

A Simulation Study of the Interoperability of TCPW with RED and ECN

YUXIANG SHAO, M.Y. SANADIDI, MARIO GERLA

Department of Computer Science
University of California, Los Angeles, CA 90095
{seanshao, medy, gerla}@cs.ucla.edu

Keywords: TCP Westwood, Random Early Detection, Explicit Congestion Notification

Abstract

Active Queue Management (AQM) schemes aim to stabilize router queue sizes and provide better fairness among a set of TCP connections. Random Early Detection (RED) is one of the first and well-known AQM schemes. Explicit Congestion Notification (ECN) is a modified version of RED in that it notifies a sender about congestion, instead of dropping its packet. Under such AQM schemes, TCP connections have been shown to perform better in terms of fairness and stability. In this paper, we study how TCP Westwood interoperates with RED and with ECN. We first present various simulations of TCPW with RED and compare that to NewReno with RED. From our study we conclude that, TCPW benefits at least as well as NewReno from RED. Second, we discuss the potential use of ECN in TCPW, and the simulation results reveal that ECN is able to reduce packet delay, and to give significant improvement in TCPW's friendliness with NewReno. Also, we present a case study in which how we use ECN information to help TCPW make better estimate about congestion level.

1. INTRODUCTION

Congestion control has been studied extensively in computer networks. There are mainly two components to perform the task of congestion control: Active Queue Management (AQM) schemes at network routers, and congestion control schemes at sender/receiver nodes. In this paper, we will study the interaction between a recently proposed TCP protocol, TCPW, and an AQM scheme, Random Early Detection (RED), and RED's enhancement, Explicit Congestion Notification (ECN), respectively.

Many TCP variants were proposed and their performance analyzed over the last two decades. Most of the TCP protocols such as TCP Tahoe, TCP Reno and NewReno reduce their sending window ($cwin$) when the sender perceives possible packet loss [18]. The reduction of $cwin$ upon packet loss is by half in case of three duplicate ACKs, or to $cwin=1$ in case of Timeout. TCP Westwood (TCPW) has been proposed to enhance the congestion control function by estimating an eligible sending rate or bandwidth share for the connection, and setting the congestion control parameters accordingly [18].

Network layer schemes, providing support of the congestion control functions in TCP, have been introduced and studied extensively. The most important class of such schemes is perhaps the AQM schemes. AQM aims to improve overall network bandwidth utilization, smooth buffer occupancy in routers, and provide better fairness among transport layer connections. The earliest and most prominent of the AQM schemes is RED. In RED, packets are randomly dropped from the router buffer when the

occupancy of the buffer exceeds certain thresholds. In Explicit Congestion Notification (ECN), routers notify a sender explicitly of congestion, instead of the implicit notification resulting from a packet drop. Because of the explicit congestion notification in ECN, the sender ability in discriminating the cause of packet losses is indeed enhanced.

In the first half of this paper, we investigate the interoperability of TCPW and RED, as RED has been designed with TCP Reno (and therefore NewReno) in mind, and it was shown that the latter benefited from the support of RED [5]. We evaluate in this paper how TCPW benefits from RED and compare to NewReno. As comparison criteria, we consider fairness of the protocols with and without RED. In the second half, we investigate whether TCPW, when coupled with ECN, can make use of the explicit notification to discriminate the cause of packet loss, and quantify the benefits resulting from ECN.

The rest of paper is organized as follows. In Section 2, we describe the various versions of TCPW we used in our study. Section 3 describes the AQM schemes, RED and ECN, and explains their benefits. In Section 4, we present experimental setup and simulation results of the interaction between TCPW and RED. Section 5 evaluates the impact of coupling ECN and TCPW. Finally, we give our conclusion in Section 6 and propose discuss future research in Section 7.

2. TCP WESTWOOD

TCP is designed to provide reliable connection-oriented transport layer service [10]. TCP uses a window mechanism to provide congestion control. A TCP congestion control scheme is in one of two phases. In the first phase, called slow start, the congestion window ($cwin$) begins with 1 segment and doubles every round trip time as long as ACKs are received indicating that no packets have been lost. If packet loss is detected during slow start, then the slow start phase will be restarted at $cwin=1$. Exponential $cwin$ increase continues in slow start until $cwin$ hits a threshold called slow start threshold ($ssthresh$). The connection is then said to enter Congestion Avoidance, where $cwin$ will increase linearly until congestion is detected again, either by a Timeout, or by the reception of three duplicate ACKs (DUPACKs). Except the above two phases, some TCP, such as TCP NewReno, also uses the Fast-Retransmit and Fast-Recovery mechanisms [1] [6] [7] [9] [12].

