Internet Congestion Control *

Steven H. Low
Computer Science and Electrical Engineering
California Institute of Technology
slow@caltech.edu

Fernando Paganini
Electrical Engineering
University of California, Los Angeles
paganini@ee.ucla.edu

John C. Doyle
Electrical Engineering and Control & Dynamical Systems
California Institute of Technology
dolye@cds.caltech.edu

October 20, 2001

Abstract

This article reviews the current TCP congestion control protocols and overviews recent advances that have brought analytical tools to this problem. We describe an optimization-based framework that provides an interpretation of various flow control mechanisms, in particular, the *utility* being optimized by the protocol's equilibrium structure. We also look at the dynamics of TCP and employ linear models to exhibit stability limitations in the predominant TCP versions, despite certain built-in compensations for delay. Finally, we present a new protocol that overcomes these limitations and provides stability in a way that is scalable to arbitrary networks, link capacities, and delays.

1 Introduction

Congestion control mechanisms in today's Internet already represent one of the largest deployed artificial feedback systems; as the Internet continues to expand in size, diversity, and reach, playing an ever-increasing role in the integration of other networks (transportation, finance,...), having a solid understanding of how this fundamental resource is controlled becomes ever more crucial. Given the scale and complexity of the network, however, and the heuristic, intricate nature of

*To appear in IEEE Control Systems Magazine, February 2002

many deployed control mechanisms (which we summarize in the next section), until recently this problem appeared to be well beyond the reach of analytical modeling and feedback control theory. Theoretical research on this topic (e.g., [1, 2, 3, 4, 5]) has dealt with simple scenarios (e.g., single bottleneck, per-flow queueing); see also recent survey in [6, 7]. Meanwhile the Internet community has turned to small-scale simulations to validate designs. All of these leave huge gaps in our understanding of real network behavior.

In the last few years, however, large strides have been taken in bringing analytical models into Internet congestion control (see e.g., [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22] and references therein). Key to these advances has been to explicitly model the *congestion measure* that communicates back to data sources the information on congestion in network resources being used; more precisely, it is assumed that each network link measures its congestion by a scalar variable (termed *price*), and that sources have access to the aggregate price of links in their path. These assumptions are implicitly present in many variants of today's TCP protocols; this framework exhibits the price signal being used in these protocols (e.g., loss probability, queueing delay). Also, it is the natural setting for exploring alternative protocols based on more explicit price signaling (e.g., bit marking).

Two types of studies are of fundamental interest. On the one hand, it is important to characterize the equilibrium conditions that can be obtained from a given congestion control protocol from the point of view of fairness, efficiency in use of resources, dependence on network parameters, etc. In this regard, the above-mentioned prices can be interpreted in economic terms (e.g., [9]) and the congestion control system as seeking the global optimum of a certain aggregate *utility function*, subject to network capacity constraints. By describing the utility implicit in existing TCP protocols [11, 12], equilibrium properties are inferred, some of which corroborate empirically observed features. This methodology will be reviewed in the section "Equilibrium Structure and Utility Optimization". A second line of inquiry concerns the *dynamics* of congestion control protocols, directly in the domain of control theory. In particular, we are interested in the *stability* of the postulated equilibria, especially in the presence of feedback delay, and in performance metrics such as speed of convergence, capacity tracking, etc. In fact, by incorporating explicit measures of congestion, recent analysis [21, 22] has shown that the predominant TCP implementation (called Reno) and its variants are prone to instabilities when combined with network delays and, more surprisingly, with increases in network capacity. We will show similar studies in the section "Dynamics and Stability".

This raises the question of whether more efficient and stable protocols could be developed with the help of this analytic framework. We note that the constraint of decentralization makes it impossible for a controller to be synthesized by, e.g., optimal control; still, one can try to postulate a plausible control law and support it with proofs of stability and performance. In this regard,

global stability results that apply to arbitrary networks have been given for new price-based flow controllers [8, 10, 15, 14], in the absence of delay; further, delay can be studied in terms of conditions on control gains to retain stability (e.g., [8, 10, 23]). Now as noted in, e.g., [24, 25], window-based protocols contain an automatic compensation for delay ("self-clocking"); this has led to recent work seeking protocols that would remain stable for sufficiently small delays [26, 27, 28]. In this vein, we describe in the final section "A Scalable Control" some of our recent work in finding a protocol that can be implemented in a decentralized way by sources and routers, and that provides linear stability for *arbitrary* delays, capacities, and routes.

Although TCP Reno has performed remarkably above expectations and is widely deployed, we emphasize a key limitation of this congestion control mechanism: by using packet loss as a congestion measure, high utilization can be achieved only with full queues, i.e., when network operates at the boundary of congestion. This seems particularly ill-suited to handle the types of traffic that have been observed in recent studies. Indeed, Internet traffic, 90% of which is TCP-based (see measurements at www.caida.org), exhibits burstiness at many time-scales, which is due to the heavy-tailed nature of file sizes [29, 30, 31, 32, 33]. In simple terms this means that most TCP connections are "mice" (short, but requiring low latency), but a few long TCP connections ("elephants," which can tolerate latency) generate most of the traffic. By controlling the network around a state with full queues, the elephants subject the mice to unnecessary loss and queueing delays. This problem can be avoided by decoupling loss from price signaling. Another limitation of using loss to measure congestion is the degradation of performance in the cases where losses are often due to other effects (e.g., wireless links). These considerations are motivating a new look at congestion control protocols; our aim in this article is to argue that a more sound analytical perspective, now available, should be brought to bear on this investigation.

2 Current TCP Protocols

TCP uses "window" flow control, where a destination sends acknowledgments for packets that are correctly received. A source keeps a variable called window size that determines the maximum number of outstanding packets that have been transmitted but not yet acknowledged. When the window size is exhausted, the source must wait for an acknowledgment before sending a new packet. Two features are important. The first is the "self-clocking" feature that automatically slows down the source when a network becomes congested and acknowledgments are delayed. The second is that the window size controls the source rate: roughly one window of packets is sent every round-trip time. The first feature was the only congestion control mechanism in the Internet before Van Jacobson's proposal in 1988 [24]. Jacobson's idea is to *dynamically* adapt window size to network congestion. In this section, we will review how TCP infers congestion and adjusts window size.

TCP also provides other end–to–end services such as error recovery and round-trip time estimation, but we will limit our attention to the congestion control aspect.

2.1 TCP Tahoe and Reno

The predominate TCP implementations are called Tahoe and Reno. The basic idea of these protocols is for a source to gently probe the network for spare capacity by linearly increasing its rate and exponentially reducing its rate when congestion is detected. Congestion is detected when the source detects a packet loss.

A connection starts cautiously with a small window size of one packet (up to four packets have recently been proposed) and the source increments its window by one every time it receives an acknowledgment. This doubles the window every round-trip time and is called `slow-start`. When the window reaches a threshold, the source enters the `congestion avoidance` phase, where it increases its window by the reciprocal of the current window size every time it receives an acknowledgment. This increases the window by one in each round-trip time, and is referred to as additive increase. The threshold that determines the transition from `slow-start` to `congestion avoidance` is meant to indicate the available capacity in the network and is adjusted each time a loss is detected. On detecting a loss, the source sets the slow-start threshold to half of the current window size, retransmits the lost packet, and re-enters `slow-start` by resetting its window to one.

This algorithm was proposed in [24] and implemented in the Tahoe version of TCP. Two refinements, called `fast recovery`, were subsequently implemented in TCP Reno to recover from loss more efficiently. Call the time from detecting a loss (through duplicate acknowledgments) to receiving the acknowledgment for the retransmitted packet the `fast retransmit/fast recover (fr/fr)` phase. In TCP Tahoe, the window size is frozen in the `fr/fr` phase. This means that a new packet can be transmitted only a round-trip time later. Moreover, the "pipe" from the source to the destination is cleared when the retransmitted packet reaches the receiver, and some of the routers in the path become idle during this period, resulting in loss of efficiency. The first refinement allows a Reno source to temporarily increment its window by one on receiving each duplicate acknowledgment while it is in the `fr/fr` phase. The rationale is that each duplicate acknowledgment signals that a packet has left the network. When the window becomes larger than the number of outstanding packets, a *new* packet can be transmitted in the `fr/fr` phase while it is waiting for a (nonduplicate) acknowledgment for the retransmitted packet. The second refinement essentially sets the window size at the end of the `fr/fr` phase to half of the window size when `fr/fr` starts and enters `congestion avoidance` directly. Hence, `slow-start` is entered only rarely in TCP Reno when the connection first starts and when a loss is detected by timeout rather than duplicate acknowledgments.

2.2 TCP Vegas

TCP Vegas [34] improves upon TCP Reno through three main techniques. The first is a new retransmission mechanism where timeout is checked on receiving the first duplicate acknowledgment, rather than waiting for the third duplicate acknowledgment (as Reno would), and results in a more timely detection of loss. The second technique is a more prudent way to grow the window size during the initial use of `slow-start` when a connection starts up and it results in fewer losses.

