Dynamics of TCP/RED and a Scalable Control

Steven H. Low Fernando Paganini Jiantao Wang Sachin Adlakha John C. Doyle

Abstract— **We demonstrate that the dynamic behavior of queue and average window is determined predominantly by the stability of TCP/RED, not by AIMD probing nor noise traffic. We develop a general multi-link multi-source model for TCP/RED and derive a local stability condition in the case of a single link with *heterogeneous* sources. We validate our model with simulations and illustrate the stability region of TCP/RED. These results suggest that TCP/RED becomes unstable when delay increases, or more strikingly, when link capacity increases. The analysis illustrates the difficulty of setting RED parameters to stabilize TCP: they can be tuned to improve stability, but only at the cost of large queues even when they are dynamically adjusted. Finally, we present a simple distributed congestion control algorithm that maintains stability for *arbitrary* network delay, capacity, load and topology.**

I. Introduction

It is well known that TCP/RED can oscillate wildly and it is extremely hard to reduce the oscillation by tuning RED parameters, e.g., [1], [2]. The additive-increase-multiplicative-decrease (AIMD) strategy employed by TCP Reno (and its variants such as NewReno and SACK) and noise-like traffic that are not effectively controlled by TCP no doubt contribute to this oscillation. Recent models e.g., [3], [4], imply however that oscillation is an inevitable outcome of the protocol itself. We present more evidence to support this view (Section II). We argue that TCP/RED oscillates not only because of its AIMD probing, and not only because of noise traffic (e.g., short lived TCP connections), but more fundamentally, it is due to instability[1]. We illustrate using *ns-2* simulations that, after smoothing out the AIMD component of the oscillation, the average behavior can either be steady with small random fluctuations (when the protocol is stable), or exhibit limit cycles of amplitude much larger than random fluctuations (when it is unstable). Moreover, this qualitative behavior persists even when a large amount of noise traffic is introduced, and even when sources have different delays. We conclude that it is stability that largely determines the dynamics of TCP/RED.

This motivates the stability characterization of TCP/RED. In Section III we develop a general nonlinear model of TCP/RED. The equilibrium structure of this model is analyzed in [5] by interpreting various TCP/AQM as carrying out distributed primal-dual algorithms over the Internet to maximize aggregate source utility in the form of congestion control. Here, we study local stability by linearizing the model around the equilibrium. The linear model generalizes the single-link identical-source model of [6]. We validate our model with simulation results and illustrate the stability region of TCP/RED. We derive a sufficient stability condition for the special case of a single link with *heterogeneous* sources. It shows that TCP/RED becomes unstable when delay increases, or more strikingly, when link capacity increases!

The gain introduced by TCP, in the case of a single link shared by identical sources, is proportional to the square of bandwidth-delay product and inversely proportional to the number of sources. Such a high gain induces instability when delay or capacity is high, and makes compensation by RED extremely difficult. In particular, RED parameters can be tuned to improve stability, but only at the cost of a large queue, even when they are dynamically adjusted.

This suggests that the current protocol is ill-suited for future networks where capacity will be large. In Section V we present a simple congestion control algorithm, developed in [7], that can be implemented in a decentralized manner by sources and links, and that is scalable: it maintains linear stability for *arbitrary* delay, capacity, load and routing. Moreover, it achieves high network utilization in equilibrium with negligible queues. We present preliminary simulations that show these benefits can be achieved without sacrificing performance such as transient response.

II. Why does TCP/RED oscillate?

What is the effect of AIMD, noise traffic, and heterogeneity of delays have on average window and instantaneous queue? In this section, we show that their effect pales in comparison with that of protocol instability.

We simulate in *ns-2* a single bottleneck link with capacity 9 pkts/ms (constant packet size = 1000bytes). The link runs RED with ECN marking in 'byte' mode (i.e., acknowledgment packets are marked with negligible probability). The RED parameters are max_p = 0.1, min_th = 50 pkts, max_th = 550 pkts, and weight for queue averaging $\alpha = 10^{-4}$. The link is shared by 50 persistent FTP sources. We have run simulations with both one-way and two-way traffic, and the behavior is very similar. The results in Figures 1 and 2 are for two-way traffic, and those in Figure 3 are for one-way traffic. Of the measurements from live Internet in [8], 85% have round trip times between 15-500ms. We perform simulations within this range of delays.

Figure 1 gives the result of two cases where connections have identical round trip propagation delay and generate traffic in both directions. Figure 1(a) shows an individual window (light curve) and the average window (dark curve), averaged over all 50 sources, both as a function of time. They are typical traces when round trip propagation delay is small (40ms in this case). Oscillations due to Reno's AIMD are prominent in the individual window, but disappear in the average window. As one would expect, since the queue av-

J. C. Doyle, S. H. Low and J. Wang are with California Institute of Technology; emails: {doyle,jiantao}@cds.caltech.edu, slow@caltech.edu. S. Adlakha and F. Paganini are with University of California, Los Angeles; emails: {paganini,sachin}@ee.ucla.edu

[1]By this, we mean that even if window is not adjusted on each acknowledgment arrival or loss event, but is adjusted periodically by the same *average* amount AIMD would over a same period, the oscillation persists.

erages individual windows, it also displays a smooth trace with small random fluctuations, as shown in Figure 1(b). We consider the *average* behavior of the protocol stable (non-oscillatory) in this case.

Figures 1(c) and (d) show the corresponding windows and queue when round trip propagation delay is increased to 200ms. Not only does the individual window oscillate with a larger amplitude, more importantly, its average displays a deterministic limit cycle. This also shows up in the queue trace. We say the protocol is in an *unstable* regime.