TCP Westwood (TCPW) is a new version of TCP that enhances the window congestion control and its back-off process [2], [18], [19], and [20]. The TCPW version used in the RED simulations in this paper is TCPW-BE [2], and the version used in the ECN study is TCPW-ABSE [19].

2.1. TCPW-BE

For TCPW-BE, a TCPW sender monitors the acknowledgment reception rate and from it estimates the data packet rate currently achieved by the connection. Whenever a sender perceives a packet loss (e.g., via 3 duplicate ACKs), the sender uses a Bandwidth Estimate (BE) to properly set the congestion window ($cwin$) and the slow start threshold ($ssthresh$). The bandwidth estimate is based on ACK rates and information they carry regarding delivered data bytes to the destination. By backing off to $cwin$ and $ssthresh$ values that are based on the estimated bandwidth (rather than simply halving the current values as NewReno does), TCPW avoids overly conservative reductions of $cwin$ and $ssthresh$; and thus gives a faster recovery. Experimental studies reveal the benefits of the intelligent backoff strategy in TCPW: better throughput, goodput, and delay performance, as well as fairness. Most importantly, TCPW is very effective in handling wireless loss. This is because TCPW uses the current estimated rate as reference for resetting the congestion window. The current rate is only marginally impacted by loss (as long as the loss is a relatively small fraction of data rate) [18].

2.2. TCPW-ABSE

Using TCPW-BE provides significantly higher utilization than Reno. However, under certain conditions, TCPW-BE overestimates a connection fair share, and can be unfriendly to Reno connections. To address this problem, Adaptive Bandwidth Share Estimates was introduced in TCPW-ABSE [19]. This scheme uses an adaptive sampling method, in which the time interval T_k associated with the k_{th} received ACK is appropriately chosen between the last ACK interarrival time and the latest Round Trip Time estimate (RTT). A larger value of T_k is used when the congestion level is high resulting in a more conservative bandwidth share or eligible rate estimate. When the congestion level is detected to be low, a smaller value of T_k is used resulting in a more aggressive eligible rate estimate. The congestion level indicator is similar to that used in TCP Vegas [1]. That is the difference between the rate a connection expects to receive and what it actually achieves in a RTT. ECN, with its provision of explicit congestion notification can also be used in determining the existence of congestion losses. We will evaluate the benefit that can be obtained from ECN in this context.

3. AQM SCHEMES

3.1. Random Early Detection (RED)

To support end-to-end TCP congestion control schemes, several active queue management (AQM) at routers have been proposed [13]. In RED, the router drops randomly chosen packets whenever its queue length is deemed long, running the risk of uncontrolled packet losses [5]. By intentionally dropping packets chosen at random, packet losses are distributed over the set of connections sharing a link.

Two thresholds are maintained, and packets may be dropped with various probabilities depending on the queue length in relation to the two thresholds. When the queue size is less than a minimum threshold ($minthresh$), no packets will be dropped. When the queue size is between $minthresh$ and a maximum

threshold ($maxthresh$), the drop probability varies (linearly according to earlier proposals) between 0 and a maximum drop probability (max_p). Max_p is generally a small number, by default equal to 0.03. If the queue size exceeds $maxthresh$, all packets will be dropped. There are three major benefits related to RED: 1) Decreasing the end-to-end delay for both “responsive” (TCP) and “not responsive” real-time traffic (e.g. UDP), 2) Preventing large number of consecutive packet losses by a single connection by ensuring some buffer space availability, even with bursty traffic, and 3) removing the higher loss bias against bursty traffic observed with Drop Tail [4] [13] [15].

3.2. Explicit Congestion Notification (ECN)

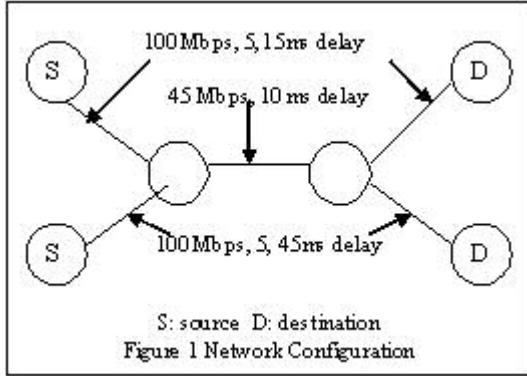
ECN [16] is an enhancement of RED. An ECN-capable router now only “marks” the packets that are supposed to be dropped, and still forwards them to the end nodes.