The third technique is a new congestion avoidance mechanism that corrects the oscillatory behavior of Reno. The idea is to have a source estimate the number of its own packets buffered in the path and try to keep this number between α (typically 1) and β (typically 3) by adjusting its window size. The window size is increased or decreased linearly in the next round-trip time according to whether the current estimate is less than α or greater than β. Otherwise the window size is unchanged. The rationale behind this is to maintain a small number of packets in the pipe to take advantage of extra capacity when it becomes available. Another interpretation of the congestion avoidance algorithm of Vegas is given in [12], in which a Vegas source periodically measures the round-trip *queueing* delay and sets its rate to be proportional to the ratio of its round trip propagation delay to queueing delay, the proportionality constant being between α and β. Hence, the more congested its path is, the higher the queueing delay and the lower the rate. The Vegas source obtains queueing delay by monitoring its round-trip time (the time between sending a packet and receiving its acknowledgment) and subtracting from it the round-trip propagation delay.

2.3 FIFO, DropTail, and RED

A Vegas source adjusts its rate based on observed queueing delay; in other words, it uses queueing delay as a measure of congestion. This information is updated by the FIFO (first-in-first-out) buffer process and fed back implicitly to sources through round-trip time measurement. A Reno source uses loss as a measure of congestion. This information is typically generated and fed back to sources through DropTail, a queueing discipline that drops an arrival to a full buffer. RED (Random Early Detection) [35] is an alternative way to generate the congestion measure (loss) to Reno sources. Instead of dropping only at a full buffer, RED maintains an exponentially weighted queue length and drops packets with a probability that increases with the average queue length. When the average queue length is less than a minimum threshold, no packets are dropped. When it exceeds a maximum threshold, all packets are dropped. When it is in between, a packet is dropped with a probability that is a piecewise linear and increasing function of the average queue length. This type of strategy is called *active queue management* (AQM).

3 Analytical Models

In this section, we describe analytical models that were developed in the last few years. A large body of literature exists on congestion control, but we will focus narrowly on these recent models.

A network is modeled as a set of L links with finite capacities $c = (c_l, l \in L)$ in packets per second. They are shared by a set of N sources. Each source i uses a set $L_i \subseteq L$ of links. The sets L_i define an $L \times N$ routing matrix

$$R_{li} = \begin{cases} 1 & \text{if } l \in L_i \\ 0 & \text{otherwise} \end{cases} .$$

A first consideration is that we will use deterministic *flow* models to describe transmission rates, in contrast to much of classical queueing theory which relies on stochastic (e.g. Poisson) models for traffic. While randomness in network arrivals is natural when modeling entire connections [36], it is less suitable at the packet level, where transmission times for congestion controlled sources are determined predominantly by feedback, as described in the previous section. Furthermore, the distributions to be used in such random models have recently come into question [29, 30, 31, 32, 33], and in any event, there are few tractable results on queueing theory in the presence of feedback (but see [37]). For these reasons, we will study feedback at a higher level of aggregation than packets, modeling rates as flow quantities. Each source i has an associated transmission rate $x_i(t)$; the set of transmission rates determines the aggregate flow $y_l(t)$ at each link, by the equation

$$y_l(t) = \sum_i R_{li} x_i(t - \tau_{li}^f), \tag{1}$$

where τ_{li}^f denote the forward transmission delays from sources to links.

The next step is to model the feedback mechanism which communicates to sources the congestion information about the network. The key idea in the line of work we are discussing is to associate with each link l a *congestion measure* $p_l(t)$, which is a positive real-valued quantity. Due to the economic interpretations to be discussed in the next section, we will call this variable a "price" associated with using link l. The fundamental assumption we make is that sources have access to the *aggregate* price of all links in their route,

$$q_i(t) = \sum_l R_{li} p_l(t - \tau_{li}^b). \tag{2}$$

Here again we allow for *backward* delays τ_{li}^b in the feedback path. As we will discuss below, this feedback model includes, to a good approximation, the mechanism present in existing protocols, with a different interpretation for price in different protocols (e.g. loss probability in TCP Reno, queueing delay in TCP Vegas).

The preceding equations can be represented in the Laplace domain in terms of the delayed forward and backward routing matrices:

$$[R_f(s)]_{li} = \begin{cases} e^{-\tau_{li}^f s} & \text{if } l \in L_i \\ 0 & \text{otherwise} \end{cases}, \qquad [R_b(s)]_{li} = \begin{cases} e^{-\tau_{li}^b s} & \text{if } l \in L_i \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Then we have, in vector form (^{T} denotes transpose):

$$y(s) = R_f(s)x(s) \tag{4}$$
$$q(s) = R_b(s)^T p(s). \tag{5}$$

To specify the congestion control system, it remains to define (i) how the sources adjust their rates based on their aggregate prices (the TCP algorithm), and (ii) how the links adjust their prices based on their aggregate rates (the AQM algorithm). At the source side, we can in general postulate a dynamic model of the form

$$\begin{aligned} \dot{z}_i &= F_i(z_i, q_i), \\ x_i &= G_i(z_i, q_i). \end{aligned} \tag{6}$$

where z_i would be a local state variable. We will, however, mostly encounter two special cases: the static case, where there is no z_i and we have $x_i(t) = G_i(q_i(t))$, and the first-order case with $z_i = x_i$.

Similarly, at the link level one can write a dynamic law

$$\begin{aligned} \dot{v}_l &= H_l(y_l, v_l), \\ \dot{p}_l &= K_l(y_l, v_l). \end{aligned} \tag{7}$$

The key restriction in the above control laws is that they must be *decentralized*, i.e. sources and links only have access to their local information. The overall structure of the congestion control system is now depicted in Figure 1, where the diagonal structure of the source and link matrices represents the decentralization requirement.

We will discuss TCP models within this general framework. In the next subsection, we will focus on equilibrium properties; dynamic issues are tackled in the following subsection "Dynamics and stability".

3.1 Equilibrium Structure and Utility Optimization

In this section we study the above feedback at equilibrium, i.e. assuming the rates and prices are at some fixed values x^*, y^*, p^*, q^*. We will see how an optimization language helps understand the properties of such equilibria.

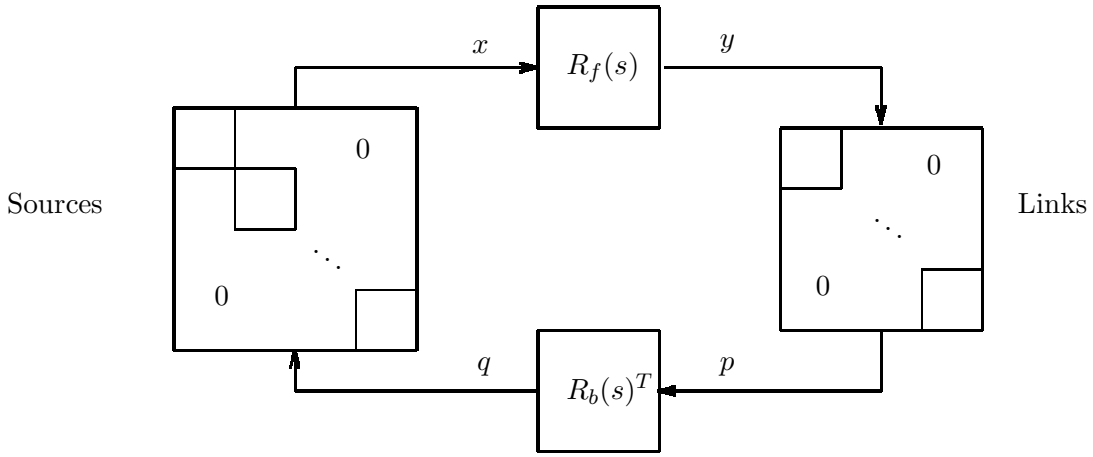


Figure 1: General congestion control structure.

The equilibrium relationships $y^* = Rx^*$, $q^* = R^T p^*$ follow immediately from (4)–(5). Here R is the static routing matrix; since we are discussing equilibrium we can set $s = 0$ in the model (3) (equivalently, setting all delays to zero).

The first basic assumption we make is that the equilibrium rates satisfy

$$x_i^* = f_i(q_i^*),$$

where $f_i(\cdot)$ is a positive, strictly monotone decreasing function. This function can be found by finding the equilibrium point in (6); in the static case it is just given by the source static law. Monotonicity is a natural assumption for all protocols: if q_i represents congestion in the source's path, the equilibrium source rate should be a monotonically decreasing function of it.

We now see that this assumption alone allows us to introduce an optimization interpretation for the equilibrium, by introducing a source *utility* function. Namely, consider the inverse $f_i^{-1}(x_i)$ of the above function, and let $U_i(x_i)$ be its integral; i.e. $U_i'(x_i) = f_i^{-1}(x_i)$. By assumption, $U_i(x_i)$ has a positive, decreasing derivative and is therefore itself monotone increasing, and strictly concave. Now by construction, the equilibrium rate will solve

$$\max_{x_i} \; U_i(x_i) - x_i q_i^*. \tag{8}$$

The above equation can be interpreted in economic terms: if $U_i(x_i)$ is the utility the source attains for transmitting at rate x_i, and q_i^* is the price per unit flow it is hypothetically charged, the above represents a maximization of individual source's profit. We emphasize that this interpretation is available with minimal assumptions about the protocol; given a model of the source (TCP) control, one can derive from it the utility function associated with the protocol, as we will see below.

The role of prices is to coordinate the actions of individual sources so as to align individual

optimality with social optimality, i.e., to ensure that the solutions of (8) also solve the problem

$$\max_{x \geq 0} \quad \sum_i U_i(x_i) \tag{9}$$

$$\text{subject to} \quad Rx \ \leq \ c, \tag{10}$$

in other words maximize aggregate utility across all sources, subject to the link capacity constraints. This problem, formulated in [38], is a convex program for which a unique optimal rate vector exists. The challenge is to solve it in a distributed manner over a large network.