What is the effect of noise-like mice traffics that are not effectively controlled by TCP/RED? To get a qualitative understanding, we add short http sources to the 50 persistent bi-directional FTP connections. Each http source sends a single-packet request to its destination, which then replies with a file of size that is exponentially distributed with a mean of 12 1KB-packets. After the source completely receives the data, it waits for a random time that is exponentially distributed with a mean of 500 msec, and repeats the process. Both the request and the response are carried over TCP connections. Two sets of simulations are conducted, the first with 60 http sources generating 10% noise (i.e., persistent FTP sources occupied 90% of bottleneck link capacity), and the second set with 180 http sources generating 30% noise. The queue traces when propagation delay is 40ms (stable) and 200ms (unstable), respectively, are shown in Figures 2(a) and (b) for a noise intensity of 10%, and in Figure 2(c) and (d) for a noise intensity of 30%. The behavior of the queue and average window (not shown here) is dominated by the stability of the protocol, not by noise-like mice traffic (compare with Figures 1(b) and (d)). In the stable regime (40ms delay), the noise traffic increases the average queue length slightly. This increases the marking probability and reduces the average window of the FTP sources.

All our previous simulations are for sources with identical propagation delay. Will the dynamic behavior be very different when sources have different delays? We repeat the previous experiments, without noise, with 50 persistent unidirectional connections having delays ranging from 40ms to 64ms at 1ms increment, with 2 sources to each delay value. We study their dynamic behavior when all delays are scaled up, or down, over a wide range. The behavior is qualitatively similar to the case of identical delay, with more severe queue oscillation. Figure 3(a) shows the instantaneous queue when the scaling factor is 0.3 (delays range from $0.3(40)$ms to $0.3(64)$ms), with an average delay of 15.6ms, averaged over all sources. Figure 3(b) shows the queue when the scaling factor is 4, with an average delay of 208ms.

Instability causes three potential problems. First, it increases jitters in source rate and delay and can be detrimental to some applications. Second, it subjects short-duration connections ('mice'), that are typically delay and loss sensitive, to unnecessary delay and loss. Finally, it can lead to under-utilization of network links if queues jump between empty and full.

Hence it is protocol stability that largely determines the dynamics of TCP/RED. We now characterize when TCP/RED is stable.

III. Dynamic model and stability

In this section we develop a model of TCP/RED and use it to predict the onset of instability. We start with a nonlinear model, make a few remarks about its equilibrium properties, and then linearize the model around the equilibrium. We validate our linear model with *ns-2* simulations, and illustrate the stability region of TCP/RED. Finally we derive a stability condition for the special case of a single link with *heterogeneous* sources.

A. Nonlinear model of TCP/RED

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

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

Associated with each link l is its marking probability[2] $p_l(t)$ at time t, and with each source s its window $w_i(t)$ at time t. TCP Reno prescribes how $w_i(t)$ is adjusted and AQM prescribes how $p_l(t)$ is updated. Together they form a delayed feedback system and can be interpreted as carrying out a distributed primal-dual algorithm to solve a welfare maximization problem over the Internet [5], [9].

Define the round trip time $\tau_i(t)$ of source i at time t by:

$$\tau_i(t) \quad = \quad d_i + \sum_l R_{li} \frac{b_l(t)}{c_l} \tag{1}$$

where d_i is the round trip propagation delay and $b_l(t)$ is the backlog at link l at time t.[3] Define source i's rate $x_i(t)$ at time t as:

$$x_i(t) \quad := \quad \frac{w_i(t)}{\tau_i(t)} \tag{2}$$

The aggregate flow rate at link l is

$$y_l(t) \quad = \quad \sum_i R_{li} x_i(t - \tau_{li}^f(t)) \tag{3}$$

where $\tau_{li}^f(t)$ is the forward delay from source i to link l. The end-to-end marking probability observed at source i is $q_i(t) = 1 - \prod_{l \in L_i}(1 - p_l(t - \tau_{li}^b(t)))$ where $\tau_{li}^b(t)$ is the backward delay from link l to source i. We assume that $p_l(t)$ are small for all t so that, approximately, the end-to-end probability is

$$q_i(t) \quad := \quad \sum_l R_{li} p_l(t - \tau_{li}^b(t)) \tag{4}$$

[2]By 'marking', we mean either dropping a packet or setting an ECN (Explicit Congestion Notification) bit in the packet.

[3]This is generally not the round trip time experienced by a packet, which visits different links in its path at different times and hence experiences queueing delays of various links at different times. This expression however sums the queueing delay of different links at the *same time t*.

a sum of delayed link probabilities. The forward and backward delays are related to the round trip time through:

$$\tau_i(t) \quad = \quad \tau_{li}^f(t) + \tau_{li}^b(t)$$

for all $l \in L_i$.

We now model TCP Reno and RED. We focus on the AIMD algorithm of TCP Reno (and its variants such as NewReno and SACK). At time t, source i transmits at rate $x_i(t)$ packets/sec; hence, it receives acknowledgments at rate $x_i(t - \tau_i(t))$, assuming every packet is acknowledged. 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 $x_i(t - \tau_i(t))(1 - q_i(t))/w_i(t)$. Similarly negative acknowledgments are received at an average rate of $x_i(t-\tau_i(t))q_i(t)$, each halving the window, and hence the window $w_i(t)$ decreases at a rate of $x_i(t - \tau_i(t))q_i(t)w_i(t)/2$. Hence, the window evolves under Reno according to

$$\begin{aligned} \dot{w}_i(t) \quad &= \quad x_i(t - \tau_i(t))(1 - q_i(t))\frac{1}{w_i(t)} \\ &\quad - x_i(t - \tau_i(t))q_i(t)\frac{w_i(t)}{2} \qquad (5) \end{aligned}$$

where $q_i(t)$ is given by (4).

To model RED, let $b_l(t)$ denote the instantaneous queue length at time t that evolves according to, when $b_l(t) > 0$,

$$\dot{b}_l(t) \quad = \quad y_l(t) - c_l \qquad (6)$$

where $y_l(t)$ is the flow rate given by (3) and c_l is the link capacity. Define the average queue length as $r_l(t)$. It is updated according to:

$$\dot{r}_l(t) \quad = \quad -\alpha_l c_l \left(r_l(t) - b_l(t)\right) \qquad (7)$$

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

$$p_l(t) \quad = \quad \begin{cases} 0 & r_l(t) \le \underline{b}_l \\ \rho_l r_l(t) - \rho_l \underline{b}_l & \underline{b}_l < r_l(t) < \overline{b}_l \\ \eta_l r_l(t) - (1 - 2\overline{p}_l) & \overline{b}_l \le r_l(t) < 2\overline{b}_l \\ 1 & r_l(t) \ge 2\overline{b}_l \end{cases} \qquad (8)$$

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

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

In summary, TCP/RED is modeled by (5–8) and their interconnection through the network is modeled by (3–4).

