queue management algorithms that aim to achieve global fairness. Indeed, we showed that the road to global fairness requires more than what is currently proposed in the AQM design literature. In particular, it requires accounting for exogenous losses and, accordingly, invoking the right control rules on the right time scale. We proposed an eXogenous-loss aware Queue Management (XQM) that promotes fairness without compromising efficiency. In contrast to traditional AQM designs, XQM uses exogenous losses as *carriers* of its own feedback signal, hiding such losses *only* when they reach levels that jeopardize global fairness, and *only* to the extent necessary to avoid such unfairness. We are currently developing robust algorithms for XQM implementations, which leverage recent and on-going work of ours on measurement and control “in-the-middle”, including novel architectures within the ITM [32], APIs [11], and services.

## REFERENCES

- [1] A.Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single node case. *Trans on Networking*, June 1993.
- [2] S. Athuraliya, S. Low, V. Li, and Q. Yin. REM: Active Queue Management. *IEEE Network*, 15(3):48–53, 2001.
- [3] Ajay Bakre and B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *Proceedings of the 15th ICDCS*, 1995.
- [4] Hari Balakrishnan, S. Seshan, and R. Kartz. Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks. *ACM Wireless Networks*, 1(4), December 1995.
- [5] D. Barman and I. Matta. Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks. In *IEEE ICNP’2002*, Paris, France, November 2002.
- [6] Saad Biaz and Nitin H. Vaidya. Distinguishing Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver. In *IEEE ASSET 1999*, Richardson, TX, March 1999.
- [7] T. Bu and D. Towsley. Fixed Point Approximations for TCP Behavior in an AQM Network. In *ACM SIGMETRICS*, Boston, MA, June 2001.

- [8] David X. Wei Cheng Jin and Steven H. Low. Internet Draft: FAST TCP for high-speed long-distance networks. *draft-jwl-tcp-fast-01.txt*, 2003.
- [9] D.M. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.
- [10] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. *Internetworking: Research and Experience*, 1990.
- [11] G. Diamant, L. Veyster, I. Matta, A. Bestavros, M. Guirguis, L. Guo, Y. Zhang, and S. Chen. itmBench: Generalized API for Internet Traffic Managers. *BU-TR 2003-032*, 2003.
- [12] E. Amir et al. UCB/LBNL/VINT Network Simulator - ns (version 2).
- [13] Tae eun Kim, Songwu Lu, and Vaduvur Bharghavan. “Improving Congestion Control Performance Through Loss Differentiation”. In *ICCCN 1999*, Boston, MA, October 1999.
- [14] S. Floyd. Internet Draft: HighSpeed TCP for Large Congestion Windows. *draft-ietf-tsvwg-highspeed-01.txt*, 2003.
- [15] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *Trans. on Networking*, August 1993.
- [16] Richard J. Gibbens and Frank P. Kelly. Distributed Connection Acceptance Control for a Connectionless Network. In *Proceedings of 16th International Teletraffic Congress*, pages 941–952, June 1999.
- [17] R.J. Gibbens and F.P. Kelly. Resource Pricing and the Evolution of Congestion Control. *Automatica*, 35:1969–1985, 1999.
- [18] The PSC Networking group. The TCP-Friendly Website. [http://www.psc.edu/networking/tcp\\_friendly.html](http://www.psc.edu/networking/tcp_friendly.html).
- [19] M. Guirguis, A. Bestavros, and I. Matta. Exogenous-Loss Awareness in Queue Management: Toward Global Fairness. *BU Tech Report*, 2003.
- [20] C.V. Hollot, V. Misra, D. Towsley, and W. Gong. A Control Theoretic Analysis of RED. In *Proceedings of IEEE INFOCOM*, April 2001.
- [21] C.V. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proceedings of IEEE INFOCOM 2001*, Anchorage, AL, April 2001.
- [22] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM’88*, Stanford, CA, August 1988.
- [23] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone. In *IEEE INFOCOM 2003*, San Francisco, CA, April 2003.
- [24] Dina Katabi, Mark Handley, and Charles Rohrs. Internet Congestion Control for High Bandwidth-Delay Product Networks. In *Proceedings of ACM SIGCOMM 2002*, San Diego, CA, August 2002.
- [25] F.P. Kelly. Mathematical Modelling of the Internet. *Mathematics Unlimited - 2001 and Beyond*, pages 685–702, 2001.
- [26] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of Operations Research Society*, 1998.
- [27] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *Proceedings of ACM SIGCOMM 2001*, pages 123–134, August 2001.
- [28] D. Lin and R. Morris. Dynamics of Random Early Detection. In *Proceedings of ACM SIGCOMM’97*, Cannes, France, September 1997.
- [29] S. Low and D. Lapsley. Optimization Flow Control, I: Basic Algorithm and Convergence. *IEEE/ACM Transactions on Networking*, 1999.
- [30] S. Low, F. Paganini, J. Wang, S. Adlakha, and J. Doyle. Dynamics of TCP/RED and a Scalable Control. In *Proceedings of IEEE INFOCOM 2002*, New York, NY, June 2002.
- [31] S. Low, L. Peterson, and L. Wang. Understanding TCP Vegas: A Duality Model. In *ACM SIGMETRICS 2001*, Boston, MA, June 2001.
- [32] Ibrahim Matta and Azer Bestavros. QoS Controllers for the Internet. In *Proceedings of the NSF Workshop on Inf. Tech.*, Mar 2000.
- [33] N. F. Maxemchuck. Dispersy Routing in High-Speed Networks., *Computer Networks and ISDN Systems*, 25:645–661, 1993.
- [34] Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM 1999*, New York, USA, March 1999.
- [35] C. Parsa and J.J. Garcia-Luna-Aceves. “Differentiating Congestion vs. Random Loss: A Method for Improving TCP Performance over Wireless Links”. In *WCNC 2000:IEEE Wireless Communications and Networking Conference*, pages 23–28, Chicago, IL, September 2000.
- [36] J. Postel. RFC-768 User Datagram Protocol. *Request For Comments, Internet Network Information Center*, 1980.
- [37] R. Puri, K. Lee, K. Ramchandran, and V. Bharghavan. An Integrated Source Transcoding and Congestion Control Paradigm for Video Streaming in the Internet. *IEEE Transactions on Multimedia*, 3(1), 2001.
- [38] S. Shenker. A Theoretical Analysis of Feedback Flow Control. In *Proceedings ACM SIGCOMM’90*, Philadelphia, PA, September 1990.