A natural way to introduce prices in regard to the above optimization is the duality approach introduced in [10] (see also [39] for related ideas, and [20] for multicast extensions). Here prices come in as Lagrange multipliers for the problem (9-10). Specifically, consider the Lagrangian

$$L(x,p) = \sum_i U_i(x_i) - \sum_l p_l(y_l - c_l) = \sum_i U_i(x_i) - q_i x_i + \sum_l p_l c_l.$$

The dual problem is

$$\min_{p \geq 0} \quad \sum_i B_i(q_i) + \sum_l p_l c_l. \tag{11}$$

where

$$B_i(q_i) \ = \ \max_{x_i \geq 0} \ U_i(x_i) - x_i q_i. \tag{12}$$

Convex duality implies that at the optimum p^*'s (which need not be unique), the corresponding x^* maximizing individual optimality (12) is exactly the unique solution to the primal problem (9)–(10). Note that (12) is identical to (8); therefore provided the equilibrium prices p^* can be made to align with the Lagrange multipliers, the individual optima, computed in a decentralized fashion by sources, will align with the global optima of (9)–(10).

The simplest link algorithm that guarantees these equilibrium prices are indeed Lagrange multipliers, as shown in [10], is based on applying the gradient projection algorithm to the dual problem (11):

$$\dot{p}_l \ = \ \begin{cases} \gamma \left(y_l(t) - c_l\right) & \text{if } p_l(t) > 0 \\ \gamma \ [y_l(t) - c_l]^+ & \text{if } p_l(t) = 0 \end{cases} \tag{13}$$

where $[z]^+ = \max\{z, 0\}$. The fact that the gradient of the Lagrangian only depends on aggregate rates y_l is key to the above decentralized implementation at the links. If the above equations are at equilibrium, we have $y_l^* \leq c_l$, with nonzero prices p_l^* corresponding to the active constraints. It follows that equilibrium prices are the Lagrange multipliers. This property is however not exclusive to this algorithm; rather, *any* pricing scheme that stabilizes queues, or queueing delays (e.g. RED or Vegas, see below), so that flow rates y_l^* are matched to capacities c_l, will allow for the same

interpretation. This is because matching rate drives the gradient of the dual problem (11)–(12) with respect to p to zero, solving the dual problem and implying that the resulting prices are Lagrange multipliers.

Another approach to price variables, proposed in [8] (and used also in [15]), is to treat them as a penalty or barrier function for the constraint (10). Here prices are assumed to be a static, increasing function of y_l, $p_l = h(y_l)$, which becomes large as y_l approaches c_l. It follows that the resulting equilibrium maximizes the global utility

$$\sum_i U_i(x_i) - \sum_l \int_0^{y_l} h_l(y)dy, \tag{14}$$

which can be seen as an approximation to the above problem (9)–(10).

To summarize the discussion so far: under very general assumptions, the equilibrium points of source protocols can be interpreted in terms of sources maximizing individual profit based on their own utility functions. Link algorithms generate prices to align, exactly or approximately, these "selfish" strategies with social welfare. Different protocols correspond to different utility functions U_i, and to different dynamic laws (6)–(7) that attempt, in a decentralized way, to reach the appropriate equilibrium. We now take a closer look at modeling Reno and Vegas in this context.

3.1.1 TCP Reno/RED

We focus *only* on the `congestion avoidance` phase of TCP Reno, in which an elephant typically spends most of its time. We take source rates as the primal variable x and link loss probabilities as prices p. In this section we assume the round-trip time τ_i of source i is constant, and that rate x_i is related to window w_i by

$$x_i(t) \quad = \quad \frac{w_i(t)}{\tau_i}. \tag{15}$$

In rigor, the window of outstanding packets at time t reflects the average rate in the interval $[t - \tau_i, t]$; the above approximation is valid since our models are not intended to provide accurate description at finer time-scales than the round-trip time.

We also make the key assumption that loss probabilities $p_l(t)$ are small so that the end-to-end probabilities $q_i(t)$ satisfy

$$q_i(t) \quad = \quad 1 - \prod_{l \in L_i} (1 - p_l(t)) \quad \simeq \quad \sum_{l \in L_i} p_l(t)$$

for all i for all t.

We now model the additive-increase-multiplicative-decrease (AIMD) algorithm of TCP Reno, in an *average* sense at time scales above the round-trip time. In particular our models do not attempt to predict a window *jump* of the sort observed under MD; rather, they should track the

mean evolution of the window as a function of ACKs and losses. We initially ignore feedback delays, since we are interested in equilibrium points.

At time t, $x_i(t)$ is the rate at which packets are sent and acknowledgments received. A fraction $(1 - q_i(t))$ of these acknowledgments are positive, each incrementing the window $w_i(t)$ by $1/w_i(t)$; hence the window $w_i(t)$ increases, on average, at the rate of $x_i(t)(1-q_i(t))/w_i(t)$. Similarly negative acknowledgments are returning at an average rate of $x_i(t)q_i(t)$, each halving the window, and hence the window $w_i(t)$ decreases at a rate of $x_i(t)q_i(t)w_i(t)/2$. Hence, since $x_i(t) = w_i(t)/\tau_i$, we have for Reno the average model

$$\dot{x}_i \quad = \quad \frac{1 - q_i(t)}{\tau_i^2} - \frac{1}{2} q_i(t) x_i^2(t). \tag{16}$$

We now consider the equilibrium of (16):

$$q_i^* \quad = \quad \frac{2}{2 + \tau_i^2 (x_i^*)^2}. \tag{17}$$

From it, we can obtain the utility function of TCP Reno, by identifying the above with Karush-Kuhn-Tucker condition $U_i'(x_i^*) = q_i^*$. This gives the utility function

$$U_i(x_i) \quad = \quad \frac{\sqrt{2}}{\tau_i} \tan^{-1}\left(\frac{\tau_i x_i}{\sqrt{2}}\right), \tag{18}$$

which seems to appear first in [9, 11]. Our description here is slightly different from that in [11] in that, here, loss probability is taken as the dual variable regardless of the link algorithms, which from this point of view affect the dynamics but not the utility function.

The relation (17) between equilibrium source rate and loss probability reduces to the well known relation (see e.g. [40, 41]):

$$x_i \quad = \quad \frac{a}{\tau_i \sqrt{q_i}}$$

when the probability q_i is small, or equivalently, when the window $\tau_i x_i$ is large compared with $\sqrt{2}$.[1] The value of the constant a, around 1, has been found empirically to depend on implementation details such as TCP variant (e.g., Reno vs. NewReno vs. SACK) and whether delayed acknowledgment is implemented. Equating $U_i'(x_i)$ with q_i, the utility function of TCP Reno becomes:

$$U_i(x_i) \quad = \quad -\frac{a^2}{\tau_i^2 x_i}.$$

This version is used in [15, 42].

We now turn to link algorithm, i.e. how loss probabilities $p_l(t)$ are generated as a function of link rate y_l. Both DropTail and RED produce losses as a function of the state of the queue, so the

[1] This corresponds to replacing $(1 - q_i(t))$ in (16) by 1, as done in [22].

question involves modeling of the queue dynamics as a function of the input rate y_l. In the context of flow models as we are discussing, it is natural to model queues as integrating the excess capacity; this option is discussed below, and in the case of RED one can then easily relate queues to loss (or marking) probability. Unfortunately, a deterministic model of DropTail which includes the queue dynamics does not appear to be tractable.

At the other extreme, many references model queues as being in steady-state, and postulate a static law $p_l = h_l(y_l)$ for loss probability as a function of traffic rate, e.g. [8, 15]. The question as to whether the steady-state queue assumption can be justified is still a subject of current research [43]; however dynamic simulation studies of Reno/RED, as in [22] or those described below (see Figure 2(b)), indicate that queue dynamics are indeed significant at the time-scale of interest.

For this reason we will consider the first option, and model queues as integrators, focusing on the RED queue management to obtain simple models of loss probabilities. Let $b_l(t)$ denote the instantaneous queue length at time t; its dynamics is then modeled by

$$\dot{b}_l \;=\; \begin{cases} (y_l(t) - c_l) & \text{if } b_l(t) > 0 \\ [y_l(t) - c_l]^+ & \text{if } b_l(t) = 0 \end{cases} . \tag{19}$$

RED averages the instantaneous queue by an exponentially weighted average; denoting $r_l(t)$ to be the averaged queue length, we can model it as a lowpass filter

$$\dot{r}_l \;=\; -\alpha_l c_l \; (r_l(t) - b_l(t)), \tag{20}$$

for some constant $0 < \alpha_l < 1$. Given the average queue length $r_l(t)$, the marking (or dropping) probability is given by a static function

$$p_l(t) = \;\; m_l(r_l(t)) := \begin{cases} 0 & r_l(t) \le \underline{b}_l \\ \rho_l r_l(t) - \rho_l \underline{b}_l & \underline{b}_l < r_l(t) < \overline{b}_l \\ \eta_l r_l(t) - (1 - 2\overline{p}_l) & \overline{b}_l \le r_l(t) < 2\overline{b}_l \\ 1 & r_l(t) \ge 2\overline{b}_l \end{cases} \tag{21}$$

where $\underline{b}_l, \overline{b}_l$, and $\overline{p}_l$ are RED parameters, and

$$\rho_l \;:=\; \frac{\overline{p}_l}{\overline{b}_l - \underline{b}_l} \qquad \text{and} \qquad \eta_l \;:=\; \frac{1 - \overline{p}_l}{\overline{b}_l}$$

Now equations (19), (20), and(21) fall into the general form (7), where the internal state v_l is comprised of b_l and r_l.