Remarks:
1. In [5], [9], we interpret the TCP/RED model (5–8) and other TCP/AQM models as carrying out distributed primal-dual algorithms to maximize aggregate source utility over the Internet. We regard source rates $x_i(t)$ as primal variables iterated by TCP, and marking probabilities $p_l(t)$ as dual variables (Lagrange multipliers) iterated by AQM. Different protocols correspond to different update rules and they maximize different utility functions U. The utility function of TCP Reno is derived to be:

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

whereas that of TCP Vegas [10] is:

$$U_i(x_i) \quad = \quad \alpha \log x_i$$

Given any network topology R, link capacities c, and TCP utility U_i, we can determine any equilibrium properties of interest by solving a simple convex program. These include throughput, loss, delay, interaction of different TCP protocols and fairness of their equilibrium rate allocation.
2. Many implementations of Reno, or its variants, halves its window at most once in each round trip time (so does *ns-2*). In this case, the multiplicative decrease term in (5) should be replaced by $-q_i(t)w_i(t)/2\tau_i(t)$. For all simulations in this paper, the marking probability is so small that the probability of having multiple marks in a round trip time is negligible. Hence the difference between the two models of multiplicative decrease is negligible, as confirmed by the validation simulations below.

B. Linear model of TCP/RED

We linearize the TCP/RED equations (5-8) to study its stability around equilibrium. We make several simplifying assumptions. First we assume that the routing matrix R has full row rank so there is a unique equilibrium loss probability vector p (Lagrange multiplier). Second we assume that only bottleneck links, whose equilibrium marking probability is strictly positive, are included in the model. Moreover we assume that the system operates in the region $\underline{b}_l < r_l(t) < \overline{b}_l$, so that the marking probability is affine in the average queue length, $p_l(t) = \rho_l(r_l(t) - \underline{b}_l)$. Finally, we make a key assumption on the time-varying round trip delay.

Round trip delay appears in two places: first, in the relation between window $w_i(t)$ and rate $x_i(t)$, as expressed in (2), and second, in the time argument of flow rate $y_l(t)$, as expressed in (3), and the end-to-end marking probability $q_i(t)$, as expressed in (4). Inclusion of instantaneous queueing delay in the first place yields a qualitatively different model than if queueing delay is ignored or assumed constant. It means that the queue is not an integrator but has a more complicated dynamics; see (11) below. As the proof of Theorem 2 shows, this dynamic is critical to the stability of TCP/RED. The resulting linear model matches simulations significantly better than if queueing delay is assumed constant. Time-varying delay in the second place makes linearization difficult, and we replace it by its (constant) equilibrium value (including equilibrium queueing delay). Hence, we use the time-varying delay (1) in (2), but approximate the delays $\tau_i(t)$, $\tau_{li}^f(t)$, $\tau_{li}^b(t)$ by their equilibrium values in (3) and (4).

With these assumptions, we linearize Reno/RED around the unique equilibrium. From (5), Reno becomes:

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

$$- \frac{1}{2} \sum_l R_{li} p_l(t - \tau_{li}^b) \, \frac{w_i(t - \tau_i) \, w_i(t)}{\tau_i(t - \tau_i)}$$

Linearization then yields (variables now denote perturbations around the equilibrium):

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

where $q_i^* = \sum_l R_{li} p_l^*$ is the equilibrium end-to-end probability, and $w_i^* = x_i^* \tau_i$ is the equilibrium window.

Around the equilibrium, the buffer process under RED evolves according to:

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

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 around the equilibrium):

$$\begin{aligned} \dot{b}_l(t) &= \sum_i R_{li} \frac{w_i(t - \tau_{li}^f)}{\tau_i} \\ &\quad - \sum_k \sum_i R_{li} R_{ki} \frac{w_i^*}{\tau_i^2 c_k} \, b_k(t - \tau_{li}^f) \end{aligned}$$

The second term above is ignored if we have neglected or assumed constant the queueing delay in 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 is 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 backlog in the path of source i reduces the *rate* at which source i packets arrive at link l, decreasing $b_l(t)$.

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

$$\begin{aligned} w(s) &= -(sI + D_1)^{-1} D_2 R_b^T(s) p(s) \\ p(s) &= (sI + D_3)^{-1} D_4 b(s) \\ b(s) &= (sI + R_f(s) D_5 R^T D_6)^{-1} R_f(s) D_7 w(s) \end{aligned}$$

where the diagonal matrices are $D_1 = \text{diag}\left(\frac{q_i^* w_i^*}{\tau_i}\right)$, $D_2 = \text{diag}\left(\frac{1}{\tau_i q_i^*}\right)$, $D_3 = \text{diag}\ (\alpha_l c_l)$, $D_4 = \text{diag}\ (\alpha_l c_l \rho_l)$, $D_5 = \text{diag}\left(\frac{w_i^*}{\tau_i^2}\right)$, $D_6 = \text{diag}\left(\frac{1}{c_l}\right)$, $D_7 = \text{diag}\left(\frac{1}{\tau_i}\right)$, and $R_f(s)$ and $R_b(s)$ are delayed forward and backward routing matrices, defined as:

$$[R_f(s)]_{li} = \begin{cases} e^{-\tau_{li}^f s} & \text{if } l \in L_i \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$[R_b(s)]_{li} = \begin{cases} e^{-\tau_{li}^b s} & \text{if } l \in L_i \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

This model generalizes the single-link identical-source model of [4] to multiple links with heterogeneous sources.

C. Validation and stability region

We present a series of experiments to validate our linear model when the system is stable or barely unstable, and to illustrate numerically the stability region.

