On TCP Reaction to Explicit Congestion Notification

Minseok Kwon and Sonia Fahmy
Department of Computer Sciences, Purdue University
250 N. University St.
West Lafayette, IN 47907–2066, USA
Tel: +1 (765) 494-6183, Fax: +1 (765) 494-0739
e-mail: {kwonm,fahmy}@cs.purdue.edu

Abstract

We investigate the behavior of a new response strategy to TCP Explicit Congestion Notification (ECN). The new strategy is more aggressive in the short term, but preserves TCP long term behavior– without modifying the router ECN marking rate. A more aggressive short term behavior gives incentives for hosts to become ECN-compliant. ECN serves as an early warning sign in this case. Our analysis demonstrates the effectiveness of the new TCP ECN behavior. Simulation results with short/long lived FTP, UDP and HTTP connections, multiple bottleneck configurations, and various TCP flavors and parameters, demonstrate higher throughput and reduced oscillations with the new response strategy.

Keywords: TCP congestion control, explicit congestion notification (ECN), random early detection (RED), active queue management, additive increase multiplicative decrease (AIMD)

I. Introduction

Congestion avoidance mechanisms have been a topic of active research since the early 1980s [19], [8], [5], [17], [30]. In the TCP congestion avoidance algorithm, the congestion window is linearly increased by one segment (additive increase) every round-trip time (RTT), and is halved (multiplicative decrease) in response to a single packet drop. Hence, the increase increment is 1 and the decrease fraction is $\frac{1}{2}$, or what we refer to as AIMD (α,β)=AIMD (1, 0.5). Recently, the relationship between the two parameters α and β has been studied in [13], [37].

Active Queue Management (AQM) at network routers has also been extensively studied in the last few years. Floyd and Jacobson proposed the Random Early Detection (RED) scheme for Internet routers in 1993 [15]. RED drops packets probabilistically when the average buffer occupancy lies between two thresholds. RED drops all packets when the maximum threshold is exceeded. Significant research has focused on refining such intelligent packet dropping or marking mechanisms [28], [32], [3], [25]. The Explicit Congestion Notification (ECN) option entails that RED routers mark (instead of drop) packets when buffer occupancy lies between the two thresholds [10]. Although the source response to ECN should match its response to packet drop, Internet standards indicate that this matching requirement is a long term one. This allows the possibility of less aggressive reduction in the short term. We have explored this idea in [26], and we further analyze it in this paper.

We react to the receipt of ECN with a relatively small window decrease, but increase less aggressively for a subsequent period of time. We reset the increase and decrease parameters back to (1, 0.5) in the event of retransmission timeouts or duplicate acknowledgments (ACKs) following the ECN decrease. The smooth response to ECN is consistent with the motivation behind ECN: to serve as an early warning for congestion. The improved performance we will demonstrate in this paper gives incentives for hosts to become ECN-compliant. The AIMD parameters for ECN are selected to preserve the long term TCP behavior, without requiring modifications to the router marking rate. The window size and sending rate do not oscillate as much as they do with ECN with (1, 0.5), thus increasing overall throughput and reducing delay variance. Note that we investigate different increase and decrease parameters only as a response to ECN, since (1, 0.5) are standardized for TCP [2]. The precise response to ECN, however, is not dictated by [34].

The remainder of this paper is organized as follows. Section II gives basic background on TCP congestion control, active queue management, and explicit congestion indication in the Internet. Section III discusses related work. Section IV explains and models our approach. Section V simulates our approach in a number of network configurations with various parameters. Finally, Section VI summarizes our conclusions and discusses future work.

II. Background

We first summarize the TCP congestion control mechanisms, and then discuss active queue management and explicit congestion notification.

A. TCP Congestion Control

A TCP connection begins in the "slow start" phase [19]. The sender initially sets its congestion window, *cwnd*, to 1 or 2 segments [2]. For each ACK received, *cwnd* is increased by one segment. This results in an exponential increase of *cwnd* over round trips. TCP uses the slow start threshold, *ssthresh*, to indicate the window size appropriate for the current network load. The slow start phase continues as long as *cwnd* is less than *ssthresh*. As soon as it exceeds *ssthresh*, TCP goes into "congestion avoidance." In congestion avoidance, for each ACK received, *cwnd* is increased by 1/*cwnd* segments, which is approximately equivalent to increasing *cwnd* by one segment every round trip (an additive increase). The TCP sender assumes network congestion when it times out waiting for an ACK, or when it receives 3 duplicate ACKs. *ssthresh* is halved (a multiplicative decrease). The Additive Increase Multiplicative Decrease (AIMD) system has been shown to be stable under certain assumptions [6], [20], [21].

B. Random Early Detection

RED maintains a long term average of the queue length (buffer occupancy) of a router using a low-pass filter. If this average queue length falls below a certain minimum threshold, all packets are admitted into the queue. If the average queue length exceeds a certain maximum threshold, all incoming packets are dropped. When the queue length lies between the minimum and maximum thresholds, incoming packets are dropped/marked with a linearly increasing

probability up to a maximum probability value, p_{max}. RED includes an option known as the "gentle" variant. With gentle RED, the packet drop/mark probability varies linearly from p_{max} to 1 as the average queue size varies from th_{max} to twice th_{max}.

C. The ECN Mechanism

The Explicit Congestion Notification (ECN) option [10], [34] allows active queue management mechanisms such as RED to probabilistically mark (rather than drop) packets, when the average queue length lies between the two thresholds. This is only possible if both the sender and receiver are ECN-capable (determined at connection setup time). In this case, the receiver echoes back to the sender the fact that some of its packets were marked. The sender thus determines that the network is approaching a congested state. The sender should reduce its congestion window as if the packet was dropped, but need not reduce it drastically (e.g., set it to one or two segments) [34]. The sender should only react once per RTT to congestion indications. With ECN, both gentle RED and vanilla RED mark (not drop) packets when the average queue size lies between the two thresholds. Gentle RED still drops less aggressively than vanilla RED between the maximum threshold and twice the maximum threshold. The primary advantage of ECN is that TCP does not have to wait for a timeout and some packet drops can be avoided.

III. Related Work

A number of studies have investigated RED and ECN performance. Variations of RED include flow RED [28], stabilizing RED [32], and BLUE [1]. The effect of RED parameter values on web traffic is studied in [7], and the effect of marking from the front of the queue is investigated in [29]. A performance study of ECN with real traffic is presented in [35]. Ott [31] investigates ECN with various response algorithms. The study considers environments where non-ECN compatibility is not required, and the marking rate can be increased. Such integrated solutions that modify router marking also include REM [3], AVQ [25] and the PI controller [18].