Assume now that these equations are in equilibrium. It is not difficult to see that in this case $y_l^* \le c_l$, and that the inequality is only strict when $p_l^* = 0$. These facts imply that the equilibrium p_l^* constitute a set of Lagrange multipliers for the dual problem (11)–(12). Thus we conclude

that if the Reno/RED reaches equilibrium, then the resulting source rates x_i^* will solve the primal problem (9)–(10) with the source utility functions given in (18). Moreover the loss probabilities p_l^* are Lagrange multipliers that solve the dual problem (11)–(12).

We remark again that the preceding analysis refers to the averaged model of TCP, as described in (16); we are not attempting to impose equilibrium on the detailed evolution of a window under AIMD, but rather on its mean evolution, as for instance would happen with the average of many identical sources (see simulations below). Still, even in this mean sense, we have not given any indication yet that TCP reaches equilibrium. Indeed in the next section we will find that it often does not, since the equilibrium is unstable and a limit cycle is observed; in particular average windows and queues can oscillate dramatically. Nevertheless, the above equilibrium analysis is useful in understanding the state that is *aimed for* by the current protocols, i.e. the resource allocation policy implicitly present in the current Internet. Furthermore, there is evidence that some of the insights derived from the equilibrium models do reflect empirical properties of the Internet. This suggests that the models might have value in describing the protocol's long-term behavior, even in an oscillatory regime. We now discuss some of these insights.

Delay and loss

The current protocol (Reno with DropTail) fills, rather than empties, bottleneck queues when the number of elephants becomes large, leading to a high loss rate and queueing delay. What is more intriguing is that increasing the buffer size does not reduce loss rate significantly, but only increases queueing delay. This delay and loss behavior is exactly opposite to the mice-elephant control strategy we aim for: to maximally utilize the network in a way that leaves network queues small so that delay sensitive mice can fly through the network with little queueing delay.

According to the duality model, loss probability under Reno is the Lagrange multiplier, and hence its equilibrium value is determined solely by the network topology and the number of sources, *independent of link algorithms and buffer size.* Increasing the buffer size with everything else unchanged does not change the equilibrium loss probability, and hence a larger backlog must be maintained to generate the same loss probability. This means that with DropTail, the buffer at a bottleneck link is always close to full regardless of buffer size. With RED, since loss probability is increasing in average queue length, the queue length must increase steadily as the number of sources grows.

Fairness

It is well known that TCP Reno discriminates against connections with large propagation delays. This is clear from (17), which implies that Reno equalizes windows for sources that experience the same loss probability, and hence their rates are inversely proportional to their round-trip times.

The equilibrium characterization (17) also exposes the "beat down" effect, where sources that go through more congested links, seeing larger q_i, receive less bandwidth. This effect is hidden in

single-link models and is often confused with delay-induced discrimination of TCP, as expressed in (17), in multi-link models. It has been observed in simulations [44] and has long been deemed unfair, but the duality model shows that it is unavoidable, and even desirable, feature of end-to-end congestion control. For each unit of increment in aggregate utility, a source with a longer path consumes more resources and hence should be beaten down. If this is undesirable, it can be remedied by weighting the utility function with delay.

3.1.2 TCP Vegas

The dynamic model and utility function U_i of TCP Vegas have been derived and validated in [12]. We briefly summarize here the results.

The utility function of TCP Vegas is

$$U_i(x_i) \quad = \quad \alpha_i d_i \log x_i$$

where α_i is a protocol parameter and d_i is the round-trip propagation delay of source i. In equilibrium, source i buffers $\alpha_i d_i$ packets in the routers in its path. The utility function implies that Vegas achieves proportional fairness in equilibrium.

The price variable in TCP Vegas is queueing delay, which evolves according to

$$\dot{p}_l \quad = \quad \frac{1}{c_l}\,(y_l(t) - c_l), \tag{22}$$

with an additional non-negativity constraint, exactly as in (13) with γ replaced by $1/c_l$. Therefore again we can interpret equilibrium prices as Lagrangian multipliers.

To describe the rate adjustment (6), let

$$\overline{x}_i(t) \quad = \quad U_i'^{-1}(q_i(t)) \quad = \quad \frac{\alpha_i d_i}{q_i(t)}$$

be the target rate chosen based on the end-to-end queueing delay $q_i(t)$ and the marginal utility U_i'. Then Vegas' source algorithm is:

$$\dot{x}_i \quad = \quad \left\{ \begin{array}{ll} \frac{1}{\tau_i^2} & \text{if } \; x_i(t) < \overline{x}_i(t) \\ -\frac{1}{\tau_i^2} & \text{if } \; x_i(t) > \overline{x}_i(t) \end{array} \right\}$$

moving the source rate $x_i(t)$ towards the target rate $\overline{x}_i(t)$ at a pace of $1/\tau_i^2$.

The models of TCP Reno and Vegas are summarized in Table 1.

3.2 Dynamics and Stability

As discussed in the previous section, static or dynamic control laws at sources and links attempt to drive the system to a desirable equilibrium point. So far we have only used dynamic models

TCP	Model	
Reno	Source control	$\dot{x}_i = \frac{1-q_i(t)}{\tau_i^2} - \frac{1}{2}q_i(t)x_i^2(t)$
RED	Link control	$\dot{b}_l = \begin{cases} (y_l(t)-c_l) & \text{if } b_l(t)>0 \\ [y_l(t)-c_l]^+ & \text{if } b_l(t)=0 \end{cases}$ $\dot{r}_l = -\alpha_l c_l\,(r_l(t)-b_l(t))$, $p_l = m_l(r_l)$
	Utility	$U_i(x_i) = \frac{\sqrt{2}}{\tau_i}\,\tan^{-1}\left(\frac{\tau_i x_i}{\sqrt{2}}\right)$
Vegas	Source control	$\dot{x}_i = \begin{cases} \frac{1}{\tau_i^2} & \text{if } x_i(t) < \overline{x}_i(t) \\ -\frac{1}{\tau_i^2} & \text{if } x_i(t) > \overline{x}_i(t) \end{cases}$
FIFO	Link control	$\dot{p}_l = \begin{cases} \frac{1}{c_l}(y_l(t)-c_l) & \text{if } p_l(t)>0 \\ \frac{1}{c_l}[y_l(t)-c_l]^+ & \text{if } p_l(t)=0 \end{cases}$
	Utility	$U_i(x_i) = \alpha_i d_i \log x_i$

Table 1: Models of TCP/AQM. In the above $\overline{x}_i(t) = U_i'^{-1}(q_i(t))$.

to derive the equilibrium points, and we note that there can be different dynamic laws with the same equilibrium, only distinguished by their dynamic properties, which we now discuss. Mainly, we would like to determine whether the equilibrium is (locally or globally) stable. We begin with a brief overview of some dynamic laws which have been proposed in the optimization framework, and which allow for analytical stability results. We will then move to study in detail the dynamics of TCP Reno/RED.

In general, one could have dynamics at both sources and links; however, most analytical results refer to systems where only one of the laws is dynamic, and the other static. In this regard, [8] denotes by primal algorithms those where the dynamics are at the sources, and by dual algorithms when the dynamics are at the links. An example of dynamics at the source is the first order law (used in [8] for a particular utility function)

$$\dot{x_i} = \kappa_i \left(U'(x_i) - q_i\right) x_i.$$

Combined with a static link law $p_l = h_l(y_l)$, it is shown in [8] that the system, in the absence of delays, has a single global attractor, which optimizes the cost function in (14); indeed this modified utility serves as a Lyapunov function for the system.

An example of dynamics at the links was already given in (13); combined with the static source control

$$x_i(t) = [U_i'^{-1}(q_i(t))]^+, \tag{23}$$

it is shown in [10] that global stability is obtained. Another link algorithm proposed in [45, 46], is

$$\dot{p}_l = \begin{cases} \gamma_l(y_l - c_l + \alpha_l b_l) & \text{if } p_l(t) > 0 \\ \gamma_l \left[y_l - c_l + \alpha_l b_l\right]^+ & \text{if } p_l(t) = 0 \end{cases}$$

where b_l is the queue length as in (19). Global stability in the absence of delay of this scheme, together with (23), has been proved in [14] by Lyapunov argument. This protocol can be implemented in a similar fashion to RED, but simulations in [45, 46] have shown a marked improvement over RED in terms of achieving fast responses and low queues.

Recently, [16] has proposed a scheme with dynamics at both links and sources, however working at different time-scales; thus stability analysis reduces to two problems, one with static links and one with static sources.

We emphasize that the Lyapunov-based stability proofs in [8, 14], while global, do not consider network delays. Some local, linearized studies in [8, 23] can be given to study tolerance to delay, and in general yield bounds on the various gain parameters of the algorithms (e.g., κ_i, γ_l must be inversely proportional to delay) to maintain local stability. In turn, the global stability analysis in [10] is done in discrete-time, possibly asynchronous, and does allow for delays, but again stability is guaranteed only if gain parameters are sufficiently small.