## APPENDIX

In this appendix, we show the effect of exogenous losses on TCP behavior, specifically, that TCP's behavior follows more than one transfer function, in accordance with the level of exogenous losses. For the sake of simplicity and ease of exposition, we derive our control-theoretic results for the case of one TCP connection ( $m = 1$ ) traversing a RED bottleneck router.

A standard control-theoretic approach to studying a non-linear dynamical system is to linearize it around an operating point. This is done by expanding the equations using Taylor series and ignoring the high order terms. After linearization, the variables in the equations describe *perturbations* around the operating point that we have chosen. The question then becomes whether the system converges to the chosen operating point, i.e. whether perturbations in the system outputs eventually vanish after the system inputs were subjected to small perturbations. Clearly, based on our choice of the operating point, we would obtain a different set of (linear) equations.

In [30], Low *et al.* assumed that the system always operates in the region of  $B_{min} < v(t) < B_{max}$ . In general, this is not always the case because of the oscillatory behavior of TCP; TCP would keep decreasing its window until there are no losses, which suggests that  $v(t)$  could drop below  $B_{min}$ . Then, TCP would keep increasing its window until there are losses, which suggests that the system may switch between the two regions,  $B_{min} < v(t) < B_{max}$  and  $v(t) \leq B_{min}$ . We are particularly interested in this oscillatory regime around  $B_{min}$  since the presence of exogenous losses tend to drive  $v(t)$  below  $B_{min}$ . Equation (9) contains the time-varying delay lag  $r(t)$ , which makes linearization hard as pointed out in [30]. As an approximation, we replace  $r(t)$  by its steady-state value. Substituting equation (1) in (9) and linearizing yields (recall that after linearization, variables represent perturbations):

$$\dot{w}(t) = \frac{\hat{q}\hat{w}}{\hat{r}}w(t) - \frac{1}{\hat{r}\hat{q}}q(t) \quad (21)$$

where  $\hat{r}$ ,  $\hat{q}$  and  $\hat{w}$  are the steady-state round trip time, total marking/loss probability and window size, respectively.

Similarly, substituting equation (1) in (3) and linearizing yields:

$$\dot{b}(t) = \frac{C}{\hat{w}}w(t - D_{sb}) - \frac{C}{\hat{w}}b(t - D_{sb}) \quad (22)$$

If we choose the operating point for RED's average buffer size  $v(t)$  to lie between  $B_{min}$  and  $B_{max}$ , from (6) and (7), we can write the following RED's linear equations:

$$\dot{v}(t) = -\beta C(v(t) - b(t)) \quad (23)$$

$$p_c(t) = \sigma v(t) \quad (24)$$

On the other hand, by choosing the operating point to be  $v(t) \leq B_{min}$ , from (6) we get:

$$p_c(t) = 0 \quad (25)$$

Equation (25) says that when  $v(t) \leq B_{min}$ , there is no feedback given from the network back to the sender.