Design of AIMD algorithms has also been an active research area. The additive increase parameter α and multiplicative decrease parameter β have been studied since the 1980s [6], [20]. Recently, Yang and Lam [37] derived the relationship between the two parameters necessary for "TCP-friendliness" (compatibility with TCP). They recommend 0.875 ($cwnd = 0.875 \times cwnd$) for multiplicative decrease, and 0.3125 for additive increase, as a response to loss (*not* as a response to ECN). In [14], [13], the authors propose equation-based congestion control (based on TCP models in [33]), and compare it to TCP using a number of AIMD parameters. The authors use a slightly different relationship between the additive increase and multiplicative decrease parameters from that in [37]: they use $\alpha = \frac{3(1-\beta)}{1+\beta}$ which gives smaller α values than those in [37], for the range of β values we are interested in. As in [37], the authors of [13] consider AIMD parameters in general, and not in the context of ECN as we do in this paper. Another class of AIMD algorithms, binomial algorithms, is studied in [4] for streaming applications. XCP generalizes ECN to achieve efficient and stable congestion control for high bandwidth-delay product networks [22].

One study that investigates the response to ECN without changing the marking rate is [16]. The authors propose an algorithm to react to ECN, and, at the same time, remove TCP's bias to short RTT connections. They modify the window increase slope to be proportional to RTT^2 (hence the *rate* increase slope is proportional to RTT). The main problem with their approach is that RTT bias elimination should be independent of ECN marking. Further, their algorithm is not TCP-friendly, and is highly sensitive to the RED maximum drop probability. We have addressed these issues in [26], and we analyze our proposed ECN response strategy in this paper.

IV. Parameterized ECN Response

In this section, we discuss our ECN response strategy, and model it in conjunction with AQM routers such as RED.

A. New ECN Response

The primary objective of our strategy is to modify ECN response, in order to increase throughput, reduce rate fluctuations, and reduce delay variance. The pseudo-code is outlined in Figure 1. We use two parameters α_{ECN} and β_{ECN} to denote the required increase and decrease parameters, respectively. Our performance study (Section V) indicates that (0.2,0.875) are good choices for (α_{ECN},β_{ECN}). We reduce the congestion window and slow start

1) **When an ACK with ECN is received:**
   Reduce *ssthresh* and *cwnd* by β_{ECN}
   Set *IncreaseSlope* to α_{ECN}
2) **On a timeout or 3 duplicate ACKs:**
   Reduce *ssthresh* and *cwnd* normally
   Reset *IncreaseSlope* to 1
3) **Congestion avoidance:**
   ***cwnd* = *cwnd* +** $\frac{IncreaseSlope}{cwnd}$

Fig. 1. Pseudo-code for parameterized response to ECN marks

threshold by β_{ECN} in response to an ECN-marked packet. We then use a modified increase slope *IncreaseSlope* in the ensuing congestion avoidance phase. The increase slope is set to α_{ECN} on the receipt of ECN-marked packets, and reset to 1 on a timeout or receipt of 3 duplicate ACKs. Note that congestion window increases and decreases follow TCP for all congestion indications other than ECN. Thus, β_{ECN} need not be reset with timeouts or 3 duplicate ACKs. The essential idea is to match the use of the increase parameter with the corresponding decrease parameter that was previously applied. This less aggressive window decrease on ECN is more consistent with the use of ECN as an early warning sign. It also gives incentives for hosts to become ECN-capable. We call this approach ECN(α, β).

B. ECN(α, β) Sending Rate

We extend the TCP sending rate models developed in [27], [33] to include the new ECN response ECN(α_{ECN},β_{ECN}). We employ the same assumptions as Padhye *et al.* [33]. We have selected Padhye's model because of its simplicity, and the fact that it is one of the best available approximations for TCP-Reno performance. Observe, however, that the model has several weaknesses. First, the model does not include the TCP fast recovery algorithm. Second, the model assumes a simple loss model, where congestion indications in one round are independent of congestion indications in other rounds. A round starts with packet transmissions corresponding to the current TCP congestion window size, and ends with the first ACK for one of the packets sent. Third, RTT is independent of congestion window size. Thus, the duration of a round is equal to one RTT in this model.

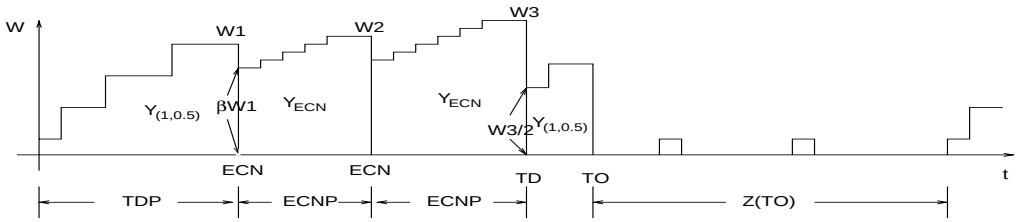


Fig. 2. Evolution of the window size with (α_{ECN},β_{ECN}) as ECN response parameters

With the new ECN behavior, there are three congestion indication types: ECN, TD (Triple-Duplicate ACKs) and TO (Time-Outs). Figure 2 depicts a sample evolution of the congestion window size using (1, 0.5) for triple duplicate acknowledgments, and (α_{ECN},β_{ECN}) for ECN. We define a TDP as a period starting from receipt of a Triple-Duplicate ACK (TD) and lasting until the next congestion indication. Similarly, we define an ECNP as a period starting from ECN and lasting until the next congestion indication. A timeout sequence is defined as a period starting from a timeout and lasting until the sender receives an acknowledgment. Let $E[Y_{(1,0.5)}]$ and $E[Y_{ECN}]$ be the expected number of packets sent during a TDP and an ECNP, respectively. Similarly, $E[A_{(1,0.5)}]$ and $E[A_{ECN}]$ are the mean durations of the TDP and ECNP, respectively. We also define $E[W_{(1,0.5)}]$ and $E[W_{ECN}]$ as the expected window size at the end of TDP and ECNP, respectively.

We introduce the term ρ to denote the fraction of ECN indications among the sum of ECNs and TDs. More precisely, $\rho = \frac{num(ECN)}{num(TD)+num(ECN)}$, where $num(x)$ denotes the number of congestion indications using method x. Note that ECNs are only counted once per round-trip time. Let p be the total congestion indication probability, defined as $p = \frac{num(ECN)+num(TD)+num(TO)}{M}$, where M is the total number of packets sent. We also define $p_t = \frac{num(TD)}{M_t}$ and

$p_e = \frac{num(ECN)}{M_e}$ to denote the congestion indication probabilities using TD and ECN, respectively. Here, M_t is the total number of packets sent during all TDPs, and M_e is the total number of packets sent during all ECNPs.