Note that delays are the only dynamics of the open loop system described in Figure 1; were it not for delay, the rate and price adaptation could be performed arbitrarily fast. Thus it is natural that gain parameters should be chosen inversely proportional to delay. Since sources measure their round-trip time, and a scaling by $1/\tau_i$ is already implicit in a window-based protocol due to (15), (the "self-clocking" feature), this raises the intriguing possibility that compensation for delay could be done automatically in a protocol such as Reno.

We now turn to a detailed study of Reno/RED that contains dynamics at both sources and links.

3.2.1 Dynamics of Reno/RED

We now study the dynamic properties around an equilibrium by linearizing the model developed before. Before we do that, we must refine the nonlinear model to include the effect of delays, which are essential to stability analysis.

For this purpose we must account for forward and backward delays in the propagation of rates and prices, as was done in (1)–(2). For the window dynamics, a first approximation would be:

$$\dot{w}_i = x_i(t - \tau_i)(1 - q_i(t))\frac{1}{w_i(t)} - x_i(t - \tau_i)q_i(t)\frac{w_i(t)}{2}, \tag{24}$$

with $q_i(t)$ as in (2). Here we incorporate the fact that the rate of incoming ACKs is determined by the source rate τ_i units of time ago. However a subtle issue that arises in these models is that

round-trip time is itself time-varying, since it depends on queueing delays. In particular, round-trip time can be expressed as

$$\tau_i(t) = d_i + \sum_l R_{li} \frac{b_l}{c_l},$$

where d_i is the round-trip propagation delay and b_l is the backlog at link l at the time the packet arrived at the link. These arrival times are difficult to capture exactly, since they depend themselves on queueing delays earlier in the path; this would mean that the time argument in b_l above would depend recursively on other queues, themselves at earlier times, and so on. We avoid this issue by assuming a common time t for all the queues,

$$\tau_i(t) \quad = \quad d_i + \sum_l R_{li} \frac{b_l(t)}{c_l}. \tag{25}$$

This simplification is acceptable provided we are not attempting for our model to have time resolution smaller than round-trip times. Still, difficulties remain. In particular if one generalizes (15) to

$$x_i(t) \quad = \quad \frac{w_i(t)}{\tau_i(t)}, \tag{26}$$

the window equation (24) would contain $x_i(t - \tau_i(t)) = \frac{w_i(t-\tau_i(t))}{\tau_i(t-\tau_i(t))}$, with a nested time argument in the round-trip time, which is not easy to interpret. For this reason we adopt the following convention: whenever round-trip time, or forward and backward delay, appear in the *argument* of a variable, we will replace it by its equilibrium value τ_i^*, τ_{li}^{f*}, τ_{li}^{b*}. This accounts for equilibrium queueing delays, but at that level does not include their variation in time. However when round-trip time appears in the dependent variable as in (26), we will consider it time-varying and use for it the model in (25). This avoids recursive time-arguments, but is admittedly an approximation, done exclusively for model tractability.[2] Confidence in these approximation can only be obtained through comparisons with packet-level simulations.

With these approximations, we obtain the following nonlinear, time-delayed model for Reno, expanding on (24):

$$\dot{w}_i = \left(1 - \sum_l R_{li} p_l(t - \tau_{li}^{b*})\right) \frac{w_i(t - \tau_i^*)}{\tau_i(t - \tau_i^*)} \frac{1}{w_i(t)} \quad - \quad \frac{1}{2} \sum_l R_{li} p_l(t - \tau_{li}^{b*}) \frac{w_i(t - \tau_i^*)\, w_i(t)}{\tau_i(t - \tau_i^*)}.$$

Assuming the routing matrix R has full rank then there is a unique equilibrium (w^*, p^*, q^*). Linearizing around it, we have (variables now denote perturbations)

$$\dot{w}_i \quad = \quad -\frac{1}{\tau_i^* q_i^*} \sum_l R_{li} p_l(t - \tau_{li}^{b*}) \quad - \quad \frac{q_i^* w_i^*}{\tau_i^*} w_i(t).$$

[2]Similar, though slightly more restrictive, approximations were made in [22]. As this paper first noted, and we corroborate below, retaining delay variations in the round-trip time is essential for the predictive power of this model.

Now consider the link dynamics of RED, as described in Table 1. For the purposes of linearization, we note that non-bottleneck links (with empty equilibrium queues) can be ignored. For bottleneck links, we make the assumption that rate increases of a source affect all bottlenecks in its path,[3] and write

$$\begin{aligned}\dot{b}_l &= \sum_i R_{li}\frac{w_i(t-\tau_{li}^{f*})}{\tau_i(t-\tau_{li}^{f*})} - c_l \\ &= \sum_i R_{li}\frac{w_i(t-\tau_{li}^{f*})}{d_i + \sum_k R_{ki}b_k(t-\tau_{li}^{f*})/c_k} - c_l.\end{aligned}$$

Let $\tau_i^* = d_i + \sum_k R_{ki}b_k^*/c_k$ be the equilibrium round-trip time (including queueing delay). Linearizing we have (variables now denote perturbations):

$$\dot{b}_l = \sum_i R_{li}\frac{w_i(t-\tau_{li}^{f*})}{\tau_i^*} - \sum_k\sum_i R_{li}R_{mi}\frac{w_i^*}{(\tau_i^*)^2 c_k}\, b_k(t-\tau_{li}^{f*}).$$

The second term above would be ignored if we did not include queueing delay in the round-trip time. The double summation sums over all links k that share any source i with link l. It says that the link dynamics in the network are coupled through shared sources. The term $\frac{w_i^*}{\tau_i^* c_k} b_k(t-\tau_{li}^{f*})$ is roughly the backlog at link k due to packets of source i, under FIFO queueing. Hence the backlog $b_l(t)$ at link l decreases at a rate that is proportional to the backlog of this shared source i at another link k. This is because the backlog in the path of source i reduces the *rate* at which source i packets arrive at link l and hence decreases $b_l(t)$.

Putting everything together, Reno/RED is described by, in Laplace domain,

$$\begin{aligned}w(s) &= -(sI+D_1)^{-1}D_2R_b^T(s)p(s) \\ p(s) &= (sI+D_3)^{-1}D_4 b(s) \\ b(s) &= (sI+R_f(s)D_5R^TD_6)^{-1}R_f(s)D_7w(s)\end{aligned} \tag{27}$$

where the diagonal matrices are

$$\begin{aligned}D_1 &= \text{diag}\left(\frac{q_i^*w_i^*}{\tau_i^*}\right), & D_2 &= \text{diag}\left(\frac{1}{\tau_i^*q_i^*}\right) \\ D_3 &= \text{diag}\,(\alpha_l c_l), & D_4 &= \text{diag}\,(\alpha_l c_l \rho_l) \\ D_5 &= \text{diag}\left(\frac{w_i^*}{(\tau_i^*)^2}\right), & D_6 &= \text{diag}\left(\frac{1}{c_l}\right), \quad D_7 = \text{diag}\left(\frac{1}{\tau_i^*}\right)\end{aligned}$$

and $R_f(s)$ and $R_b(s)$ are defined in (3).

To gain some insight into the system's behavior, let us specialize to the case of a single link with N identical sources (see generalization to heterogeneous sources in [47]). The transfer functions are

[3] We do not model the throttling effect that upstream links may have on downstream ones.

(dropping all subscripts and eliminating $b(s)$)

$$w(s) = -\frac{e^{-\tau^{b*}s}}{p^*(\tau^* s + p^* w^*)}\, p(s) \tag{28}$$

$$p(s) = \frac{\alpha c \rho}{s + \alpha c}\, \frac{N e^{-\tau^{f*}s}}{\tau^* s + e^{-\tau^{f*}s}}\, w(s) \tag{29}$$

where $w^* = c\tau^*/N$ is the equilibrium window size and p^* is the equilibrium loss probability. When forward delay $\tau_f^* = 0$, the model (28)–(29) reduces to that in [22]. It is easy to show that $\tau^{f*} < \tau^*$ implies that (29) is open-loop stable. Hence the closed loop system is stable if and only if the loop function

$$L(s) = \frac{\alpha c \rho}{s + \alpha c}\, \frac{1}{\tau^* s + e^{-\tau^{f*}s}}\, \frac{N}{p^*(\tau^* s + p^* w^*)}\, e^{-\tau^* s}$$

does not encircle $(-1, 0)$ as s traverses the closed D contour in the complex plane. Substitution with the equilibrium values of p^* and w^* yields (using $p^* \simeq 2/w^{*2}$)

$$L(s) = \frac{\alpha c \rho}{s + \alpha c}\, \frac{1}{\tau^* s + e^{-\tau^{f*}s}}\, \frac{c^3 \tau^{*3}}{2N(c\tau^{*2} s + 2N)}\, e^{-\tau^* s}. \tag{30}$$

The first factor above is due to queue averaging in RED. The second factor describes the relation between windows and buffer size; the term $e^{-\tau^{f*}s}$ arises due to the effect of queueing delays in equation (27). If $\tau^{f*} = 0$ it reduces to a lowpass filter, of time constant $1/\tau^*$. The third term is due to Reno; it has a DC gain of $\frac{c^3\tau^{*3}}{4N^2} = \frac{w^{*3}N}{4}$, and a pole at $\frac{2N}{c\tau^{*2}} = \frac{2}{w^*\tau^*}$. Under typical conditions, the equilibrium window satisfies $w^* \gg 2$ so the system has high gain at low frequencies, and a pole which is slower than that of the second term. Finally, we have the round-trip feedback delay.

