

TCP Increase/Decrease Behavior with Explicit Congestion Notification (ECN)

Minseok Kwon and Sonia Fahmy

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-1398
{kwonm,fahmy}@cs.purdue.edu

Abstract—We investigate the effect of TCP Explicit Congestion Notification (ECN) with a new response strategy that is more aggressive in the short term, but preserves TCP long term behavior, without modifying the router marking rate. A less aggressive ECN decrease gives more incentives for end systems to become ECN-compliant, as ECN serves as an early warning sign in this case. Our analysis and simulation results demonstrate the effectiveness of the new algorithm in improving throughput and reducing fluctuations. We model a multiple bottleneck scenario with various types of traffic, and evaluate the effect of a number of parameters, including TCP flavor, increase/decrease parameters, buffer size, and random early detection (RED) parameters.

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

I. INTRODUCTION

End system congestion avoidance mechanisms have been a topic of active research since the early 1980s [1], [2], [3], [4], [5]. The TCP congestion avoidance algorithm uses additive increase when no losses are detected and multiplicative decrease with loss. The congestion window is halved in response to a single packet drop and is linearly increased by one segment every round trip time (RTT) during the congestion avoidance phase. Thus the increase slope is 1 and the decrease fraction is $\frac{1}{2}$, or what we refer to as AIMD (α, β) or AIMD (1,0.5).

Router-based control gained attention in 1993 when Floyd and Jacobson proposed the Random Early Detection (RED) scheme [6]. A lot of research has focused on refining drop mechanisms [7], [8], [9], [10]. The main aim of these schemes is to drop packets before buffer overflow to give an early warning to the sources, avoid TCP synchronization and bias against bursty connections, and punish misbehaving flows. RED drops probabilistically when the average buffer occupancy lies between two thresholds, and drops all packets when the maximum threshold is exceeded. The Explicit Congestion Notification (ECN) mechanism proposes that RED only mark (and not drop) the packets when buffer occupancy lies between the two thresholds. Although the source response to ECN should match its response to packet drop, [11] indicates that this matching requirement is a long term one. This allows the possibility of less aggressive reduction in the short term. We explore this idea in this paper.

We react to the receipt of ECN with a relatively small window decrease, but modify the increase parameter to increase less aggressively for a period of time. Our algorithm resets the increase and decrease parameters back to (1,0.5) in the event of retransmission timeouts or duplicate ACKs following the ECN decrease. This smooth ECN response corresponds to its motivation to serve as an early warning sign of congestion; and the improved performance we show in this pa-

per gives incentives for hosts to become ECN-compliant. The AIMD parameters for ECN are carefully chosen to preserve the long term behavior of TCP without modifying the router marking rate from standard RED. The window size and hence sending rate are less oscillatory, improving overall throughput and delay variance. We model the algorithm throughput and steady state marking and dropping probabilities. We investigate different increase and decrease values only as a response to ECN, since (1,0.5) are standardized for TCP [12], but the response to ECN is not enforced in [13]. Simulation results with short/long lived FTP, UDP and web connections, multiple bottleneck configurations, and various TCP flavors and parameters, show the effectiveness of our strategy.

The paper is organized as follows. Section II gives some 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 algorithm, and section V simulates it 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 [1]. The sender initially sets its congestion window, $cwnd$, to 1 or 2 segments [12]. For each ACK received, the $cwnd$ is increased by one segment, resulting in an exponential increase of the $cwnd$ over round trips. TCP uses the slow start threshold, $ssthresh$, to indicate the appropriate window size depending on 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 the $cwnd$ by one segment every round trip (an additive increase). The TCP sender assumes there is congestion when it times out waiting for an ACK or when it receives 3 duplicate acknowledgments. $ssthresh$ is halved (a multiplicative decrease). The Additive Increase Multiplicative Decrease (AIMD) system has been shown to be stable under certain assumptions [14], [15], [16].

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

—This research has been sponsored in part by the Purdue Research Foundation, and the Schlumberger Foundation technical merit award.

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 drop probability value, p_{max} . RED includes an option known as the “gentle” variant. 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} .