As in [33], let b be the number of packets acknowledged by a single ACK. Let $\mathcal{Q}$ be the probability that a loss indication ending a TDP is a timeout. Let $E[R]$ denote the packets sent during a timeout sequence, and $E[Z^{TO}]$ denote the duration of the timeout sequence. Both $E[R]$ and $E[Z^{TO}]$ can be computed according to the probability distribution of the number of timeouts in a timeout sequence. According to [33], $E[R] = \frac{1}{1-p}$ and $E[Z^{TO}] = T_0 \frac{f(p)}{1-p}$ where T_0 is the TCP timeout period, and $f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6$. Also, $\mathcal{Q} \approx min(1, \frac{3}{E[W]})$, where $E[W]$ in our case becomes $(1-\rho)E[W_{(1,0.5)}] + \rho E[W_{ECN}]$. The sending rate B for the ECN(α,β) strategy can thus be modeled as:

$$B = \frac{(1-\rho)E[Y_{(1,0.5)}] + \rho E[Y_{ECN}] + \mathcal{Q} \times E[R]}{(1-\rho)E[A_{(1,0.5)}] + \rho E[A_{ECN}] + \mathcal{Q} \times E[Z^{TO}]} \tag{1}$$

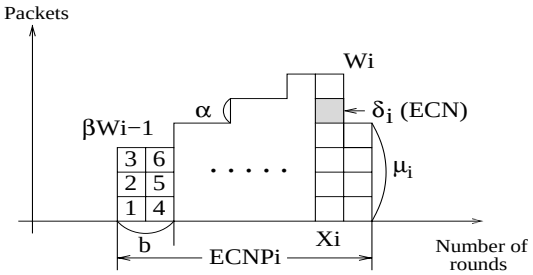


Fig. 3. Packets sent during an ECNP

We first compute $E[Y_{ECN}]$ and $E[A_{ECN}]$. As illustrated in Figure 3, $Y_i = \delta_i + W_i - 1$ packets are sent during the i^{th} ECNP. Here, Y_i is the number of packets sent in the i^{th} ECNP, δ_i is the first ECN-marked packet, and W_i is the window size at the end of the i^{th} ECNP. It follows that:

$$E[Y_{ECN}] = E[\delta] + E[W_{ECN}] - 1 \tag{2}$$

$E[\delta]$ can be computed as:

$$E[\delta] = \sum_{k=1}^{\infty} (1-p_e)^{k-1} p_e k = \frac{1}{p_e} \tag{3}$$

since the probability that $\delta_i = k$ denotes the probability that the first $k-1$ packets are successfully acknowledged and the kth packet is ECN marked. This is equivalent to $(1-p_e)^{k-1}p_e$. Therefore, $E[Y_{ECN}]$ can be rewritten as:

$$E[Y_{ECN}] = \frac{1-p_e}{p_e} + E[W_{ECN}] \tag{4}$$

The window size increases linearly between βW_{i-1} and W_i, with slope $\frac{\alpha}{b}$ (we remove the ECN subscript of α and β for readability), for the i^{th} ECNP and the $(i-1)^{st}$ TDP or ECNP (Figures 2 and 3). Let X_i denote the round when congestion indication is received. Thus, we have:

$$W_i = \beta W_{i-1} + \frac{\alpha X_i}{b}, i = 1, 2, \cdots \tag{5}$$

Since Y_i packets are transmitted during ECNP$_i$,

$$Y_i = \sum_{k=0}^{\frac{\alpha X_i}{b}-1} (\beta W_{i-1} + k)\, b + \mu_i \tag{6}$$

where μ_i is the number of packets sent in the last round (see Figure 3). From equations (5) and (6), we have:

$$E[W_{ECN}] = \frac{\alpha}{b(1-\beta)} E[X_{ECN}] \tag{7}$$

$$E[Y_{ECN}] = \frac{\alpha E[X_{ECN}]}{2}((1+\beta)E[W_{ECN}] - 1) + \frac{E[W_{ECN}]}{2} \tag{8}$$

where $E[\mu] = \frac{E[W_{ECN}]}{2}$, assuming that μ_i is uniformly distributed between 1 and W_i. From equations (4), (7) and (8), $E[W_{ECN}]$, $E[Y_{ECN}]$ and $E[X_{ECN}]$ can be computed as:

$$E[W_{ECN}] = \frac{b(1-\beta) + 1 + \sqrt{(b(1-\beta)+1)^2 + \frac{8b(1-\beta^2)(1-p_e)}{p_e}}}{2b(1-\beta^2)} \tag{9}$$

$$E[Y_{ECN}] = \frac{1-p_e}{p_e} + E[W_{ECN}] \tag{10}$$

$$E[X_{ECN}] = \frac{b(1-\beta)}{\alpha} E[W_{ECN}] \tag{11}$$

The duration of the i^{th} ECNP is $A_i = \sum_{j=1}^{X_i+1} r_{ij}$, where r_{ij} is the RTT of the jth round of the ith ECNP. It follows that:

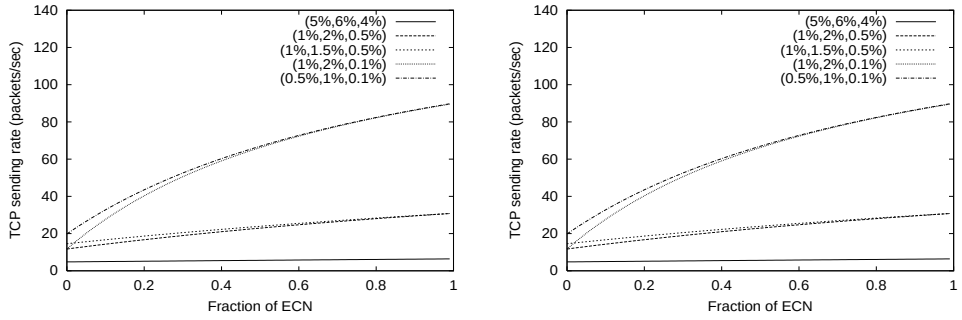
$$\begin{aligned} E[A_{ECN}] &= E[r](E[X_{ECN}]+1) = RTT(E[X_{ECN}]+1) \\ &= RTT\left(\frac{b(1-\beta)}{\alpha}E[W_{ECN}] + 1\right) \end{aligned} \tag{12}$$

$E[W_{(1,0.5)}]$, $E[Y_{(1,0.5)}]$ and $E[A_{(1,0.5)}]$ can be derived using $(\alpha, \beta) = (1, 0.5)$ for $E[W_{ECN}]$, $E[Y_{ECN}]$ and $E[A_{ECN}]$, respectively, and p_t for p_e. Hence, we have:

$$E[W_{(1,0.5)}] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p_t)}{3bp_t} + \left(\frac{2+b}{3b}\right)^2} \tag{13}$$

$$E[Y_{(1,0.5)}] = \frac{1-p_t}{p_t} + E[W_{(1,0.5)}] \tag{14}$$

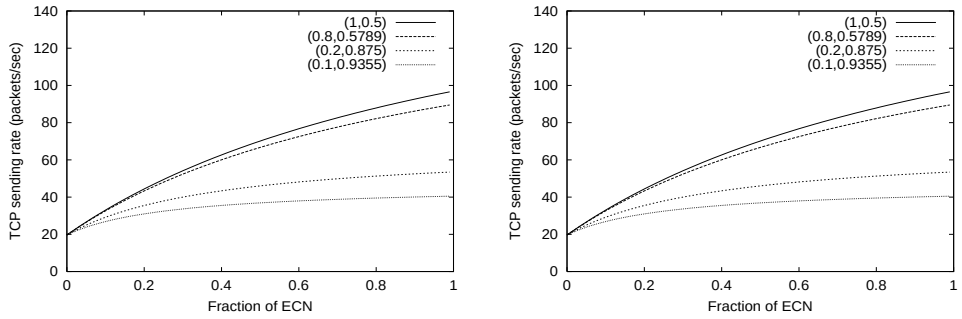
$$E[A_{(1,0.5)}] = RTT(E[W_{(1,0.5)}]+1) \tag{15}$$