The above loop gain consists of a stable function times a pure delay. The latter will provide significant phase, starting at frequencies of the order of $1/\tau$. Therefore a classical Bode plot analysis says that closed loop stability will require that the loop gain at those frequencies be below unity.[4] This suggests that it is difficult or RED to stabilize Reno, as extensive simulation experience has shown. In particular, assuming τ or c become large, at a fixed frequency ω the term will be approximately

$$\frac{c^2\tau^*}{2Nj\omega}$$

which grows in magnitude, and has 90 degrees of phase, and is thus destabilizing. Similar conclusions happen when N is small. Note that instability for high τ is not surprising, however here it shows that Reno is not successful in scaling down gains by τ, as was suggested as a stabilizing method in the beginning of the section, based on the self-clocking property. Perhaps more striking

[4] Equivalently, one might say that it is impossible for a stable loop to track variations which are faster than the pure delay of the loop.

is the destabilizing effect of high capacity; as routers become faster Reno is bound to go into an unstable regime. Intuitively, this is because higher capacity leads to higher equilibrium rate. This produces higher gain in the multiplicative term of AIMD since sources update more frequently, on each acknowledgment, with a larger amplitude.

3.2.2 Simulation Studies

The equilibrium model has been validated in [48] for Reno and in [12] for Vegas. In this subsection, we present simulation results to validate our linear dynamic model when the system is stable or barely unstable. They also illustrate numerically the stability region of Reno/RED.

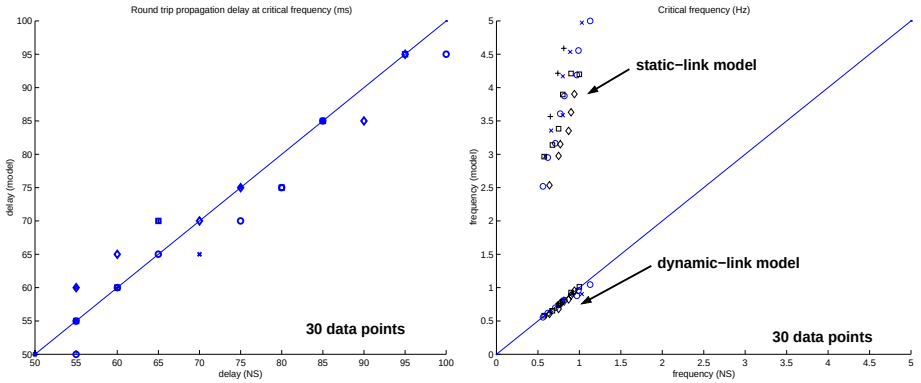
We consider a single link of capacity c pkts/ms shared by N sources with identical round-trip propagation delay d ms. For $N = 20, 30, \ldots, 60$ sources, capacity $c = 8, 9, \ldots, 15$ pkts/ms, and propagation delay $d = 50, 55, \ldots, 100$ ms, we examine the Nyquist plot of the loop gain of the feedback system ($L(j\omega)$ in (30) above). For each (N, c) pair, we determine the delay $d_m(N, c)$, at 5ms increment, at which the smallest intercept of the Nyquist plot with the real axis is closest to -1. This is the delay at which the system (N, c) transits from stability to instability according to the linear model. For this delay, we compute the critical frequency $f_m(N, c)$ at which the phase of $L(j\omega)$ is $-\pi$. Note that the computation of $L(j\omega)$ requires equilibrium round-trip time τ, the sum of propagation delay $d_m(N, c)$ and equilibrium queueing delay. The queueing delay is calculated from the equilibrium model. Hence, for each (N, c) pair that becomes barely unstable at a delay between 50ms and 100ms, we obtain the critical (propagation) delay $d_m(N, c)$ and the critical frequency $f_m(N, c)$.

We repeat these experiments in *ns-2*, using persistent FTP sources and RED with ECN marking. The RED parameters are (0.1, 40pkts, 540pkts, 10^{-4}). For each (N, c) pair, we examine the queue and window trajectories to determine the critical delay $d_{ns}(N, c)$ when the system transits from stability to instability. We measure the critical frequency $f_{ns}(N, c)$, the fundamental frequency of queue oscillation, from the FFT of the queue trajectory. Thus, corresponding to the linear model, we obtain the critical delay $d_{ns}(N, c)$ and frequency $f_{ns}(N, c)$ from simulations.

We compare model prediction with simulation. Figure 2(a) plots the critical delay $d_m(N, c)$ computed from the linear model versus the critical delay $d_{ns}(N, c)$ from *ns-2* simulations. Each data point (30 of them) corresponds to a particular (N, c) pair. The dotted line is where all points should lie if the linear model agrees perfectly with the simulation. Figure 2(b) gives the corresponding plot for critical frequencies $f_m(N, c)$ versus $f_{ns}(N, c)$. The agreement between model and simulation seems quite reasonable (recall that delay values have a resolution of 5ms).

Consider a static link model where loss probability is a function of link flow rate:

$$p_l(t) \quad = \quad f_l(y_l(t))$$



(a) Critical delay (ms) (b) Critical frequency (Hz)

Figure 2: Validation: comparison of critical (round-trip propagation) delay and critical frequency computed from linear model and measured from simulation.

Then the linearized model is (variables now are perturbations)

$$p_l(t) \quad = \quad f_l'(y_l^*)\ y_l(t)$$

where $f_l'(y_l^*)$ is the derivative of f_l evaluated at equilibrium. Also shown in Figure 2(b) are critical frequency predicted from this static-link model (with $f_l'(y_l^*) = \rho$), using the same Nyquist plot method described above. It shows that queue dynamics is significant at the time-scale of interest.

Figure 3 illustrates the stability region implied by the linear model. For each N, it plots the

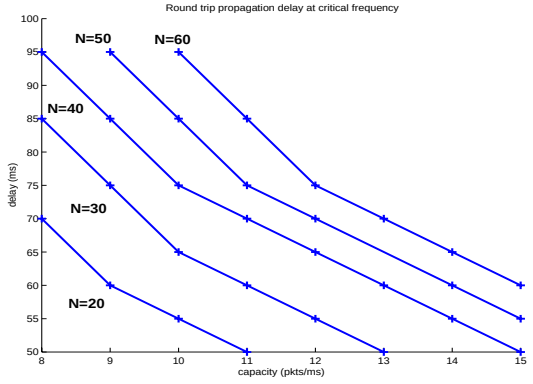


Figure 3: Stability region: for each N, the region above the curve is unstable and that below is stable.

critical delay $d_m(N, c)$ versus capacity c. The curve separates stable (below) from unstable regions (above). The negative slope shows that TCP/RED becomes unstable when delay or capacity is

large. As N increases, the stability region expands, i.e., small load induces instability. A larger delay or capacity, or a smaller load, leads to a larger equilibrium window; this confirms the folklore that TCP behaves poorly at large window size.

4 A Scalable Control

The above discussion suggests that the current protocol may be ill-suited for future networks where both delay and capacity can be large. This has motivated the search for protocols which scale properly so as to maintain stability in the presence of these variations. In regard to delay stability, scaling of source gains by $1/\tau_i$ was first suggested in [25], and subsequently proved in [26, 27, 28] to provide stability for "primal" laws involving first-order source control and static link marking, provided delay or control gain is sufficiently small. The concept of scaling down gains by delay was already mentioned before in the context of optimization-based stability proofs; when the scaling is done by a common global constant, this can be very conservative. In contrast, individualized scaling as suggested here has the appealing feature that sources with low round-trip times can respond quickly, and take advantage of available bandwidth, and it is only those sources whose fast response compromises stability (those with long delays) that must slow down.

In this section we describe a protocol, developed in [49], that can be implemented in a decentralized way by sources and links and that satisfies some basic objectives: high network utilization in equilibrium and local stability for *arbitrary* delays, capacities, and routing. These requirements impose certain constraints on the linearized dynamics: integration at links, and conditions on the gain at sources and links. We will present a global implementation by nonlinear algorithms at sources and links that are consistent with the linearization requirements. We will also discuss signaling implications of this protocol, and conclude with a packet-level simulation that validates the theoretical results.

4.1 Objectives and Linear Design

We now lay out a series of objectives for the feedback control laws in purely local (linearized) terms. These will lead us to a local control law, which is proved in [49] to achieve these objectives.

A first objective is that the target capacity c_l is matched at equilibrium; as in (13), this can be achieved when prices *integrate* the excess capacity (variables denote perturbations in this subsection):

$$\dot{p}_l = \gamma_l y_l.$$

Is this the only choice? If exact tracking of capacity is desired, there must be an integrator in the loop; and as explained in [49], to have stability, we must perform the integration at the

lower-dimensional end of the problem, i.e., at the links. So the above is the simplest law consistent with this objective; the constant γ_l will be chosen later.

The next, main objective is to have local dynamic stability for arbitrary network delays, link capacities, and routing topologies.