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 (11) below). 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 duality model [5]. 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 $f_m(N, c)$ from the analytical model. For all experiments, we have fixed the parameters at $\alpha = 10^{-4}$, $\rho = 0.1/(540 - 40) = 0.0002$, and $\beta = 0.5$.

We repeat these experiments in *ns-2*, using persistent FTP sources and RED with ECN marking. The RED parameters are $(0.1$, 40pkts, 540pkts, $10^{-4})$, corresponding to the α and ρ values in the model. 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 4(a) plots the critical delay $d_{ns}(N, c)$ from *ns-2* simulations versus the critical delay $d_m(N, c)$ computed from the linear model. Each data point corresponds to a particular (N, c) pair. The dashed line is where all points should lie if the linear model agrees perfectly with the simulation. Figure 4(b) gives the corresponding plot for critical frequencies $f_{ns}(N, c)$ versus $f_m(N, c)$. The agreement between model and simulation seems quite reasonable (recall that delay values have a resolution of 5ms).

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

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

Then the linearized model is (variables now are perturbations)

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

where $f_l'(y_l^*)$ is the derivative of f_l evaluated at equilibrium. Also shown in Figure 4(b) are critical frequency predicted from this static-link model (with $f_l'(y_l^*) = \rho$, this does not affect the critical frequency), using the same Nyquist plot method described above. It shows that queue dynamics is significant at the time-scale of interest.

Figure 4(c) illustrates the stability region implied by the linear model. For each N, it plots the 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. Intuitively, 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.

D. Stability: single-link heterogeneous sources

We now characterize the stability region in the case of a single link with N *heterogeneous* sources. Specializing the linear model of the last subsection to this case, writing forward delay as a fraction $\beta \in (0,1)$ of round trip time, $\tau_i^f = \beta\tau_i$, and dropping link subscript l, we obtain the loop gain as

$$\begin{aligned} L(s) \;=\; & \sum_i \frac{1}{\tau_i p^*(\tau_i s + p^* w_i^*)} \cdot \frac{\alpha c \rho}{s + \alpha c} \cdot \\ & \frac{1}{s + \frac{1}{c}\sum_n \frac{x_n^*}{\tau_n} e^{-\beta\tau_n s}} \cdot e^{-\tau_i s} \qquad (11) \end{aligned}$$

The first term on the right-hand side describes TCP dynamics, the second term RED averaging, the third term buffer process, and the last term network delay. The special case where all sources have identical round trip times, $\tau_i = \tau$, and forward delays are zero, $\beta = 0$, is analyzed in [4]. They provide sufficient conditions for closed-loop stability and use them to tune RED parameters α and ρ.

We start with a lemma that collects some equilibrium properties we use below. It can be proved directly from the fixed point of (5-8); or see [5]. Let $\overline{\tau} := \max_i \tau_i$, $\underline{\tau} := \min_i \tau_i$ and $\hat{\tau} := \left(\sum_i \frac{1}{\tau_i}\right)^{-1}$.

Lemma 1: Let p^* be the equilibrium loss probability, w_i^* and x_i^* be the equilibrium window and rate respectively. Then $p^* = 2/(2 + (c\hat{\tau})^2)$, $w_i^* = c\hat{\tau}$ for all sources i, $x_i^* = w_i^*/\tau_i$ and $\sum_i x_i^*/c = 1$.

Let

$$\theta_0 \;:=\; \arctan\frac{\pi(1-\beta)}{2\beta} \;\in\; \left(0, \frac{\pi}{2}\right) \qquad (12)$$

$$h_{red}(v,\theta) \;:=\; \frac{e^{-j(v+\theta)}}{v(jv + \alpha c\underline{\tau})(jv + p^* w_1^*)}, \quad 0 \le \theta \le \pi - \theta_0 \qquad (13)$$

We will show that the Nyquist plot of $h_{red}(v,\theta)$ at $\theta = \pi - \theta_0$ determines the stability of TCP/RED. Let v_0 be the angle at which the phase of $h_{red}(v, \pi - \theta_0) = -\pi$.

Theorem 2: Assume the equilibrium window $w_i^* \ge \sqrt{2}$. Then the closed-loop system described by (11) is stable if

$$c^3\overline{\tau}^3 \cdot |h_{red}(v_0, \pi - \theta_0)| \;\le\; \frac{1-\beta}{\alpha\rho}$$

where $\overline{\tau} := \max_i \tau_i$.

Proof (Sketch). The closed-loop system is stable if $L(s)$ does not pass through $(-1, 0)$ in the complex plane as s takes value in the right-half plane. To show this, re-write (11) as

$$L(s) \;=\; \frac{c^2\alpha\rho}{p^*} \sum_i \frac{w_i^*/\tau_i}{c} \frac{z_i(s)}{w_i^*}$$

where

$$z_i(s) \;=\; \frac{e^{-\tau_i s}}{(s + \alpha c)(\tau_i s + p^* w_i^*)} \cdot \frac{1}{(s + \sum_n \frac{x_n^*}{c\tau_n} e^{-\beta\tau_n s})} \qquad (14)$$

Lemma 1 implies that

$$L(s) \;=\; \frac{c\alpha\rho}{\hat{\tau} p^*} \sum_i \frac{x_i^*}{c} z_i(s) \qquad (15)$$

i.e., $L(s)$ lies in the convex hull defined by the N points $z_i(s)$ in the complex plane. We will prove that this convex hull is bounded away from $(-1, 0)$, in two steps.

First, by bounding the magnitude and phase of the last term in (14), we can show that

$$L(j\omega) \;\in\; \frac{\overline{\tau}^2 c\alpha\rho}{(1-\beta)\hat{\tau} p^*} \cdot \text{co}\,\{h_{red}(v,\theta) \;:\; v \ge 0, 0 \le \theta \le \pi - \theta_0\} \qquad (16)$$

i.e., we have embedded the convex hull defined by (14) in the larger convex hull of $h_{red}(v,\theta)$ defined in (13).