(a) (α_{ECN},β_{ECN})=(0.2,0.875) (b) (α_{ECN},β_{ECN})=(0.8,0.5789)

Fig. 4. TCP sending rate using our new ECN response (α_{ECN}, β_{ECN}), according to the model in equation (1)

In Figures 4 and 5, we use equation (1) to plot the TCP sending rate with the new ECN response (α_{ECN}, β_{ECN}) for different ρ values (ρ is the fraction of ECN among the sum of ECNs and TDs). We use $RTT = 0.2$ seconds, $b = 2$, and $T_0 = 3.0$ seconds. Figure 4 shows the TCP sending rates for different p, p_t, p_e, and ρ values, with two fixed (α_{ECN}, β_{ECN}) values. In the figure, (5%,6%,4%) means $p = 5\%$, $p_t = 6\%$, and $p_e = 4\%$. The figure demonstrates that the TCP sending rate increases as ρ increases for $p_e \leq p \leq p_t$. As the probabilities p and p_e become smaller, the increase is more pronounced (e.g., (1%,2%,0.1%) and (0.5%,1%,0.1%)). Our simulations in Section V-B show that $p_t \leq p \leq p_e$ rarely occurs. Figure 5 depicts the TCP sending rates for different (α_{ECN}, β_{ECN}) and ρ values, for ($p = 1\%$, $p_t = 1\%$, $p_e = 0.1\%$) and ($p = 0.5\%$, $p_t = 1\%$, $p_e = 0.1\%$). In both cases, the TCP sending rate



(a) Congestion indication probability $p = 1\%$, $p_t = 1\%$, $p_e = 0.1\%$

(b) Congestion indication probability $p = 0.5\%$, $p_t = 1\%$, $p_e = 0.1\%$

Fig. 5. TCP sending rate using our new ECN response (α_{ECN}, β_{ECN}), according to the model in equation (1)

increases as ρ increases. We observe that the TCP sending rates with $(1, 0.5)$ and $(0.8, 0.5789)$ are larger than the TCP sending rates with $(0.2, 0.875)$ and $(0.1, 0.9355)$. This indicates that TCP-ECN with $(1, 0.5)$ and with $(0.8, 0.5789)$ are more aggressive than TCP-ECN with $(0.2, 0.875)$ and $(0.1, 0.9355)$. These results show that early congestion indication with our new ECN response (α_{ECN}, β_{ECN}) increases performance. In addition, this model may be used for equation-based ECN(α,β) congestion control since the values of RTT, ρ, p, p_e, and p_t can be easily computed.

C. RED-ECN as a Feedback Control System

We model n ECN(α_{ECN}, β_{ECN}) flows sharing a single RED-ECN router, following a similar methodology to that in [9]. Our goal is to determine the equilibrium point $(p_s, \overline{q}_s)$, where p_s is the packet drop/mark probability, and $\overline{q}_s$ is the average queue length in steady state. Note that the model in [9] makes several simplifying assumptions. The model considers n TCP-Reno flows sharing only *one* bottleneck link where packets may be discarded. The TCP flows are established from senders to receivers, while the opposite direction only carries ACKs. The number of flows n remains constant for a long period of time, and all flows have long durations. All flows are also assumed to have the same average round-trip time.

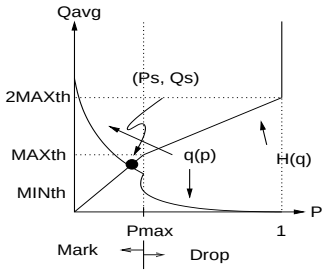


Fig. 6. Equilibrium point of TCP(α_{ECN},β_{ECN}) and RED-ECN

We extend the model in [9] to model ECN(α, β) at the sender and RED-ECN at the router. RED serves as a feedback control system in this context. Unlike previous work, we consider a RED-ECN router that can both mark and drop in different ranges of average queue length (as the ECN proposal specifies).

Assume the ECN(α_{ECN}, β_{ECN}) flows have the same average round-trip time RTT, such that $RTT = RTT_0 + \overline{q}/c$, where RTT_0 is the propagation time that excludes queuing delay, and $\overline{q}/c$ represents the queuing delay. c is the bottleneck link capacity and $\overline{q}$ is the average queue size.

The ECN(α_{ECN}, β_{ECN}) TCP sending rate from Section IV-B is used as the bandwidth, $\hat{B}$, in two different modes, marking and dropping, as a function of p:

$$\hat{B}(RTT, p, \rho) = \begin{cases} B(RTT, p, 1) & 0 < p \leq p_{max} \\ B(RTT, p, 0) & p > p_{max} \end{cases} \tag{16}$$

Assume that the link bandwidth is fully utilized:

$$\hat{B}(RTT_0 + \overline{q}/c, p, \rho) = c/n \tag{17}$$

At the router, we use RED-ECN, $H(\overline{q})$, as a feedback control function in the "gentle" RED mode, defined in Section II-C. Q_{size} denotes the maximum queue size. Note that RED-ECN marks packets when $q_{min} \leq \overline{q} < q_{max}$, but drops packets when $\overline{q} \geq q_{max}$:

$$H(\overline{q}) = \begin{cases} 0 & 0 \leq \overline{q} < q_{min} \\ \frac{\overline{q}-q_{min}}{q_{max}-q_{min}} p_{max} & q_{min} \leq \overline{q} < q_{max} \\ \frac{1-p_{max}}{q_{max}}(\overline{q} - q_{max}) + p_{max} & q_{max} \leq \overline{q} < 2q_{max} \\ 1 & 2q_{max} \leq \overline{q} \leq Q_{size} \end{cases} \tag{18}$$

Finally, we obtain two relations between p and $\overline{q}$. One is from the inverse function of $\hat{B}$ in equation (17), and the other one is from equation (18):

$$\begin{cases} \overline{q} = c(\hat{B}^{-1}(p, c/n) - RTT_0) \\ p = H(\overline{q}) \end{cases} \tag{19}$$

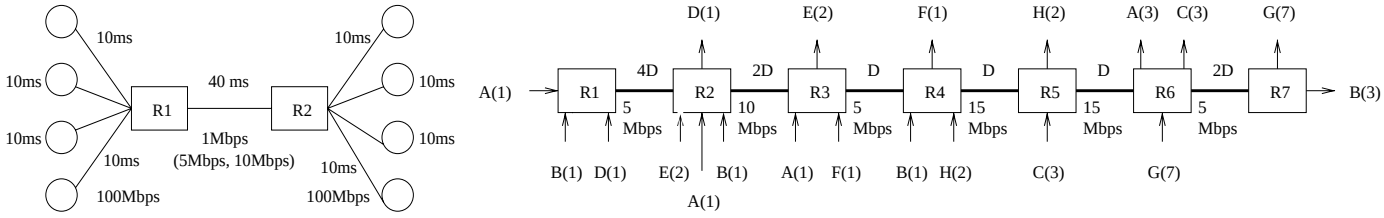
The equilibrium point $(p_s, \overline{q}_s)$ obtained from the above equations is illustrated in Figure 6. Note that the x-axis of the graph is divided into two parts, corresponding to marking and dropping.