We now argue more carefully for the desired scaling as a function of delay. As remarked before, delays are the only dynamics of the open loop; therefore, they are the only thing that sets a time-scale to our closed-loop behavior. Thus we aim here for a system where a scaling of all delays by a common factor would result in identical time responses except for time scale. Consider first a single-link, single-source problem. The network delay and the above link integrator will yield a term

$$\frac{e^{-\tau s}}{s}$$

in the loop transfer function. Thinking, e.g., in terms of its Nyquist plot, instability will always occur at high values of τ unless the gain is made a function of τ. Indeed, introducing a gain K/τ in the loop (specifically at the source) leads to a loop gain

$$K\frac{e^{-\tau s}}{\tau s},$$

which is scale-invariant: namely, its frequency response is a function of $\tau\omega$, so Nyquist plots for all values of τ would fall on a single curve, and by choosing K appropriately one can ensure stability for all τ. Further, the time responses of such a loop will give the desired invariance up to scale. Inspired by this, for the multi-link multi-source case, we are led to include a factor $1/\tau_i$ in the control gain at each source i.

Now we turn to invariance to capacity and routing. It is crucial that the overall loop gain introduced by the routing matrices R_f, R_b be kept under control; intuitively, as more sources or links participate in the feedback the gain must be appropriately scaled down. The difficulty is implementing this in a decentralized fashion, without access to the global routing information. To scale down the gain due to R_f at links, we exploit the fact that, at equilibrium, the aggregate source rates add up to capacity, so $R_f(0)x^* = c$. Since sources know their rates and links their capacities, one is led to introduce a gain $\frac{1}{c_l}$ at each link, and a gain x_i^* at each source. To scale down the gain due to R_b at sources, we introduce a gain $\frac{1}{M_i}$ at each source, M_i being the number of bottleneck links in the source's path.

Summarizing the above requirements in their simplest possible form, we propose the following linear control laws:

- For the source, a static gain (from q_i to x_i) of

$$-\kappa_i \;:=\; -\frac{\alpha_i x_i^*}{M_i \tau_i} \tag{31}$$

where α_i is a gain parameter to be chosen, x_i^* is the equilibrium rate, τ_i is round-trip time, and M_i is a bound on the number of bottleneck links in source i's path. The sign is used to provide negative feedback from prices to rates.

- For the link, an integrator with gain normalized by capacity,

$$p_l \quad = \quad \frac{1}{c_l s} y_l. \tag{32}$$

Note that the normalization gives this price units of *time*. In practice, c_l is chosen to be strictly less than the actual link capacity in order to maintain zero buffer in equilibrium. If c_l were the actual link capacity, p_l would represent the queueing delay at the link and is the price signal used in TCP Vegas (see (22)); so here we can think of p_l as a "virtual" queueing delay (see [16] for related ideas).

It is proved in [49] that, provided the routing matrix R has full row rank, and the gains $\alpha_i < 1$, the feedback system with source algorithm (31) and link algorithm (32) is linearly stable for *arbitrary* delays, link capacities, and routing.

4.2 Global, Nonlinear Implementation

We now describe a global implementation by links and sources that have suitable equilibrium points, around which the linearization satisfies the requirements laid out above.

The price dynamics can be implemented by the link algorithm

$$\dot{p}_l \quad = \quad \begin{cases} \frac{1}{c_l}\left(y_l(t) - c_l\right) & \text{if } p_l(t) > 0; \\ \frac{1}{c_l}\left(y_l(t) - c_l\right)^+ & \text{if } p_l(t) = 0 \end{cases} .$$

That is, prices integrate excess capacity in a normalized way, and are saturated to be always non-negative. At equilibrium, bottlenecks with nonzero price will have $y_l = c_l$ as required. Non-bottlenecks with $y_l < c_l$ will have zero price.

For the sources, the linearization requirement (31) leads to a differential equation

$$\frac{\partial f_i}{\partial q_i} = -\frac{\alpha_i f_i(q_i)}{M_i \tau_i},$$

which can be solved analytically, and gives the control law

$$x_i \quad = \quad f_i(q_i) \quad := \quad x_{\max,i}\, e^{-\frac{\alpha_i q_i}{M_i \tau_i}} \tag{33}$$

as the static source law. Note that the smaller the delay, the more responsive a source is in varying its rate as a function of price. The larger the delay, the more conservative the source control is, in order to avoid instability.

Here, $x_{\max,i}$ is a maximum rate parameter, which can vary for each source, and in fact can also be scheduled to depend on M_i, τ_i. All that is required is that it does not depend on q_i. For instance, $x_{\max,i}$ can be chosen to compensate for the effect of delay τ_i on equilibrium rate.

The utility function corresponding to the source control is

$$U_i(x) = \frac{M_i \tau_i}{\alpha_i} x \left[1 - \log \left(\frac{x}{x_{\max,i}} \right) \right], \quad \text{for } x \leq x_{\max,i};$$

as illustrated in Figure 4.

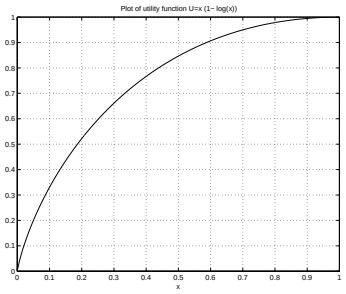


Figure 4: Utility function of source algorithm (31).

The above is the simplest nonlinear source law that gives the required scaling; as discussed in [49], some more degrees of freedom are available by letting α_i be a function of q_i; however it is important to emphasize that the stability requirements do pose constraints on the family of utility functions.

4.3 Packet-level Implementation Requirements

We briefly discuss here the information needed at sources and links to implement the dynamic laws we defined, and resulting communication requirements.

Links must have access to their aggregate flow y_l; this quantity is not directly known, but can be estimated from arriving packets (see, e.g. [45]). Note also that since the rate is integrated there is smoothing already built into the process. The target capacity c_l is assumed known. Indeed, a simple way to implement the price updates is to maintain a "virtual queue" counter that is incremented with received packets, decremented at the virtual capacity rate. Then prices are obtained by dividing this counter by the virtual capacity.

Sources must have access to the round-trip time τ_i, which can be obtained by timing packets and their acknowledgments. Note that since the target equilibrium state is with empty queues, at equilibrium we will have $\tau_i^* = d_i$; it is therefore recommended to use an estimate of d_i (typically the minimum observed round-trip time), as is done in Vegas. This avoids the possibility that temporary excursions of the real queue (inevitable at the packet level) would have a destabilizing effect via the round-trip time.

Also, sources must receive two parameters from the network: the aggregate price q_i, and the number of bottlenecks M_i. To communicate q_i, the technique of random exponential marking [45] can be used. Here, an ECN bit would be marked at each link l with probability $1 - \phi^{-p_l}, \quad \phi > 1$. Assuming independence, the overall probability that a packet from source i gets marked is $1 - \phi^{-q_i}$, and therefore q_i can be estimated from marking statistics at each source. This requires the knowledge of ϕ which is presumably a global constant set a priori. Alternatively packet dropping can be used in lieu of marking as in RED; provided drop probabilities are low, the superposition of probabilities holds to a good approximation.

Regarding M_i, in the simplest implementation one would simply employ an upper bound, which could be a globally known constant, or based on the total number of links in the route, found e.g. from traceroute information in IP. For a more aggressive control one would need to communicate in real-time, how many links are bottlenecks. This can be done in an analogous way as what is done with prices, using a second ECN bit. Assume that links which are bottlenecks mark this bit with probability $1 - \phi^{-1}$, and those who are not do not mark. Then the probability of marking on a route is $1 - \phi^{-M_i}$, which allows for the estimation of M_i at the sources in real time. This introduces a time-varying source law whose closed loop stability requires further study.

Once the relevant parameters are measured, or estimated, the static exponential law in (33) can be used to determined the desired rate. Multiplying by the measured round-trip time gives the desired window (after a round off) which can then directly be used to control transmission. Thus we have an implementation scenario which only adds moderate amount of overhead to the current Internet protocols.

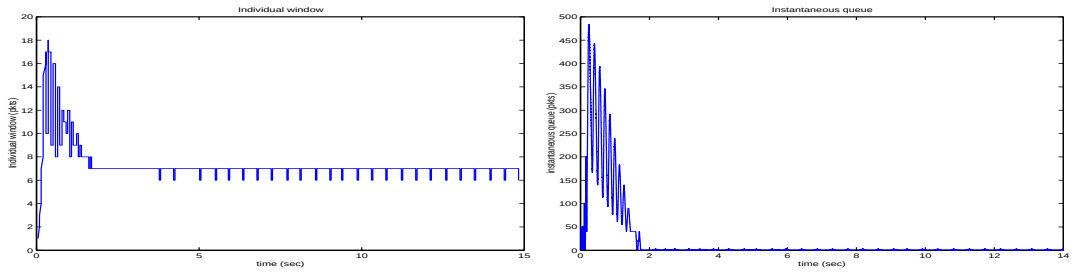
4.4 Simulation Studies

A packet level implementation is done using parallel simulator Parsec [50]. The simulation includes window management, link queueing and delay, but at this point does not include marking; prices are communicated as floating point numbers. We simulate a single link with capacity 9 pkts/ms that is shared by 50 sources. Figure 5 shows the *individual* window and queue as a function of time when the round-trip propagation delay is 40ms and 200ms, respectively. As expected, both the individual window and queue converge regardless of delay. Longer delay sets a larger time-scale for the closed loop behavior.

Acknowledgments. The simulation results are taken from [47] and conducted by Sachin Adlakha and Jiantao Wang.