The way ECN works is as follows: as the average queue length exceeds a certain limit ($minthresh$), the router will set the Congestion Experienced (CE) bit in the outgoing packets probabilistically. The probability of marking an outgoing packet increases linearly as the average queue length at the router increases. When the average queue length gets past the value of $maxthresh$, each and every outgoing packets will have the CE bit set (i.e. the probability of marking is 1). When a receiver gets such packets, it responds by setting the ECN-Echo bit in the returning ACK packets. The receiver sets the ECN-Echo bit on every ACK after it sees one packet with the CE bit set. Finally, after the sender gets *one* ACK with ECN-Echo, it will react by reducing its congestion window. It will also set the Congestion Window Reduced (CWR) bit of its next outgoing packet to inform the receiver that actions have been taken against the congestion indicated.

One advantage of using ECN is improving the delay of individual packets by avoiding packet losses. Whilst individual packet losses may not have significant impact on bulk data traffic which carries hundreds of thousands of packets, such losses result in noticeable delay in interactive traffics such as TELNET [3]. By deploying ECN in the network, better and smoother TELNET sessions can be supported. ECN also improves the fairness of TCP by avoiding packet drops from queue tails whenever the buffer overflows, which are typical of Drop-Tail routers. In addition, ECN can improve the overall throughput of those TCP connections with coarse retransmit timers. By avoiding packet losses and thus reducing the number of timeouts, we can cut down the amount of time that the links become idle due to a long retransmit timeout.

4. TCPW AND RED

All the simulations below contain at least two senders and at least two receivers, all sharing a bottleneck link (Figure 1). The main system parameters considered in this study are: (1) Buffer size at the router, (2) Round trip propagation time, RTT, and (3) Number of connections.



We setup two simulation scenarios:

- (1) Two connections running the same TCP variant, and traversing a bottleneck link, but with different end-to-end propagation times. We study fairness of TCP NewReno and of TCPW in this context.
- (2) Ten connections with identical propagation times and using the same TCP variant; we report on fairness as measured by Jain's index [11], and the total throughput of all the connections.

Our simulations were done in ns-2 [14], and the source code of TCPW (BE) is obtained from [17]. We report the results of scenarios 1 and 2 in sections 4.1 and 4.2, respectively.

4.1. Long RTT vs. Short RTT Connection

In this case we run two connections with the same TCP variant, but with different propagation times. The parameters used in this scenario are: 45Mbps bottleneck link, 40ms as short delay, and 200ms as long delay. We compare the performance of TCPW and NewReno with and without RED. The RED parameters we used are $min_{th} = 50\%$ of the queue size, $max_{th} = 80\%$, and $max_p = 0.1$.

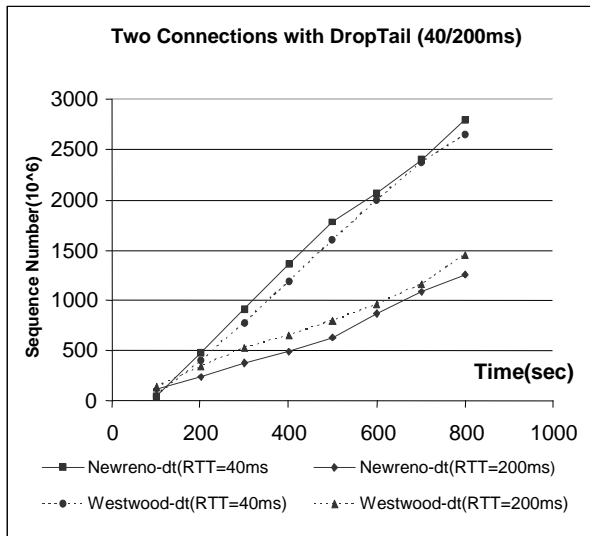


Figure 2. Long Short Connection without RED

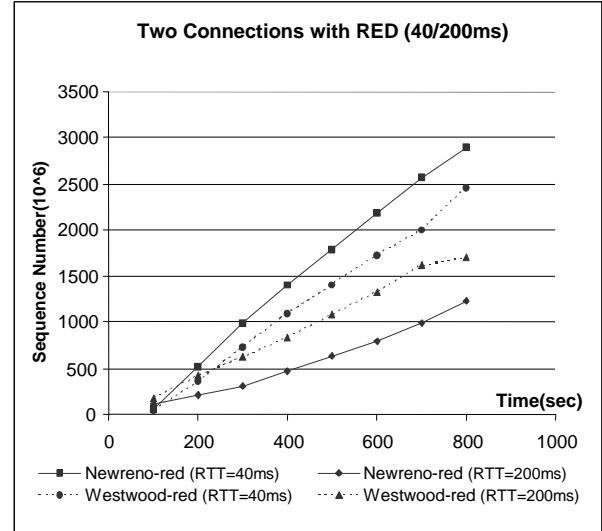


Figure 3. Long Short Connection with RED

In Figures 2 and 3, the Y-axis is in sequence number (per one million), and the X-axis is time in seconds.