V. Performance Analysis

In this section, we validate our analysis, and we evaluate the performance of the new ECN response strategy. We first discuss the simulation setup and the performance metrics, and then analyze the results.

A. Simulation Setup

We use the network simulator ns-2.1b6 [36] in this study. The configuration for the first set of simulations is the WAN configuration depicted in Figure 7(a). We run three different simulations with different bottleneck capacities: 1, 5 and 10 Mbps. We simulate 20 unlimited FTP TCP connections with 5 sessions at each sending node. The total simulation time is 100 seconds.



(a) Simple wide area network configuration (WAN)

(b) The Generic Fairness Configuration-2 (GFC-2)

Fig. 7. Topologies used in the simulations

We also use the Generic Fairness Configuration-2 (GFC-2), illustrated in Figure 7(b). GFC-2 contains multiple bottlenecks, and connections with different round-trip times. We use D = 5 ms. There are 22 unlimited bulk-data FTP TCP connections in each direction, 6 UDP connections (modeled as 0.5 Mbps CBR (Constant Bit Rate) flows), and 22 HTTP traffic flows. The HTTP traffic is generated using a Poisson Process where the inter-object, inter-page and inter-session times follow an exponential distribution. The simulation time is 60 seconds.

We use a timer granularity of 100 ms and a segment size of 1000 bytes. All routers in our simulations use gentle RED with packet marking for ECN [12]. As previously explained, with gentle RED, the packet drop probability varies linearly from p_{max} to 1 as the average queue size varies from th_{max} to twice th_{max}. The buffer size is 168 KB. The th_{min} value is $\frac{1}{12}\times$ the buffer size, and the th_{max} value is $\frac{1}{4}\times$ the buffer size ($3 \times th_{min}$), as recommended by RED

designers [12]. We run 5 simulations and average them. (To ensure comparability, we fix the random number generator seed for each run when we compare different approaches.)

We use the following performance metrics: (1) **Goodput (Mbps):** Total data received *at the application level* by all receivers during the simulation time, divided by the simulation time; (2) **Packet Drop Percentage:** The ratio of dropped packets to the total number of packets sent during the simulation time (multiplied by 100); and (3) **Response Time (seconds):** The (mean and maximum) time between the instant when the request of an HTTP client is triggered, and the instant when the last requested page from a server arrives at that client.

B. Results and Discussion

We first validate the throughput and feedback models discussed in previous sections. In Figure 8, we validate the throughput model for various numbers of flows on the WAN configuration. The x-axis denotes values from the simulations, while the y-axis denotes predicted values from our throughput model. The closer the points are to the $y = x$ line, the more accurate the model is. The figure shows that our model predicts throughput values close to the simulated ones, especially when the bottleneck bandwidth is 1 Mbps and 5 Mbps.

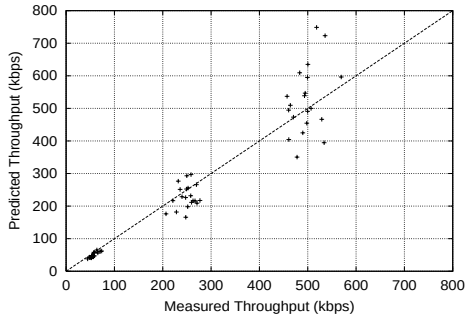


Fig. 8. Predicted versus simulated throughput values

Figure 9 depicts the fraction of packets sent during ECNPs among the packets sent during the entire simulation, for all 20 flows. In the figure, all the flows in the configurations with 5 and 10 Mbps bottleneck links send more than 90% of their packets during ECNPs. All the flows in the configuration with 1 Mbps bottleneck link send more than 80% packets during ECNPs. This result indicates that the majority of packets are sent during ECNPs. The congestion indication probabilities p, p_t, and p_e for each flow in the simulation with a 10 Mbps bottleneck are plotted in Figure 10. The figure validates that our choices of p, p_t and p_e (used in Figures 4 and 5) are indeed reasonable ($p_e < p \ll p_t$).

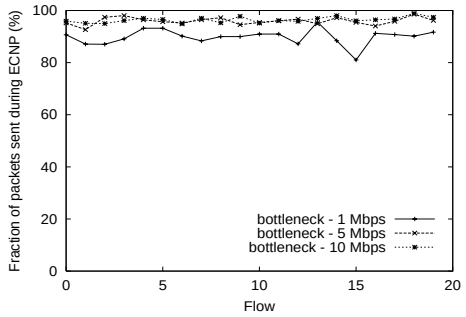


Fig. 9. Fraction of packets sent during ECNPs among all packets sent during the simulation period for all 20 flows

We also compare the steady-state equilibrium points $(p_s, \overline{q}_s)$ acquired from our model with simulation measurements. Figure 11(a) depicts the steady-state equilibrium point $(p_s, \overline{q}_s)$ predicted by the model. The predicted $\overline{q}$ and $p = H(q)$ from equation (19) intersect at approximately (0.08, 40). The model $p = H(q)$ uses 14 packets as th_{min} on the y-axis, 42 for th_{max}, and 84 for $2 \times th_{max}$ ($2 \times th_{max}$ is used in "gentle" RED). RED-ECN marks packets if $p < p_{max} = 0.1$, and drops them if $p \geq p_{max} = 0.1$. Figure 11(b) illustrates that the actual average RED queue size (from the simulations) lies between 40 and 42 KB. The measured packet mark/drop probability over the entire simulation is approximately 8% (not shown). This verifies that our model is accurate with an active queue management scheme such as RED-ECN.

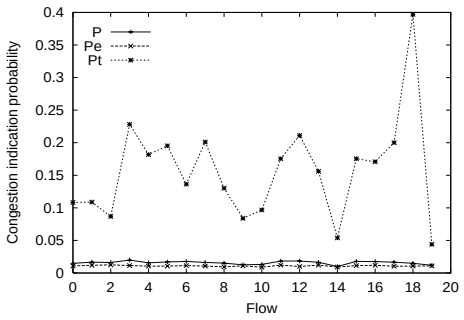
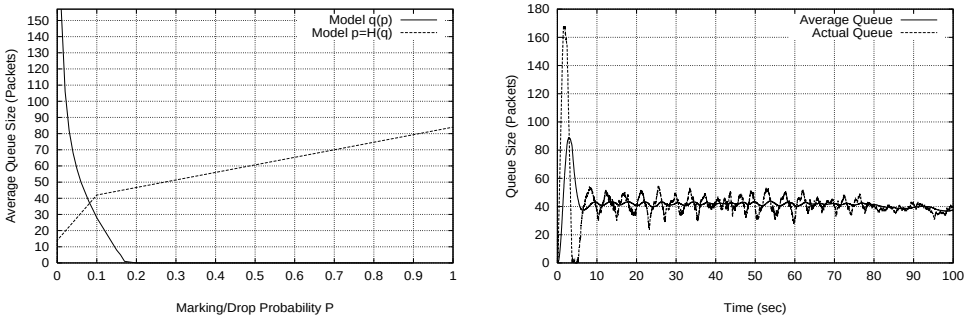


Fig. 10. Congestion indication probabilities p, p_t, and p_e in the simulation for all 20 flows



(a) Predicted steady-state average queue

(b) Actual average queue size from simulation

Fig. 11. RED Equilibrium point for (α_{ECN},β_{ECN})

Table I compares the performance of TCP-Reno without ECN, TCP-Reno with ECN, and ECN(α, β) for three different bottleneck link bandwidths (1, 5 and 10 Mbps) on the WAN configuration. We use α_{ECN} of 0.2 and β_{ECN} of 0.875 in this set of simulations. From the table, it is clear that ECN(α, β) has higher performance than TCP-Reno without ECN, and similar performance to TCP-Reno with ECN in terms of goodput and packet drop percentage.