*a) Closed Loop System::* Assuming the system operates in the region  $B_{min} < v(t) < B_{max}$  and taking the Laplace Transforms for the above linear model (21)-(24), we get:

$$w(s) = \frac{D_2 e^{-D_{bs}s}}{s + D_1} (p_c(s) + p_e(s)) \quad (26)$$

$$b(s) = \frac{K_1 e^{-D_{sb}s}}{s + K_1 e^{-D_{sb}s}} w(s) \quad (27)$$

$$p_c(s) = \frac{\beta C \sigma}{s + \beta C} b(s) \quad (28)$$

where  $D_1 = \frac{\hat{q}\hat{w}}{\hat{r}}$ ,  $D_2 = \frac{-1}{\hat{q}\hat{r}}$ , and  $K_1 = \frac{C}{\hat{w}}$ .

Hence the closed loop transfer function in the Laplace domain is given by:

$$\frac{w(s)}{p_e(s)} = \frac{D_2(s + \beta)(s + K_1 e^{-D_{sb}s})e^{-D_{bs}s}}{(s + D_1)(s + \beta)(s + K_1 e^{-D_{sb}s}) - D_2 K_1 \beta \sigma e^{-ds}} \quad (29)$$

*b) Open Loop System::* Observe that if the system operates instead in the region  $v(t) \leq B_{min}$ , then equation (25) replaces (24) and we have an open loop system whose transfer function is given by:

$$\frac{w(s)}{p_e(s)} = \frac{D_2 e^{-D_{bs}s}}{s + D_1} \quad (30)$$

Figure 11 illustrates the open loop and closed loop transfer functions of TCP over RED in the presence of exogenous losses  $p_e$ . One can easily see that if  $p_e$  is high enough to cause the TCP sender to shrink its window  $w$  to the point that the average RED queue size is below  $B_{min}$ , then  $p_c$  equals zero and the closed loop is broken, i.e. TCP operates as an open loop control system. Even with no exogenous losses, the system may oscillate between two different regimes, e.g., the average queue length may oscillate around  $B_{min}$ . This switching between totally different behaviors, especially under time-varying exogenous losses, prevents us from applying traditional control-theoretic transient analysis techniques, thus we resort to the numerical solution of the non-linear equations.

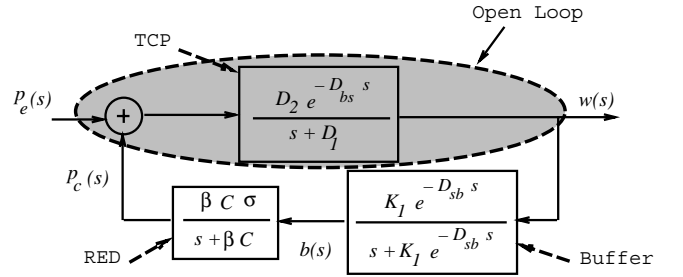


Fig. 11. Linearized block diagram

Figure 12 shows the effect of exogenous losses on the DropTail buffer trajectory. Here we set the propagation delay of all 20 connections to 100 msec. Buffer trajectories are represented by the buffer size on the x-axis, while the y-axis represents the change in the buffer size. This kind of plots are used in control theory to visualize non-linear behaviors. Figure 12(left) shows how the buffer oscillates in a limit cycle when there are no exogenous losses. Introducing a quiescent value of 1% exogenous losses leads to convergence of the buffer trajectory to a non-zero value, where the system is efficient and stable.

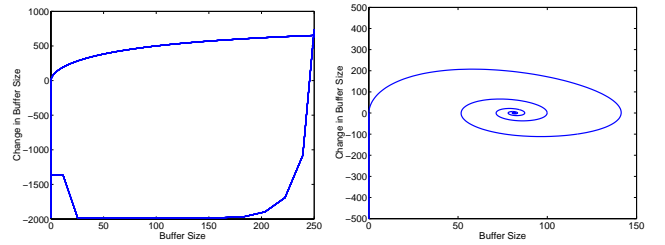


Fig. 12. Buffer trajectories with no exogenous losses (left) and with 1% exogenous losses (right).