We then show that the hypothesis of the theorem bounds this set away from $(-1, 0)$. Note that the trajectory of $h_{red}(v,\theta)$ is the curve

$$C(v) \;:=\; \frac{e^{-jv}}{v(jv + \alpha c\underline{\tau})(jv + p^* w_1^*)}$$

rotated by θ in the negative direction. Since the magnitude $|C(v)|$ is decreasing in v, the left boundary of the convex hull in (16) is determined by $h_{red}(v,\theta)$ at $\theta = \pi - \theta_0$. By examining the curvature of the plane curve $h_{red}(v, \pi - \theta_0)$ as v varies from 0 to $+\infty$ [11], it can be shown that the boundary of this set crosses the real axis at $|h_{red}(v_0, \pi - \theta_0)|$. Hence $L(j\omega)$ will not pass through $(-1, 0)$ if

$$\frac{\overline{\tau}^2 c\alpha\rho}{(1-\beta)\hat{\tau} p^*} \cdot |h_{red}(v_0, \pi - \theta_0)| \;<\; 1$$

This condition can then be shown to hold under the hypothesis of the theorem. ■

The idea of bounding the Nyquist plot of $L(j\omega)$ in the convex hull of $h_{red}(v,\theta)$ is inspired by the proof in [12] of a different algorithm.

IV. RED PARAMETER SETTING

The parameters $\alpha \in (0,1]$ and $\rho > 0$ on the RHS of the stability condition in Theorem 2 are the weight in RED's exponential averaging of queue length and the slope of the marking probability, respectively. For stability, their *product* should be small. A small α leads to a sluggish response because information on instantaneous queue length is incorporated into the feedback very slowly. A small ρ implies a large delay: it can be shown that the average queue length r in equilibrium is $r = \underline{b}_l + 2/\rho(2 + (c\hat{\tau})^2)$. Indeed, when the sources are identical, $\tau = \overline{\tau} = \underline{\tau} = \hat{\tau}N$. The LHS of the stability condition becomes $\frac{c^3\tau^3}{2N}|h|$. It suggests that TCP/RED becomes unstable when delay τ or capacity c increases, confirming the simulation results in the last subsection. Roughly, when c doubles, the equilibrium rate doubles, and hence window is halved with twice the magnitude at twice the frequency, resulting in a quadratic increase in control gain and pushing the system into instability.

It is suggested in [15] that the RED parameter max_p be dynamically adjusted: reduce max_p as N decreases and raise it otherwise. Raising max_p, or reducing max_th - min_th, is equivalent to increasing ρ (= max_p/(max_th-min_th)) in the direction consistent with the stability condition in Theorem 2. Theorem 2 sets an upper bound on ρ, given N, c, τ (and α), and hence a lower bound on equilibrium queue length, to ensure stability. Adapting RED parameters *cannot* prevent the inevitable choice between stability and performance: either ρ is set small to stabilize the queue, around a large value, or, alternatively, it is set large to reduce the queue, at the expense of violent oscillation.

The same stability analysis can also be applied to other AQMs, such as Virtual Queue [16], [17], [18] and REM/PI [19], [6], and clarifies the role of AQM. The stability proof relies on bounding a set of the form

$$K \cdot \text{co}\{h(v,\theta)\}$$

to the right of $(-1,0)$. The gain K and the trajectory h depend on TCP as well as AQM. For instance, for the case of a single link with capacity c shared by N identical sources with delay τ, TCP and network delay contribute a factor

$$h_{tcp} = \frac{e^{-jv}}{jv + p^* w_1^*}$$

to the trajectory h and a factor

$$K_{tcp} = \frac{c^2\tau^2}{2N} \qquad (17)$$

to the gain K, assuming equilibrium window is large so that $p^* = 2/w_i^2 = 2N/c\tau$. This high gain (17) is mainly responsible for instability at high delay, high capacity or low load. AQM compensates for these effects by shaping h and reducing K. With RED, for instance,

$$\begin{aligned} h(v,\theta) &= \frac{1}{jv + \alpha c\tau}\,\frac{e^{-j\theta}}{v} \cdot h_{tcp} \\ K &= \frac{c\tau\alpha\rho}{1-\beta} \cdot K_{tcp} \end{aligned}$$

The first term in h is due to RED averaging, the second term is due to queue dynamics that also bounds $\theta \le \pi - \theta_0$. Hence both the queue and RED add phase lag to h. More importantly, RED adds another $c\tau$ to the gain K, necessitating a small $\alpha\rho$ for stability and leading to sluggish response and large equilibrium queue. The factor $\tau/(1-\beta)$ in K comes from the queue.

V. A SCALABLE CONTROL

Measurements in [8] shows that delay is still large in the current Internet (85% of the round trip time measurements range in 15–500 ms). The results in the previous sections suggest that the current protocol may be ill-suited in such an environment. Moreover, the situation will become worse in the future where network capacity will be large. It also seems difficult to design effective AQM to compensate for the high gain introduced by TCP. In this section we describe a protocol, developed in [7], that can be implemented in a decentralized way by sources and links, and that is scalable: it maintains linear stability for *arbitrary* delay, capacity, load and routing. Moreover, it achieves high network utilization in equilibrium with small queue. These requirements impose certain constraints on the linearized dynamics: integration at links, and conditions on the gain at sources and links.

A. Algorithm

We summarize the congestion control algorithm of [7]. It consists of a static source algorithm and a first-order dynamic link algorithm. The key idea is to compensate for delay at sources by scaling down the gain on rates by their individual round trip times, and to compensate for loop gain introduced by capacity and routing by scaling down the control gain at links by their capacities and scaling it up at sources by their current rates. In other words, a source reacts more slowly if its round trip delay is large or if its rate is small; a link updates its congestion measure (called 'price') more slowly if it has a larger capacity. Note that network delay is the *only* open-loop parameter not under our control, and it *should* set the time-scale of the system response.