C. The ECN Mechanism

The Explicit Congestion Notification (ECN) option [11], [13] allows active queue management mechanisms such as RED to probabilistically mark (rather than drop) packets when the average queue length lies between two thresholds, 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, so the sender knows that the network is approaching a congested state. The sender should therefore 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 [13]. 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, but gentle RED drops less aggressively between the maximum threshold and twice the maximum threshold. The main advantages of ECN are 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 been conducted on RED and ECN performance. Variations of RED include flow RED [7], stabilizing RED [8], and BLUE [17]. The effect of RED parameter values on web traffic is studied in [9], and the effect of marking from the front of the queue is investigated in [10]. A recent performance study on ECN with real traffic is presented in [18]. Ott [19] investigates ECN with various response algorithms. The study considers environments where non-ECN compatibility is not required and the marking rate can be increased, as with the REM, AVQ and PI controller proposals.

Design of AIMD algorithms has also been an active research area. The additive increase (α) and multiplicative decrease (β) parameters have been studied in early work [14], [15], but more recently Yang and Lam [20] have derived the relationship between the two parameters necessary for TCP friendliness. 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 [21], [22], the authors propose equation-based congestion control (based on TCP models in [23]) and compare it with TCP using a number of AIMD parameters. The authors use a slightly different relationship between the additive increase and multiplicative decrease parameters from [20] (they use $\alpha = \frac{3(1-\beta)}{1+\beta}$ which gives smaller α values than that in [20] for the range of β values we are interested in). As in [20], the authors of [22] focus on AIMD parameters in general, and not in the context of ECN as we do in this paper. Another class of AIMD algo-

Fig. 1. Pseudo-code for ECN with modified AIMD parameters

```

When an ACK with ECN is received:
  Reduce ssthresh and cwnd by  $\beta_{ECN}$ 
  Set IncreaseSlope to  $\alpha_{ECN}$ 

On a timeout or 3 duplicate ACKs:
  Reduce ssthresh and cwnd normally
  Reset IncreaseSlope to 1

Congestion avoidance:
   $cwnd = cwnd + \frac{IncreaseSlope}{cwnd}$ 

```

gorithms, binomial algorithms, is studied in [24] for streaming applications.

A study that investigates the response to ECN without changing the marking rate is [25]. The authors propose an algorithm to react to ECN and at the same time remove TCP’s bias to short RTT connections by modifying the window increase slope to become proportional to RTT^2 (hence the rate increase slope is proportional to RTT). The main problem with [25] is that RTT bias elimination should be independent of ECN marking. Further, the algorithm is not TCP-friendly and is highly sensitive to the RED maximum drop probability. We address each of these issues in our work. We only study the ECN reaction, however. Differentiated services with intelligent traffic conditioners have addressed the RTT sensitivity problem in [26].

IV. PROPOSED ECN ADDITIVE INCREASE MULTIPLICATIVE DECREASE BEHAVIOR

In this section, we describe our new ECN response algorithm and model it in connection with active queue management (AQM) routers such as RED.

A. New ECN Response

The main objective of this algorithm is to modify ECN response in order to improve throughput, reduce rate fluctuations and reduce delay variance. The pseudo-code for the basic algorithm is given in figure 1. The algorithm uses 2 parameters α_{ECN} and β_{ECN} to indicate the required increase and decrease parameters. Our performance study indicates (in section V) that (0.2,0.875) are good choices for $(\alpha_{ECN}, \beta_{ECN})$. The algorithm reduces the congestion window and slow start threshold by β_{ECN} in response to an ECN-marked packet, and uses a modified slope *IncreaseSlope* in the ensuing congestion avoidance phase. The increase slope is set to α_{ECN} with ECN, and reset to 1 with a timeout or receipt of 3 duplicate ACKs. Note that the congestion window increase and decrease follows TCP algorithms for all congestion indications other than ECN. Thus, β_{ECN} need not be reset with timeouts or 3 duplicate ACKs. The main idea is to match the use of the additive parameter with the corresponding decrease parameter that was applied. This less aggressive ECN decrease is more consistent with the use of ECN as an early warning sign, giving incentives for hosts to become ECN-capable. We call this algorithm $ECN(\alpha, \beta)$.