Table II compares the performance of TCP-Reno without ECN, with ECN, and ECN(α, β) for the GFC-2 configuration. Again, we use α_{ECN} of 0.2 and β_{ECN} of 0.875 [13] in this set of simulations. From the table, it is clear that ECN(α, β) exhibits higher performance than TCP-Reno without ECN or with ECN. ECN(α, β) total goodput (HTTP, FTP and UDP) is approximately 4 Mbps higher than TCP-Reno without ECN, and approximately 3 Mbps higher than TCP-Reno with ECN. In addition, ECN(α, β) exhibits the shortest mean response time for HTTP traffic. With respect to packet drop percentage, ECN(α, β) achieves significant improvement over TCP-Reno without ECN and with ECN for various traffic types.

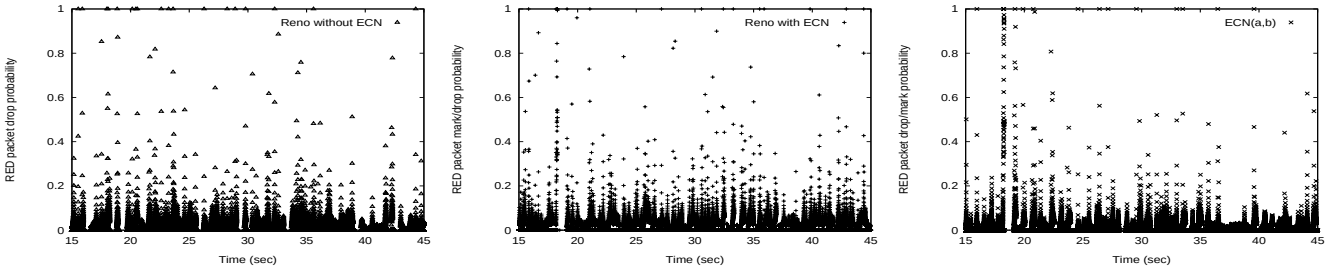
1) Responsiveness of ECN(α, β): Although ECN(α, β) achieves higher performance than TCP Reno without ECN or with ECN as shown above, our less dramatic reduction of congestion window as a response to ECN may cause slow responsiveness to a sudden surge of traffic. Recently, equation-based congestion control, e.g., TFRC [14], was shown to adequately respond to congestion due to its self-clocking mechanism [4]. In order to investigate the responsiveness

TABLE I
PERFORMANCE OF ECN(α, β)– WAN

Algorithm	Total Goodput			Packet Drop Percentage		
	1 Mbps	5 Mbps	10 Mbps	1 Mbps	5 Mbps	10 Mbps
Reno without ECN	0.980	4.988	9.527	8.93	3.63	2.06
Reno with ECN	0.981	4.999	9.920	5.53	0.59	0.39
ECN(α, β)	0.979	4.984	9.942	3.02	0.95	0.53

TABLE II
PERFORMANCE OF ECN(α, β)– GFC-2

Algorithm	HTTP Response Time		Goodput			Packet Drop Percentage			
	Mean	Max	HTTP	UDP	FTP	HTTP	UDP	FTP	Total
Reno without ECN	14.260	50.449	1.509	2.855	38.772	7.187	4.966	0.883	1.637
Reno with ECN	12.194	50.517	0.845	2.830	40.805	7.837	5.797	0.147	1.110
ECN(α, β)	11.481	52.237	2.339	2.881	41.890	4.278	4.146	0.119	0.854



(a) TCP-Reno without ECN (b) TCP-Reno with ECN (c) ECN(α, β)

Fig. 12. Drop or mark probability in GFC-2 simulation)

of ECN(α, β), we add 10 unlimited bulk-data sessions to the GFC-2 configuration. These additional sessions all start transmission after 20 seconds of the simulation time, and stop at time 40 seconds (20 seconds before the 60-second simulation ends). The RED drop probability without ECN, and the mark or drop probability with ECN, at the queues at the links from R2 and from R3 are shown in Figure 12 for the 3 algorithms (in separate simulation runs). We do not observe significantly higher values with ECN(α, β) than with ECN during the period of sudden congestion. Table III indicates that ECN(α, β) outperforms TCP Reno without ECN and with ECN in this scenario as well for (α, β) = (0.2,0.875).

TABLE III
RESPONSIVENESS OF ECN(α, β)– GFC-2

Algorithm	HTTP Response Time		Goodput			Packet Drop Percentage			
	Mean	Max	HTTP	UDP	FTP	HTTP	UDP	FTP	Total
Reno without ECN	13.607	49.732	0.615	5.783	35.789	9.635	5.480	1.003	1.804
Reno with ECN	12.082	54.730	0.657	5.958	37.648	8.456	5.918	0.190	1.157
ECN(α, β)	12.841	48.550	1.010	5.805	38.494	6.661	4.627	0.128	0.903

2) Effect of α_{ECN} and β_{ECN} Values: This section explores various pairs of (α_{ECN}, β_{ECN}) values to investigate their effects on performance. All parameter values other than α_{ECN} and β_{ECN} are set as in the previous experiment. The (α_{ECN}, β_{ECN}) pairs we use here follow the rule proposed in [13] except for (1, 0.9). We selected (1, 0.9) to compare values that follow the rule in [13] with a more aggressive (α_{ECN}, β_{ECN}) pair. From table IV, we see that (0.2, 0.875), (0.4, 0.7647) and (0.8, 0.5789) demonstrate marked goodput improvement over (1, 0.5) (with (0.2, 0.875) performing best). (α_{ECN}, β_{ECN}) values that follow the TCP-friendliness rule achieve good performance in terms of goodput as well as HTTP response time. Although (1, 0.9) exhibits competitive performance, its aggressiveness may be harmful to other TCP connections.

3) TCP-friendliness in Heterogeneous Configurations: To analyze the TCP-friendliness of our approach in heterogeneous configurations, we use the simple configuration in Figure 13. In this simulation, we use a 120 KB RED buffer size. The topology, link delays, and bandwidths are depicted in the figure. The RTTs of the 3 flows TEST1, TEST2, and TEST3 are the same as flows TCP1, TCP2, and TCP3, respectively. The first set of simulations evaluates the compatibility with TCP-Reno with ECN support, and the second compares with TCP-Reno without ECN.