Consider the network model described in Section III-A. Let $p_l(t)$ be the price at link l at time t and c_l be a *virtual* capacity that is strictly less than real link capacity. Each link l adjusts its price using the aggregate input rate $y_l(t) = \sum_s R_{ls}x_s(t)$:

$$\dot{p}_l(t) = \begin{cases} \frac{y_l - c_l}{c_l} & \text{if } p_l > 0; \\ \max\{0, \frac{y_l - c_l}{c_l}\} & \text{if } p_l(t) = 0 \end{cases} \qquad (18)$$

Therefore prices integrate excess capacity in a normalized way, and are saturated to be always non-negative. At equilibrium, bottlenecks with nonzero price will have $y_l^* = c_l$, giving high utilization. Non-bottlenecks with $y_l^* < c_l$ will have zero price. Since c_l is less than real capacity, queue is negligible in equilibrium. If c_l were the real capacity, $p_l(t)$ would be the real queueing delay, a congestion signal used in TCP Vegas [9].

Let $x_i(t)$ be the rate of source i at time t, τ_i its round trip time (assumed constant), and M_i the number of congested links in its path (or an upper bound). Given aggregate price $q_i(t) = \sum_l R_{li} p_l(t)$, source i sets its rate to be exponential in $q_i(t)$:

$$x_i(t) = x_{\max,i}\, e^{-\frac{\alpha_i q_i(t)}{M_i \tau_i}} \tag{19}$$

Here $x_{\max,i}$ is a maximum rate parameter, and $\alpha \in (0,1)$. The utility function corresponding to the source control is

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

Suppose the routing matrix R has full row rank. Then there is a unique equilibrium rate and price vector (x^*, p^*). The linearized system around the equilibrium is described by (variables now denote perturbation):

$$\dot{p}_l(t) = \frac{y_l(t)}{c_l}, \quad \text{for all } l \tag{20}$$

$$\dot{x}_i(t) = -\frac{\alpha_i x_i^*}{M_i \tau_i}, \quad \text{for all } s \tag{21}$$

where the source rates $x(t)$ and link prices $p(t)$ are interconnected by the delayed routing matrices defined in (9–10).

The following theorem, proved in [7], guarantees the stability of the algorithm when the network scales up arbitrarily in delay, capacity and load.

Theorem 3 ([7]) Suppose all links included in R are bottlenecks, i.e., $c = Rx^*$ in equilibrium, and R has full row rank. Then the closed-loop system described by (20–21) and (9–10) is linearly stable for *arbitrary* delays τ_i and link capacities c_l.

B. Implementation and performance

A simple way to implement the link algorithm (18) is to maintain a "virtual queue" counter that is incremented with received packets and decremented at the virtual capacity rate. Then prices are obtained by dividing this counter by the virtual capacity.

Sources measure their round-trip time τ_i. Since the target state is with empty queues, at equilibrium, τ_i is the propagation delay; it is therefore recommended to use an estimate of d_i (typically the minimum observed round-trip time) in source update (19), as is done in Vegas. This avoids the possibility that temporary excursions of the real queue 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 M_i of bottlenecks. To communicate q_i, the technique of random exponential marking [19] can be used. Here, a packet would be marked at each link l with probability $1 - \phi^{-p_l}$, $\phi > 1$. 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. This requires the knowledge of ϕ which is presumably a global constant set a priori. Regarding M_i, in the simplest implementation one would simply employ an upper bound.

A preliminary packet level implementation is done using parallel simulator Parsec [20]. 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 for a wide range of N, c, d with the same parameters as those in Figure 1. Figure 5 shows the *individual* window and queue: as expected, both the individual window and queue converge regardless of delay. Longer delay sets a larger time-scale for the closed loop behavior.

At this point one might wonder whether the stability of this protocol has been obtained at the expense of performance, i.e. by making the time-response very slow. However a comparison with Figure 1 shows this is not the case. Indeed, both Reno and the new protocol require here about 50 round-trip times to reach steady state. For instance, in the case of a 200ms delay, it takes about 10 seconds for Reno to reach its limit cycle, and the same amount of time for our protocol to reach equilibrium with an empty queue.

VI. Conclusion

We have presented simulation results to demonstrate that it is protocol stability more than other factors that determines the dynamics of TCP/RED. We have developed a multi-link multi-source model that can be used to study the stability of general TCP/AQM. We have presented a sufficient stability condition for the case of a single link with heterogeneous sources, and illustrated the form of TCP/RED's stability region. It implies that TCP/RED becomes unstable when the network scales up in delay or capacity. Our analysis indicates the role, and the difficulty, of RED in stabilizing TCP. We have described a new distributed algorithm that maintains local stability for arbitrary delay, capacity, load and routing, and presented preliminary simulation results to illustrate its behavior. As future work, we are developing a prototype of this algorithm and conduct a comprehensive performance comparison with the current protocol.

References

[1] Martin May, Thomas Bonald, and Jean-Chrysostome Bolot, "Analytic evaluation of RED performance," in *Proceedings of IEEE Infocom*, March 2000.

[2] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for web traffic," in *Proceedings of ACM Sigcomm*, 2000.

[3] Victor Firoiu and Marty Borden, "A study of active queue management for congestion control," in *Proceedings of IEEE Infocom*, March 2000.

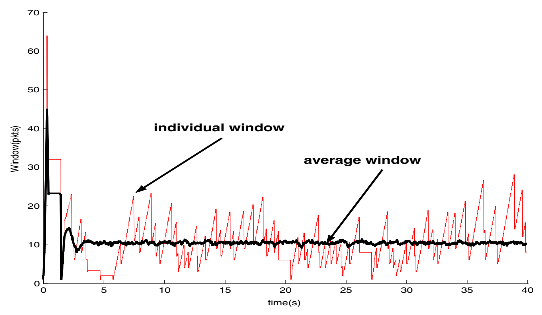
[4] Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong, "A control theoretic analysis of RED," in *Proceedings of IEEE Infocom*, April 2001, `http://www-net.cs.umass.edu/papers/papers.html`.

[5] 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`.

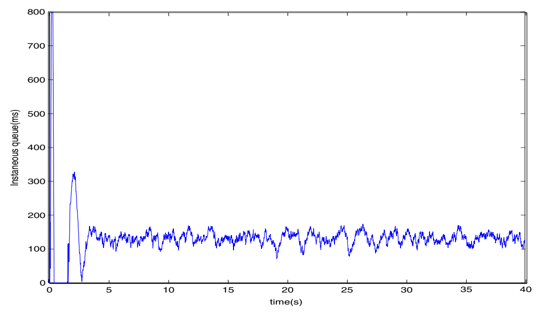
[6] Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of IEEE Infocom*, April 2001, `http://www-net.cs.umass.edu/papers/papers.html`.