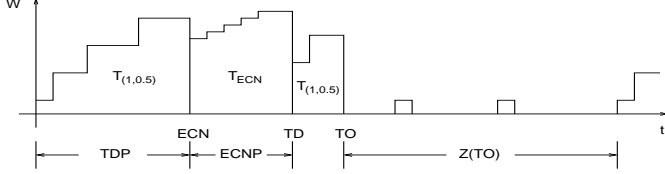


Fig. 2. Evolution of window size including $(\alpha_{ECN}, \beta_{ECN})$ for ECN

B. TCP Sending Rate

We extend the TCP sending rate model developed in [27], [23] to include the new response $(\alpha_{ECN}, \beta_{ECN})$ to ECN. With the new ECN behavior, there are three congestion indication types: ECN, TD (Triple-Duplicate ACKs) and TO (Time-Outs). Figure 2 shows a sample evolution of window size using $(1, 0.5)$ for triple duplicate acknowledgments and $(\alpha_{ECN}, \beta_{ECN})$ for ECN. $T_{(1,0.5)}$ and T_{ECN} represent the amount of data sent during periods starting after TD and ECN respectively.

We use the same notation as in [23]. The sending rate (or bandwidth) B is defined as in [23] except that $E[Y]$ is divided into $E[T_{(1,0.5)}]$ and $E[T_{ECN}]$, and $E[A]$ becomes $E[A_{(1,0.5)}]$ and $E[A_{ECN}]$. We introduce ρ which denotes the fraction of ECN indications among the sum of ECNs and TDs. Note that ECN indications are only counted once per round-trip time. $E[T_{(1,0.5)}]$ and $E[T_{ECN}]$ represent the mean number of packets sent during a TDP (a period starting from Triple-Duplicate ACKs up to the next congestion indication) and ECNP (a period starting from ECN up to the next congestion indication) respectively. Similarly, $E[A_{(1,0.5)}]$ and $E[A_{ECN}]$ are the mean durations of TDP and ECNP respectively. Q is the probability that a loss indication ending a TDP is a timeout; $E[R]$ represents packets sent during timeout sequence; and $E[Z^{TO}]$ is the duration of the timeout sequence as in [23]. Therefore, the sending rate B can be modeled as:

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

where $\rho = \frac{n(ECN)}{n(TD) + n(ECN)}$ and $n(x)$ denotes the number of congestion indications by method x . p is defined as the total congestion indication probability, including $n(ECN)$, $n(TD)$ and $n(TO)$. $E[T_{(1,0.5)}]$ and $E[A_{(1,0.5)}]$ are equal to $E[Y]$ and $E[A]$ in [23] respectively. $E[T_{ECN}]$ and $E[A_{ECN}]$ can be obtained from $E[Y]$ and $E[A]$ in [20]. The value of b , the number of packets acknowledged by a single ACK, is set to 2 as in [23]. We use approximated forms to simplify computations:

$$E[T_{(1,0.5)}] = \frac{1-p}{p} + \sqrt{\frac{4}{3p}} \quad (2)$$

$$E[A_{(1,0.5)}] = RTT \left(\sqrt{\frac{4}{3p}} + 1 \right) \quad (3)$$

$$E[T_{ECN}] = \frac{1-p}{p} + \sqrt{\frac{\alpha_{ECN}}{(1-\beta_{ECN}^2)p}} \quad (4)$$

$$E[A_{ECN}] = RTT \left(\sqrt{\frac{4(1-\beta_{ECN})}{\alpha_{ECN}(1+\beta_{ECN})p}} + 1 \right) \quad (5)$$

From (1) to (5) (we also remove the ECN subscript),

$$B = \frac{\frac{1-p}{p} + (1-\rho) \sqrt{\frac{4}{3p}} + \rho \sqrt{\frac{\alpha}{(1-\beta^2)p}} + Q \frac{1}{1-p}}{RTT \left((1-\rho) \sqrt{\frac{4}{3p}} + \rho \sqrt{\frac{4(1-\beta)}{\alpha(1+\beta)p}} + 1 \right) + QT_0 \frac{f(p)}{1-p}} \quad (6)$$

B can be approximated for small values of p as:

$$B(RTT, p, \rho) \approx \frac{1}{RTT \left((1-\rho) \sqrt{\frac{4p}{3}} + \rho \sqrt{\frac{4(1-\beta)p}{\alpha(1+\beta)}} \right) + T_{out}} \quad (7)$$

$$\text{where } T_{out} = T_0 \min(1, 3\sqrt{\frac{3p}{4}}) p (1 + 32p^2)$$

C. RED-ECN as a Feedback Control System

We model n modified $(\alpha_{ECN}, \beta_{ECN})$ TCP flows sharing a single RED-ECN router as in [28] to determine the equilibrium point (p_s, \bar{q}_s) where p_s is the packet drop/mark probability and \bar{q}_s is the average queue length in steady state. We extend [28] 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 modified TCP flows have the same average round trip time RTT , such that $RTT = RTT_0 + \bar{q}/c$, where RTT_0 is the propagation time that excludes queuing delay and \bar{q}/c represents queuing delay. c is the bottleneck link capacity and \bar{q} is the average queue size.