TABLE IV

EFFECT OF α_{ECN} AND β_{ECN} VALUES– GFC-2

(α_{ECN}, β_{ECN})	HTTP Response Time		Goodput			Packet Drop Percentage			
	Mean	Max	HTTP	UDP	FTP	HTTP	UDP	FTP	Total
(1, 0.5)	12.194	50.517	0.845	2.830	40.805	7.837	5.797	0.147	1.110
(0.8, 0.5789)	11.328	48.538	1.516	2.844	42.195	6.302	5.389	0.137	1.011
(0.4, 0.7647)	10.282	49.643	1.294	2.862	41.604	5.481	4.787	0.113	0.909
(0.2, 0.875)	11.481	52.237	2.339	2.881	41.890	4.278	4.146	0.119	0.854
(1, 0.9)	12.060	52.623	1.665	2.733	42.400	6.664	9.013	0.386	1.773

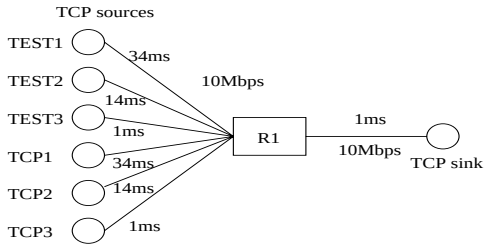


Fig. 13. Simple single bottleneck configuration

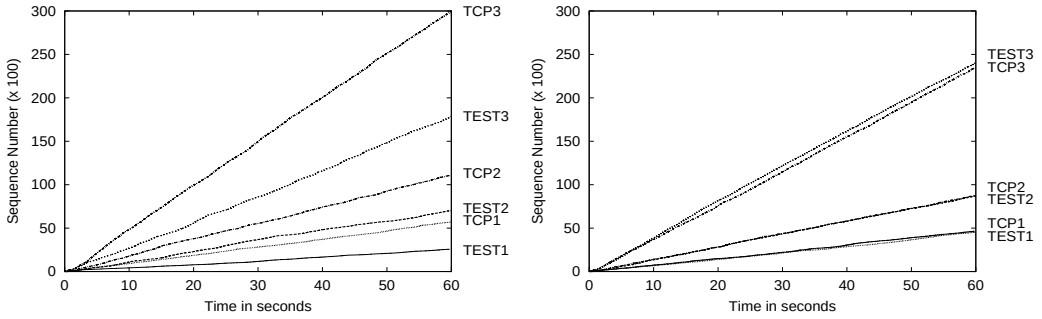
Each set of simulations consists of two simulations. The first is a mixed simulation where the first three flows TEST1, TEST2, and TEST3 use ECN(α, β), and the remaining flows TCP1, TCP2, and TCP3 use TCP-Reno. The second simulation is a homogeneous simulation in which all six flows TEST1-TEST3 and TCP1-TCP3 are TCP-Reno. We plot the sequence numbers over time for the mixed flows scenario (upper graph (a)) and the all TCP-Reno flows scenario (lower graph (b)). We compare the two graphs to see how much the ECN(α, β) flows affect co-existing TCP-Reno flows TCP1-TCP3. This verifies whether TCP-friendliness is preserved.

Figure 14 shows that TEST1, TEST2, and TEST3, which use our new ECN response strategy, do not significantly affect the other three flows TCP1, TCP2, and TCP3, which use TCP-Reno with ECN support. Comparing the upper graph with the lower graph, we observe that TCP1, TCP2, and TCP3 achieve similar rates in both the heterogeneous and homogeneous versions. The figure shows that ECN(α, β) sends packets less aggressively than TCP-Reno with ECN, since it considers ECN as an early warning sign. ECN(α, β) with $\alpha_{ECN} = 0.3125$ is more aggressive than with $\alpha_{ECN} = 0.2$. However, even with $\alpha_{ECN} = 0.3125$, ECN(α, β) is still less aggressive than TCP-Reno with ECN. Due to the disparity of RTTs, the sequence number curves with similar RTTs are similar in the homogeneous simulations. The corresponding result for Reno without ECN support is shown in Figure 15. In this case, the rate of Reno without ECN (TCP1-3 in the upper graph) is affected more than the version in Figure 14(a). The TCP-Reno flow without ECN with the shortest RTT (TCP3) suffers some throughput degradation compared to the ECN(α, β) flow TEST3. TCP-Reno flows, however, still achieve almost the same throughput values as the homogeneous case (Figure 14(b)), except for that very short RTT connection, which anyway has an excessive bandwidth share.

VI. CONCLUSIONS AND FUTURE WORK

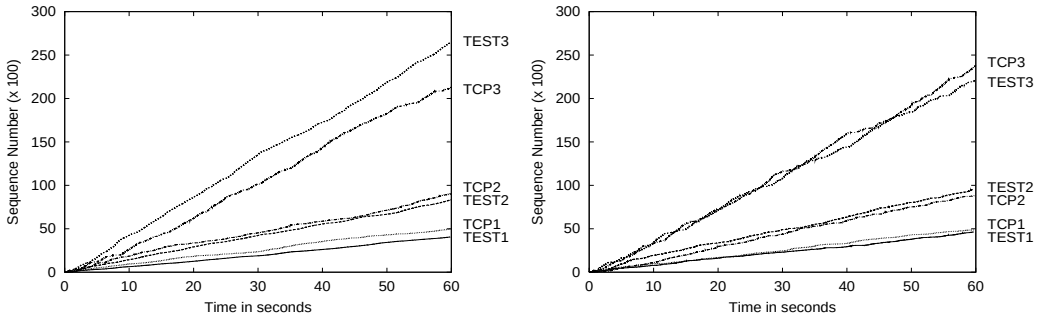
We have explored a smooth TCP response to ECN marks, which considers ECN as an early warning sign. We use a more aggressive short term behavior, while preserving the long term behavior equivalent to packet drop. We model TCP throughput to consider the new ECN response strategy, and verify that our model is accurate. Our simulation results in a multiple bottleneck scenario with bulk FTP, bursty HTTP connections, and UDP flows demonstrate that our ECN response strategy does indeed reduce fluctuations, increase goodput, and reduce delay. This provides incentives for host ECN-compliance. The increase and decrease parameters (0.2, 0.875) appear to be the best choice for preserving the long-term behavior of TCP and achieving high performance.

We are currently evaluating an adaptive version of ECN(α,β) that dynamically adjusts (α,β) based on the number of ECN-marked packets received during an interval of time. We are also studying fairness among flows that use or do not use ECN (especially with heterogeneous RTTs). We plan to exploit ECN(α,β) to shorten the recovery time



(a) Heterogeneous configuration (b) Homogeneous configuration

Fig. 14. Sequence numbers from simulations of Reno with ECN support



(a) Heterogeneous configuration (b) Homogeneous configuration

Fig. 15. Sequence numbers from simulations of Reno with no ECN support

after transient congestion detection. This is especially important in the context of high-speed networks [22], [23]. We will also examine the interaction between our new algorithm and a variety of active queue management schemes. In particular, variants of ECN(α,β) with implicit or explicit feedback mechanisms for high-speed networks such as [11], [24] will be an interesting topic for further study.