[7] 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`.

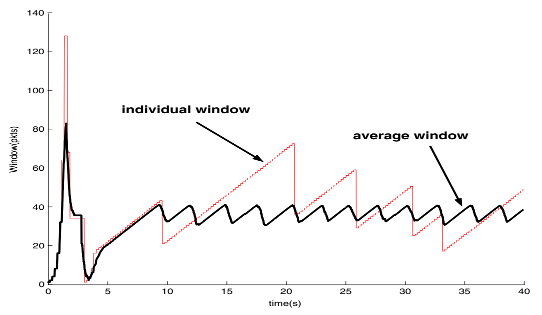
[8] Mark Allman, "A web server's view of the transport layer," *ACM Computer Communication Review*, vol. 30, no. 5, October 2000.
[9] Steven H. Low, Larry Peterson, and Limin Wang, "Understanding Vegas: a duality model," *J. of ACM, to appear*, 2002, `http://netlab.caltech.edu/pub.html`.
[10] Lawrence S. Brakmo and Larry L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–80, October 1995, `http://cs.princeton.edu/nsg/papers/jsac-vegas.ps`.
[11] Barrett O'Neill, *Elementary Differential Geometry*, Academic Press, 1966.
[12] Glenn Vinnicombe, "On the stability of end-to-end congestion control for the Internet," Tech. Rep., Cambridge Univiversity, CUED/F-INFENG/TR.398, December 2000.
[13] R. Johari and D Tan, "End-to-end congestion control for the internet: Delays and stability," Tech. Rep., 2000, Cambridge Univ. Statistical Laboratory Research Report 2000-2.
[14] L. Massoulie, "Stability of distributed congestion control with heterogeneous feedback delays," Tech. Rep., Microsoft Research, Cambridge UK, TR 2000-111, 2000.
[15] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self–configuring RED gateway," in *Proceedings of INFOCOM'99*, March 1999, `http://www.eecs.umich.edu/~wuchang/work/infocom99.ps.Z`.
[16] R. J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, 1999.
[17] 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`.
[18] 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`.
[19] 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`.
[20] "Parallel simulation environment for complex systems," `http://pcl.cs.ucla.edu/projects/parsec/`.



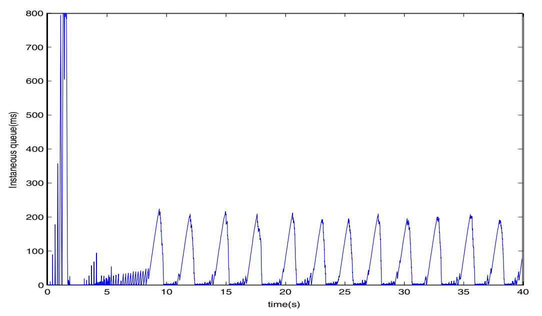
(a) Window (delay = 40ms)



(b) Queue (delay = 40ms)

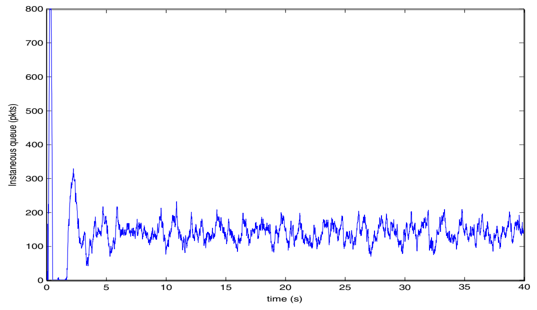


(c) Window (delay = 200ms)

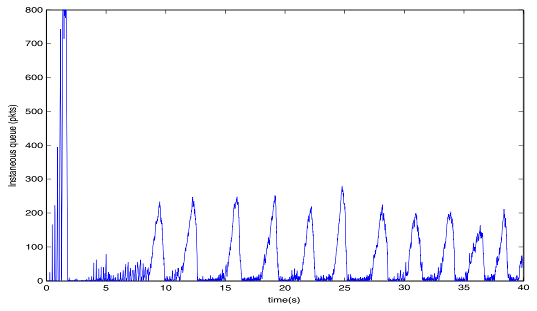


(d) Queue (delay = 200ms)

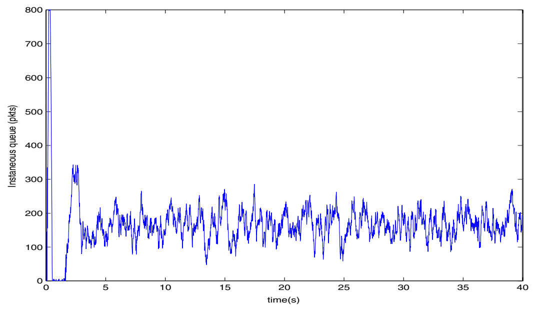
Fig. 1. Window and queue traces without noise traffic. Simulation parameters: 50 sources, capacity = 9 pkts/ms, RED = (0.1, 50, 550, 10^{-4}), marking with 'byte' mode; two-way traffic.