The modified TCP sending rate from section IV-B is used to represent the bandwidth, \hat{B} , in two different modes, marking and dropping, as a function of p :

$$\hat{B}(RTT, p, \rho) = \begin{cases} B(RTT, p, 1) & 0 < p \leq p_{max} \\ B(RTT, p, 0) & p > p_{max} \end{cases} \quad (8)$$

Assume that the link bandwidth is fully utilized:

$$\hat{B}(RTT_0 + \bar{q}/c, p, \rho) = c/n \quad (9)$$

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

$$H(\bar{q}) = \begin{cases} 0 & 0 \leq \bar{q} < q_{min} \\ \frac{\bar{q} - q_{min}}{q_{max} - q_{min}} p_{max} & q_{min} \leq \bar{q} < q_{max} \\ \frac{1 - p_{max}}{q_{max}} (\bar{q} - q_{max}) + p_{max} & q_{max} \leq \bar{q} < 2q_{max} \\ 1 & 2q_{max} \leq \bar{q} \leq Q \end{cases} \quad (10)$$

Finally, we have two relations between p and \bar{q} . One is from the inverse function of \hat{B} in (9) and the other one is from (10):

$$\begin{cases} \bar{q} = c(\hat{B}^{-1}(p, c/n) - RTT_0) \\ p = H(\bar{q}) \end{cases} \quad (11)$$

The equilibrium point (p_s, \bar{q}_s) obtained from the above equations is illustrated in figure 3. Note that the x -axis on the figure is divided into two parts, corresponding to marking and dropping.

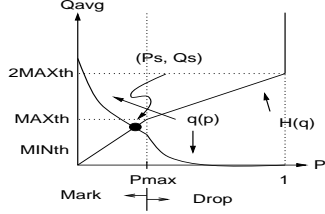


Fig. 3. Equilibrium point of TCP($\alpha_{ECN}, \beta_{ECN}$) and RED-ECN

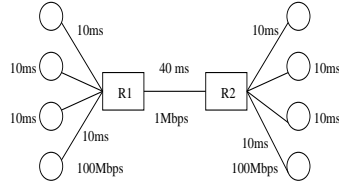


Fig. 4. Simple wide area network configuration

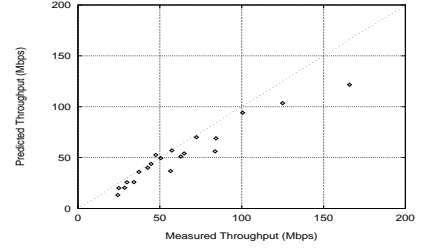


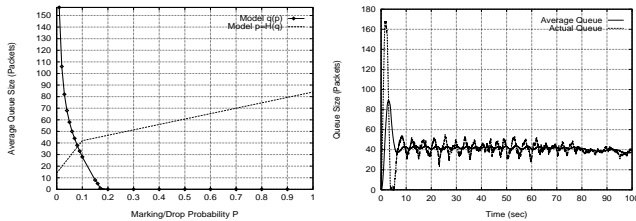
Fig. 5. Predicted versus simulated throughputs

D. Prediction versus Measurement

We run simulations using ns-2.1b6 [29] on the simple configuration shown in figure 4 in order to investigate the effectiveness of the throughput and feedback models proposed in the previous sections. We use 20 unlimited FTP TCP connections with 5 sessions at each sending node. The timer granularity is 100 ms and the segment size is 1000 Bytes. Gentle RED (described in section II-C) is used. A buffer size of 168 kBytes is used where the th_{max} value is 14 kBytes while th_{max} is 42 kBytes. Total simulation time is 100 seconds.

We examine the accuracy of the throughput model for various number of flows (figure 5). The x -axis indicates simulation values while the y -axis represents 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 not too far from the simulated ones.