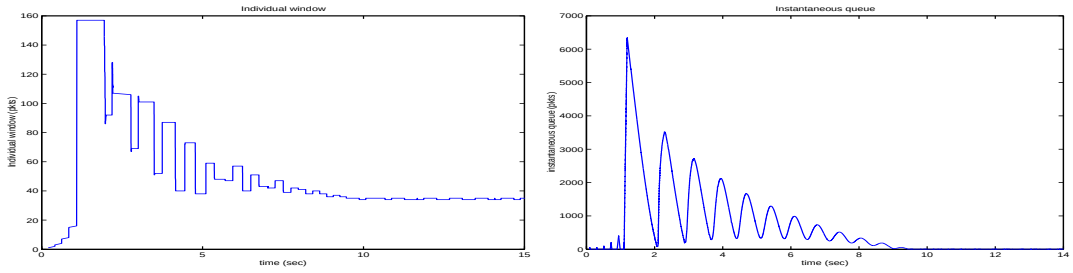
References

[1] L. Benmohamed and S. M. Meerkov. Feedback control of congestion in store-and-forward networks: the case of a single congested node. *IEEE/ACM Transactions on Networking*, 1(6):693–707, December



(a) Individual window (delay = 40ms)

(b) Queue (delay = 40ms)



(c) Individual window (delay = 200ms)

(d) Queue (delay = 200ms)

Figure 5: Individual window and queue traces. Simulation parameters: 50 sources, capacity = 9 pkts/ms, $\alpha = 0.8$, virtual capacity = 95%.

1993.

[2] S. Chong, R. Nagarajan, and Y.-T. Wang. Designing stable ABR flow control with rate feedback and open loop control: first order control case. *Performance Evaluation*, 34(4):189–206, December 1998.

[3] E. Altman, T. Basar, and R. Srikant. Congestion control as a stochastic control problem with action delays. *Automatica*, December 1999.

[4] H. Ozbay, S. Kalyanaraman, and A. Iftar. On rate-based congestion control in high-speed networks: design of an h_∞ based flow controller for single bottleneck. In *Proc. American Control Conference*, 1998.

[5] S. Mascolo. Congestion control in high-speed communication networks using the smith principle. *Automatica*, December 1999.

[6] E. J. Hernandez-Valencia, L. Benmohamed, R. Nagarajan, and S. Chong. Rate control algorithms for the ATM ABR service. *European Transactions on Telecommunications*, 8:7–20, 1997.

[7] R. Srikant. Control of communication networks. In Tariq Samad, editor, *Perspectives in Control Engineering: Technologies, Applications, New Directions*, pages 462–488. IEEE Press, 2000.

[8] Frank P. Kelly, Aman Maulloo, and David Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of Operations Research Society*, 49(3):237–252, March 1998.

[9] Frank P. Kelly. Mathematical modelling of the Internet. In *Proc. 4th International Congress on Industrial and Applied Mathematics*, July 1999. `http://www.statslab.cam.ac.uk/~frank/mmi.html`.

[10] Steven H. Low and David E. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999. `http://netlab.caltech.edu`.

[11] Steven H. Low. A duality model of TCP flow controls. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, September 18-20 2000. `http://netlab.caltech.edu`.

[12] Steven H. Low, Larry Peterson, and Limin Wang. Understanding Vegas: a duality model. In *Proceedings of ACM Sigmetrics*, June 2001. `http://netlab.caltech.edu/pub.html`.

[13] Sanjeewa Athuraliya and Steven H. Low. Optimization flow control with Newton–like algorithm. *Journal of Telecommunication Systems*, 15(3/4):345–358, 2000.

[14] Fernando Paganini. On the stability of optimization-based flow control. In *Proceedings of American Control Conference*, 2001. `http://www.ee.ucla.edu/~paganini/PS/remproof.ps`.

[15] Srisankar Kunniyur and R. Srikant. End–to–end congestion control schemes: utility functions, random losses and ECN marks. In *Proceedings of IEEE Infocom*, March 2000. `http://www.ieee-infocom.org/2000/papers/401.ps`.

[16] Srisankar Kunniyur and R. Srikant. A time–scale decomposition approach to adaptive ECN marking. In *Proceedings of IEEE Infocom*, April 2001. `http://comm.csl.uiuc.edu:80/~srikant/pub.html`.

[17] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, October 2000.

[18] J. Mo, R. La, V. Anantharam, and J. Walrand. Analysis and comparison of TCP Reno and Vegas. In *Proceedings of IEEE Infocom*, March 1999.

[19] Richard La and Venket Anantharam. Charge-sensitive TCP and rate control in the Internet. In *Proceedings of IEEE Infocom*, March 2000. `http://www.ieee-infocom.org/2000/papers/401.ps`.

[20] Koushik Kar, Saswati Sarkar, and Leandros Tassiulas. Optimization based rate control for multirate multicast sessions. In *Proceedings of IEEE Infocom*, April 2001.

[21] V. Misra, W-B Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting tcp flows with an application to RED. In *Proceedings of ACM SIGCOMM*, 2000.

[22] Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. A control theoretic analysis of RED. In *Proceedings of IEEE Infocom*, April 2001. `http://www-net.cs.umass.edu/papers/papers.html`.

[23] Fernando Paganini. Flow control via pricing: a feedback perspective. In *Proceedings of the 2000 Allerton Conference*, October 2000.

[24] V. Jacobson. Congestion avoidance and control. *Proceedings of SIGCOMM'88, ACM*, August 1988. An updated version is available via `ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z`.

[25] Frank P. Kelly. Models for a self-managed Internet. In *Proceedings of Royal Society Meeting*, December 1999. `http://www.statslab.cam.ac.uk/~frank/smi.html`.

[26] R. Johari and D Tan. End-to-end congestion control for the internet: Delays and stability. Technical report, 2000. Cambridge Univ. Statistical Laboratory Research Report 2000-2.

[27] L. Massoulie. Stability of distributed congestion control with heterogeneous feedback delays. Technical report, Microsoft Research, Cambridge UK, TR 2000-111, 2000.

[28] Glenn Vinnicombe. On the stability of end-to-end congestion control for the Internet. Technical report, Cambridge Univiversity, CUED/F-INFENG/TR.398, December 2000.

[29] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self–similar nature of Ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[30] Vern Paxson and Sally Floyd. Wide–area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[31] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self–similarity through high variability: statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, 1997.

[32] M. E. Crovella and A. Bestavros. Self–similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[33] Xiaoyun Zhu, Jie Yu, and John C. Doyle. Heavy tails, generalized coding, and optimal web layout. In *Proceedings of IEEE Infocom*, April 2001.

[34] Lawrence S. Brakmo and Larry L. Peterson. TCP Vegas: end to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8), October 1995. `http://cs.princeton.edu/nsg/papers/jsac-vegas.ps`.

[35] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, August 1993. `ftp://ftp.ee.lbl.gov/papers/early.ps.gz`.

[36] G. de Veciana, T. J. Lee, and T. Konstantopoulos. Stability and performance analysis of networks supporting elastic services. *IEEE/ACM Transactions on Networking*, 9(1), February 2001.

[37] F. Baccelli and D. Hong. AIMD, fairness and fractal scaling of TCP traffic. Technical report, INRIA, Paris, France, 2001. RR 4155.

[38] Frank P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997. `http://www.statslab.cam.ac.uk/~frank/elastic.html`.

[39] H. Yaiche, R. R. Mazumdar, and C. Rosenberg. A game theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Transactions on Networking*, 8(5), October 2000.

[40] T. V. Lakshman and Upamanyu Madhow. The performance of TCP/IP for networks with high bandwidth–delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997. http://www.ece.ucsb.edu/Faculty/Madhow/Publications/ton97.ps.

[41] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM Computer Communication Review*, 27(3), July 1997. http://www.psc.edu/networking/papers/model_ccr97.ps.

[42] L. Massoulie and J. Roberts. Bandwidth sharing: objectives and algorithms. In *Infocom'99*, March 1999. http://www.dmi.ens.fr/\%7Emistral/tcpworkshop.html.

[43] Glenn Vinnicombe. A new TCP with guaranteed network stability. Technical report, Cambridge Univiversity, preprint, June 2001.

[44] S. Floyd. Connections with multiple congested gateways in packet–switched networks, Part I: one–way traffic. *Computer Communications Review*, 21(5), October 1991.

[45] Sanjeewa Athuraliya, Victor H. Li, Steven H. Low, and Qinghe Yin. REM: active queue management. *IEEE Network*, May/June 2001. Extended version in *Proceedings of ITC17*, Salvador, Brazil, September 2001. http://netlab.caltech.edu.

[46] Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings of IEEE Infocom*, April 2001. http://www-net.cs.umass.edu/papers/papers.html.

[47] S. H. Low, F. Paganini, J. Wang, S. A. Adlakha, and J. C. Doyle. Linear stability of TCP/RED and a scalable control. In *Proceedings of 39th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October 2001. http://netlab.caltech.edu.

[48] Sanjeewa Athuraliya and Steven H. Low. An empirical validation of a duality model of TCP and queue management algorithms. In *Proceedings of Winter Simulation Conference*, December 2001.

[49] Fernando Paganini, John C. Doyle, and Steven H. Low. Scalable laws for stable network congestion control. In *Proceedings of Conference on Decision and Control*, December 2001. http://www.ee.ucla.edu/~paganini.

[50] Parallel simulation environment for complex systems. http://pcl.cs.ucla.edu/projects/parsec/.