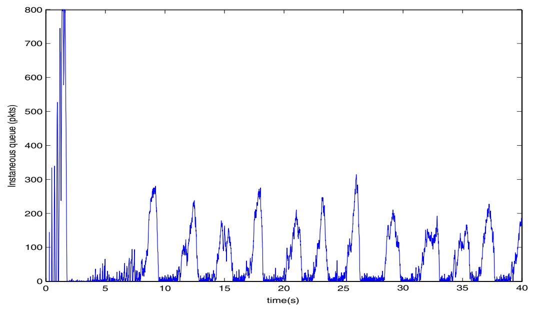
(a) Queue (delay = 40ms, 10% noise)



(b) Queue (delay = 200ms, 10% noise)

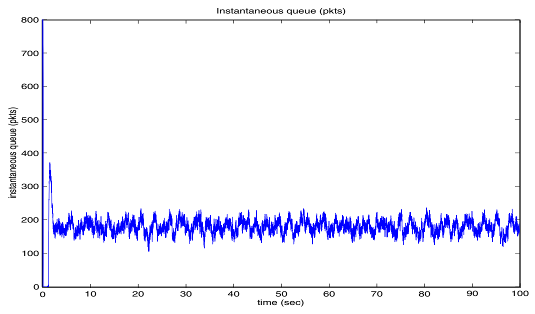


(c) Queue (delay = 40ms, 30% noise)

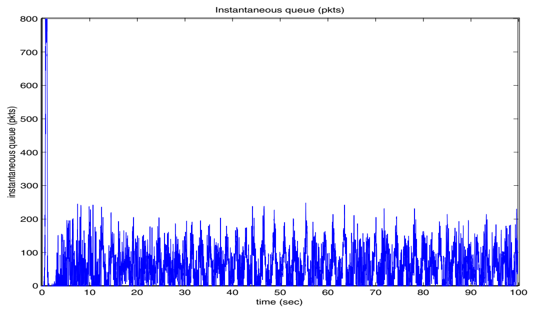


(d) Queue (delay = 200ms, 30% noise)

Fig. 2. Queue traces with noise traffic. Simulation parameters: 50 sources, capacity = 9 pkts/ms, RED = (0.1, 50, 550, 10^{-4}), marking with 'byte' mode; two-way traffic.

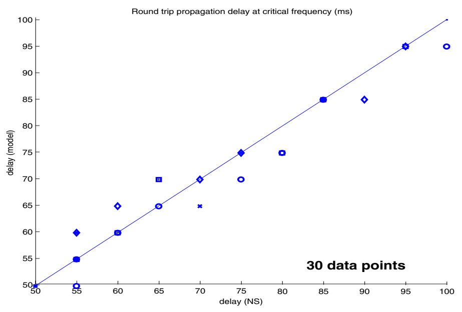


(a) Queue (delays from 12ms to 19ms)

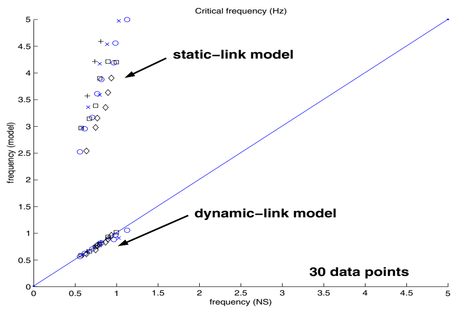


(b) Queue (delays from 160ms to 254ms)

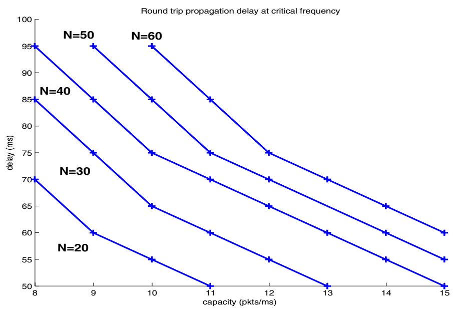
Fig. 3. Queue traces with heterogeneous delays. Simulation parameters: 50 sources, capacity = 9 pkts/ms, RED = (0.1, 50, 550, 10^{-4}), marking with 'byte' mode; one-way traffic.



(a) Critical delay (ms)

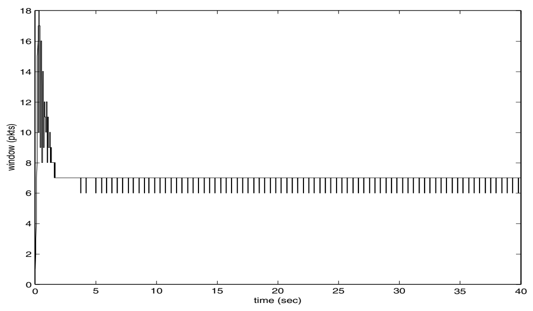


(b) Critical frequency (Hz)

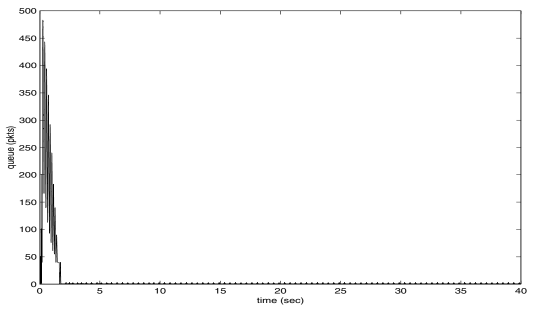


(c) Stability region

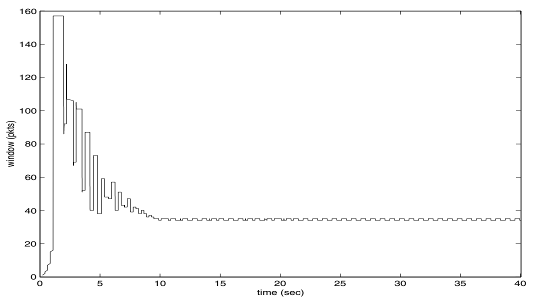
Fig. 4. Validation and stability region. For each N, the region above the curve is unstable and that below is stable.



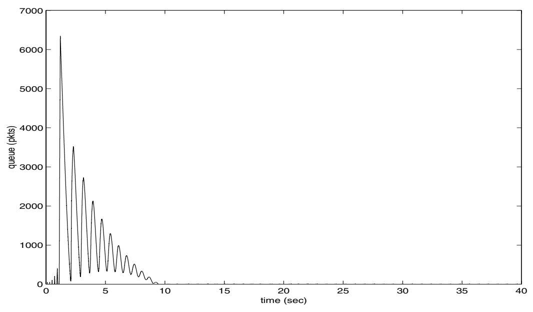
(a) Individual window (delay = 40ms)



(b) Queue (delay = 40ms)



(c) Individual window (delay = 200ms)



(d) Queue (delay = 200ms)

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