We also compare the steady-state equilibrium points (p_s, \bar{q}_s) acquired from our model with simulation measurements. Figure 6(a) depicts the steady-state equilibrium point (p_s, \bar{q}_s) predicted by the model. The predicted \bar{q} and $p = H(q)$ from equation (11) 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 $2th_{max}$ used in “gentle” RED. RED-ECN marks if $p < p_{max} = 0.1$ and drops if $p \geq p_{max} = 0.1$. Figure 6(b) illustrates that the actual average RED queue size (from the simulations) lies between 40 and 42 kBytes. The measured packet mark/drop probability over the entire simulation is approximately 8% (not shown). This verifies that our model works well with an active queue management scheme such as RED-ECN.



(a) Prediction of steady-state average queue

(b) Actual average queue size from simulation

Fig. 6. RED Equilibrium point for ($\alpha_{ECN}, \beta_{ECN}$)

V. PERFORMANCE ANALYSIS

We first discuss the simulation setup and the performance metrics, and then analyze the results.

A. Simulation Setup

The network simulator ns-2.1b6 [29] is used in this study. We use the Generic Fairness Configuration-2 (GFC-2), illustrated in Figure 7, which contains multiple bottlenecks and connections with different round-trip times. We use $D = 5$ ms. There are a total of 22 unlimited bulk-data FTP TCP connections in each direction, 6 UDP connections modeled as 0.5 Mbps CBR (Constant Bit Rate), and 22 Web traffic flows. The web traffic is generated using a Poisson Process where the inter-object, inter-page and inter-session times are defined by an exponential distribution. 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 [30]. 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 used is 168 kBytes. The th_{min} value used is $\frac{1}{12} \times$ the buffer size and th_{max} value is $\frac{1}{4} \times$ the buffer size ($3 \times th_{min}$) as recommended by RED designers [30]. We run 5 simulations and average them. We fix the random number generator seed for each run to obtain comparable results when we compare different algorithms.

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 Ratio**: The ratio of dropped packets to the total number of packets sent during the simulation time; and (3) **Response Time (seconds)**: The (mean and maximum) time between when the request of a web client is triggered and when the last requested page from a server arrives at that client.

B. Results and Discussion

Table I shows the performance of TCP-Reno without ECN, with ECN, and ECN(α, β) for the GFC-2 configuration. We

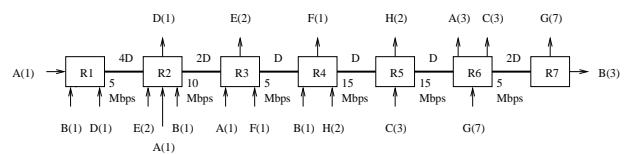


Fig. 7. The Generic Fairness Configuration-2

use α_{ECN} of 0.2 and β_{ECN} of 0.875 [22] in this set of simulations. From the table, it is clear that $ECN(\alpha, \beta)$ exhibits better performance than TCP-Reno without ECN or with ECN. $ECN(\alpha, \beta)$ total goodput (HTTP, FTP and UDP) is around 4 Mbps higher than TCP-Reno without ECN and around 3 Mbps higher than TCP-Reno with ECN. In addition, $ECN(\alpha, \beta)$ exhibits the shortest mean response time for web traffic. With respect to packet drop ratio, $ECN(\alpha, \beta)$ achieves significant improvement over TCP-Reno without ECN and with ECN for various traffic types.

B.1 Responsiveness of $ECN(\alpha, \beta)$

Although $ECN(\alpha, \beta)$ achieves higher performance than TCP Reno without ECN or with ECN as shown above, the less dramatic reduction of congestion window as a response to ECN in $ECN(\alpha, \beta)$ may cause slow responsiveness to a sudden surge of traffic. Recently, equation-based congestion control such as TFRC [21] has been shown to respond adequately to dynamic congestion because of the self-clocking mechanism [24]. In order to investigate the responsiveness of $ECN(\alpha, \beta)$, 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 queue at the links from R2 and from R3 are shown in figure 8 for the 3 algorithms (in separate simulation runs). We do not observe significantly more high values with $ECN(\alpha, \beta)$ than with ECN during the period of sudden congestion. Table II indicates that $ECN(\alpha, \beta)$ outperforms TCP Reno without ECN and with ECN in this scenario as well for $(\alpha, \beta) = (0.2, 0.875)$.