ACKNOWLEDGMENTS

This research has been sponsored in part by the Schlumberger Foundation technical merit award, and NSF grant ANI-0238294 (CAREER).

REFERENCES

[1] BLUE: An alternative approach to active queue management. In *Proc. of the ACM NOSSDAV*, pages 41–50, June 2001. http://www.thefengs.com/wuchang/work/blue/nossdav01.ps.

[2] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, April 1999. Also see http://tcpsat.lerc.nasa.gov/tcpsat/papers.html.

[3] S. Athuraliya, D. E. Lapsley, and S. H. Low. An Enhanced Random Early Marking Algorithm for Internet Flow Control. In *Proc. of the IEEE INFOCOM*, volume 3, pages 1425–1434, March 2000. http://www.ieee-infocom.org/2000/papers/494.ps.

[4] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. Dynamic behavior of slowly-responsive congestion control algorithms. In *Proc. of the ACM SIGCOMM*, pages 263–274, August 2001. http://www.acm.org/sigcomm/sigcomm2001/p21.html.

[5] L. Brakmo, S. O'Malley, and L. Peterson. TCP vegas: New techniques for congestion detection and avoidance. In *Proc. of the ACM SIGCOMM*, pages 24–35, August 1994. http://netweb.usc.edu/yaxu/Vegas/Reference/vegas93.ps.

[6] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN Systems*, 17(1):1–14, June 1989. http://www.cis.ohio-state.edu/∼jain/papers/cong_av.htm.

[7] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith. Tuning RED for web traffic. In *Proc. of the ACM SIGCOMM*, pages 139–150, August 2000. http://www.acm.org/sigs/sigcomm/sigcomm2000/.

[8] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. In *ACM Computer Communication Review*, volume 26, pages 5–21, July 1996. ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z.

[9] V. Firoiu and M. Borden. A study of active queue management for congestion control. In *Proc. of the IEEE INFOCOM*, volume 3, pages 1435–1444, March 2000. http://www.ieee-infocom.org/2000/papers/405.pdf.

[10] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5):8–23, October 1994. http://www.aciri.org/floyd/.

[11] S. Floyd. HighSpeed TCP for Large Congestion Windows. Internet draft draft-floyd-tcp-highspeed-02.txt, work in progess, February 2003. http://www.icir.org/floyd/hstcp.html.

[12] S. Floyd. RED web page. http://www.aciri.org/floyd/red.html, 2003.

[13] S. Floyd, M. Handley, and J. Padhye. A comparison of equation-based and AIMD congestion control. http://www.aciri.org/tfrc/, May 2000.

[14] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of the ACM SIGCOMM*, pages 43–56, August 2000. http://www.acm.org/sigs/sigcomm/sigcomm2000/.

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

[16] T. Hamann and J. Walrand. A new fair window algorithm for ECN-capable TCP (New-ECN). In *Proc. of the IEEE INFOCOM*, volume 3, pages 1528–1536, March 2000. http://www.ieee-infocom.org/2000/papers/153.pdf.

[17] J. Hoe. Improving the start-up behavior of a congestion control scheme for TCP. In *Proc. of the ACM SIGCOMM*, pages 270–280, August 1996. http://www.acm.org/sigcomm/ccr/archive/1996/conf/hoe.ps.

[18] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proc. of the IEEE INFOCOM*, volume 3, pages 1726–1734, April 2001. http://www.ieee-infocom.org/2001/792.pdf.

[19] V. Jacobson. Congestion avoidance and control. In *Proc. of the ACM SIGCOMM*, volume 18, pages 314–329, August 1988. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[20] R. Jain, K. K. Ramakrishnan, and D. M. Chiu. Congestion avoidance in computer networks with a connectionless network layer. Digital Equipment Corporation, Technical Report, DEC-TR-506, August 1987, 17 pp. Also in C. Partridge, Ed., Innovations in Internetworking, Artech House, Norwood, MA, 1988, pp. 140-156.

[21] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore. The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks. *IEEE/ACM Transactions on Networking*, 8(1):87–98, February 2000. http://www.cis.ohio-state.edu/∼jain/papers/erica.htm.

[22] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-delay Product Networks. In *Proc. of the ACM SIGCOMM*, pages 89–102, August 2002. http://www.acm.org/sigs/sigcomm/sigcomm2002/papers/xcp.html.

[23] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. Submitted, December 2002. http://www-lce.eng.cam.ac.uk/ ctk21/scalable/.

[24] S. Kunniyur. AntiECN Marking: A Marking Scheme for High Bandwidth Delay Connections. In *Proc. of the IEEE ICC*, May 2003. http://www.seas.upenn.edu/ kunniyur/.

[25] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *Proc. of the ACM SIGCOMM*, pages 123–134, August 2001. http://www.acm.org/sigcomm/sigcomm2001/p10.html.

[26] M. Kwon and S. Fahmy. TCP increase/decrease behavior for explicit congestion notification (ECN). In *Proceedings of IEEE ICC*, volume 4, pages 2335–2340, April 2002. http://www.cs.purdue.edu/homes/fahmy/.

[27] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5:336–350, June 1997.

[28] D. Lin and R. Morris. Dynamics of random early dectection. In *Proc. of the ACM SIGCOMM*, volume 27, pages 127–136, September 1997.

[29] C. Liu and R. Jain. Improving explicit congestion notification with the mark-front strategy. *Computer Networks*, 35(2-3):185–201, February 2001. http://www.cis.ohio-state.edu/∼jain/papers/ecnfront.htm.

[30] M. Mathis and J. Mahdavi. Forward acknowledgment: Refining TCP congestion control. In *Proc. of the ACM SIGCOMM*, pages 281–291, August 1996. Also see http://www.psc.edu/networking/papers/papers.html.

[31] T. J. Ott. ECN protocols and the TCP paradigm. In *Proc. of the Workshop on Modeling of Flow and Congestion Control Mechanisms*, September 2000. ftp://ftp.research.telcordia.com/pub/tjo/ECN.ps.

[32] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proc. of the IEEE INFOCOM*, volume 3, pages 1346–1355, March 1999. http://www.ieee-infocom.org/1999/papers/09e_04.pdf.

[33] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. of the ACM SIGCOMM*, volume 28, pages 303–314, September 1998. http://gaia.cs.umass.edu/.

[34] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ECN) to IP. RFC 2481, January 1999.

[35] J. Salim and U. Ahmed. Performance evaluation of explicit congestion notification (ECN) in IP networks. RFC 2488, July 2000. http://www7.nortel.com:8080/CTL/ecnperf.pdf.

[36] UCB/LBNL/VINT groups. UCB/LBNL/VINT Network Simulator. http://www.isi.edu/nsnam/ns/, 2003.

[37] Y. Yang and S. Lam. General AIMD congestion control. In *Proc. of the IEEE ICNP*, November 2000. http://www.cs.utexas.edu/users/lam/Vita/IEEE/YangLam00.pdf.