B.2 Effect of α_{ECN} and β_{ECN}

This section explores various pairs of $(\alpha_{ECN}, \beta_{ECN})$ values to investigate their effects on performance. All parameter values other than α_{ECN} and β_{ECN} are set as before. All the $(\alpha_{ECN}, \beta_{ECN})$ pairs used follow the rule proposed in [22] except for (1,0.9). We choose (1,0.9) to compare values that follow the rule in [22] with a more aggressive $(\alpha_{ECN}, \beta_{ECN})$ pair. From table III, we see that (0.1, 0.9355), (0.2, 0.875), (0.4, 0.7647) and (0.8, 0.5789) show marked goodput improvement over (1, 0.5) (with (0.2, 0.875) performing best). $(\alpha_{ECN}, \beta_{ECN})$ values that follow the TCP-friendly rule achieve good performance in terms of goodput as well as Web response time. Although (1, 0.9) exhibits competitive performance, its aggressiveness may be harmful to other TCP connections.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have modified the TCP response to ECN marks using 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 the throughput and steady-state drop/mark probability to consider the modified TCP response to ECN, verify that our model is accurate. Our simulation results in a multiple bottleneck scenario with bulk FTP, bursty web connections and UDP have shown that our ECN response strategy does indeed reduce fluctuations, improve goodput, and reduce delay, thus providing in-

centives 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 the optimum performance.

A number of issues remain to be studied. In terms of evaluating the effect of the increase and decrease parameters α and β , Floyd [22] shows that the bandwidth ratio of AIMD with (α, β) is $\frac{\sqrt{1+\beta} \times \sqrt{\alpha}}{\sqrt{3 \times (1-\beta)}}$ times the goodput of AIMD with (1,0.5). The drop ratio of AIMD with (α, β) is also shown to be $\frac{\alpha \times (1+\beta)}{3 \times (1-\beta)}$ times packet drop ratio of AIMD with (1,0.5). We will model $ECN(\alpha, \beta)$ similarly and compare its behavior to $ECN(1,0.5)$. We are also evaluating the use of feedback information to adapt the traffic conditioner at a differentiated services ingress edge router [26].

REFERENCES

- [1] V. Jacobson, "Congestion avoidance and control," in *Proceedings of the ACM SIGCOMM*, August 1988, vol. 18, pp. 314–329, <ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z>.
- [2] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP," in *ACM Computer Communication Review*, July 1996, vol. 26, pp. 5–21, <ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z>.
- [3] L. Brakmo, S. O'Malley, and L. Peterson, "TCP vegas: New techniques for congestion detection and avoidance," in *Proceedings of the ACM SIGCOMM*, August 1994, pp. 24–35, <http://netweb.usc.edu/yaxu/Vegas/Reference/vegas93.ps>.
- [4] J. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," in *Proceedings of the ACM SIGCOMM*, August 1996, pp. 270–280, <http://www.acm.org/sigcomm/ccr/archive/1996/conf/hoeh.ps>.
- [5] M. Mathis and J. Mahdavi, "Forward acknowledgment: Refining TCP congestion control," in *Proceedings of the ACM SIGCOMM*, August 1996. Also see <http://www.psc.edu/networking/papers/papers.html>.
- [6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993, <ftp://ftp.ee.lbl.gov/papers/early.ps.gz>.
- [7] D. Lin and R. Morris, "Dynamics of random early detection," in *Proceedings of the ACM SIGCOMM*, September 1997, vol. 27, pp. 127–136.
- [8] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proceedings of the IEEE INFOCOM*, March 1999.
- [9] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for web traffic," in *Proceedings of the ACM SIGCOMM*, August 2000.
- [10] C. Liu and R. Jain, "Improving explicit congestion notification with the mark-front strategy," *Submitted to Computer Networks*, May 2000, <http://www.cis.ohio-state.edu/~jain/papers/ecnfront.htm>.
- [11] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, October 1994, <http://www.aciri.org/floyd/>.
- [12] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, April 1999, Also see <http://tcpsat.lerc.nasa.gov/tcpsat/papers.html>.
- [13] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP," RFC 2481, January 1999.
- [14] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *Journal of Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, June 1989, http://www.cis.ohio-state.edu/~jain/papers/cong_av.htm.
- [15] 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.
- [16] 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*, vol. 8, no. 1, pp. 87–98, February 2000, <http://www.cis.ohio-state.edu/~jain/papers/erica.htm>.
- [17] W. Feng, D. Kandlur, D. Saha, and K. Shin, "BLUE: A new class of active queue management algorithms," U. Michigan CSE-TR-387-99, April 1999, <http://www.eecs.umich.edu/~wuchang/blue/>. Also appears in NOSSDAV 2001.
- [18] U. Ahmed and J. Salim, "Performance evaluation of explicit congestion notification (ECN) in IP networks," <http://www7.nortel.com:8080/CTL/ecnperf.pdf>, 2000.

TABLE I
PERFORMANCE OF ECN(α, β)

Algorithm	HTTP Response Time		Goodput			Packet Drop Ratio			
	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

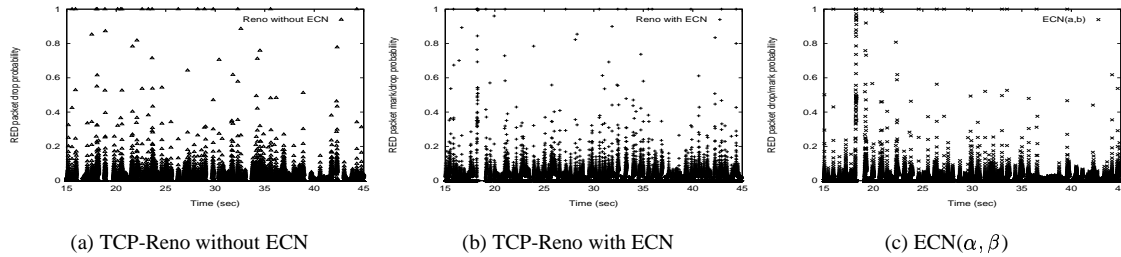


Fig. 8. Drop or mark ratio for responsiveness of ECN(α, β)

TABLE II
RESPONSIVENESS OF ECN(α, β)

Algorithm	HTTP Response Time		Goodput			Packet Drop Ratio			
	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

TABLE III
EFFECT OF α_{ECN} AND β_{ECN} IN ECN(α, β)

$(\alpha_{ECN}, \beta_{ECN})$	HTTP Response Time		Goodput			Packet Drop Ratio			
	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
(0.1, 0.9355)	11.494	52.145	1.745	2.909	41.990	5.339	3.279	0.125	0.707
(0.05, 0.9672)	11.464	48.681	1.350	2.943	40.749	5.566	2.157	0.106	0.516
(1, 0.9)	12.060	52.623	1.665	2.733	42.400	6.664	9.013	0.386	1.773

- [19] T. Ott, "ECN protocols and the TCP paradigm," [ftp://ftp.research.telcordia.com/pub/tjo/ECN.ps](http://ftp.research.telcordia.com/pub/tjo/ECN.ps), May 1999.
- [20] Y. Yang and S. Lam, "General AIMD congestion control," Tech. Rep. UTCS-TR-2000-09, The University of Texas at Austin, May 2000, <http://www.cs.utexas.edu/users/lam/NRL/TechReports/>.
- [21] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of the ACM SIGCOMM*, August 2000.
- [22] S. Floyd, M. Handley, and J. Padhye, "A comparison of equation-based and AIMD congestion control," <http://www.aciri.org/tfrc/>, May 2000.
- [23] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of the ACM SIGCOMM*, September 1998, vol. 28, pp. 303–314, <http://gaia.cs.umass.edu/>.
- [24] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic behavior of slowly-responsive congestion control algorithms," in *Proceedings of the ACM SIGCOMM*, August 2001, http://www.acm.org/sigcomm/sigcomm2001/technical_program.html.
- [25] T. Hamann and J. Walrand, "A new fair window algorithm for ECN-capable TCP (New-ECN)," in *Proceedings of the IEEE INFOCOM*, March 2000.
- [26] A. Habib, S. Fahmy, and B. Bhargava, "Design and evaluation of an adaptive traffic conditioner for differentiated services networks," in *Proceedings of IEEE ICCCN*, October 2001.
- [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*, vol. 5, pp. 336–350, June 1997.
- [28] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proceedings of the IEEE INFOCOM*, March 2000, <http://www.ieee-infocom.org/2000/papers/405.pdf>.
- [29] UCB/LBNL/VINT groups, "UCB/LBNL/VINT Network Simulator," <http://www.isi.edu/nsnam/ns/>, May 2001.
- [30] S. Floyd, "RED web page," <http://www.aciri.org/fby/red.html>, August 2001.