From Figures 2 and 3 we can see that, with or without RED, TCPW shows more fairness than NewReno. Also, with RED, TCPW gains in fairness over Droptail. In fact, for all buffer sizes we found that TCPW provides better fairness than NewReno in this long - short propagation time scenario.

4.2. Multiple connections with same RTT

We run $N=10$ connections of the same TCP variant. The parameters used in this scenario are: 45-Mbps bottleneck; RTT is the same for all connections. We compare the performance of TCPW and NewReno with and without RED. The RED parameters we used here are: $min_{th} = 10\%$, $max_{th} = 25\%$, and $max_p = 0.1$. These parameters are adjusted relative to the setting in Section 4.1 experiments due to the larger number of connections in the current set.

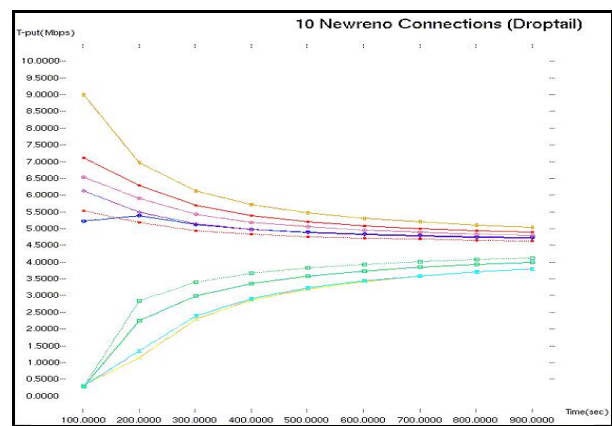


Figure 4. 10 NewReno connections with Droptail

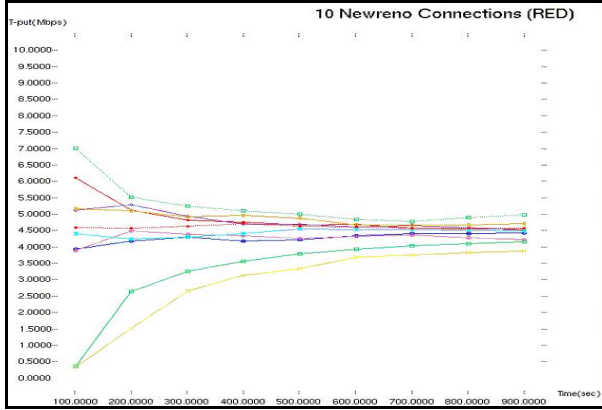


Figure 5. 10 NewReno connections with RED

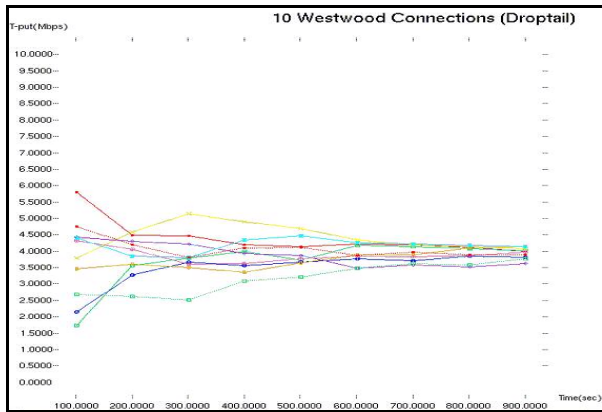


Figure 6. 10 TCPW connections with Droptail

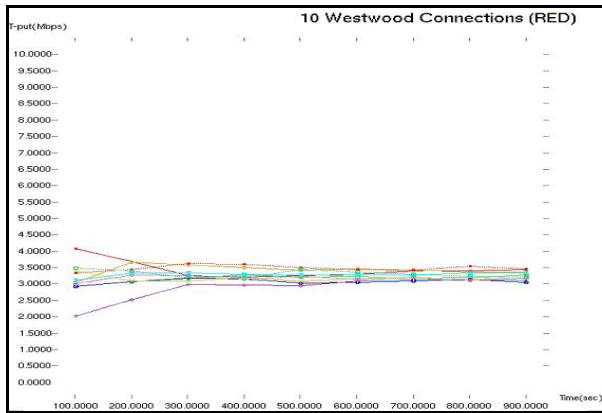


Figure 7. 10 TCPW Connections with RED

Figures 4-7 show the 10 TCP connections with Droptail and with RED. The buffer size of this case is equal to pipe size – bottleneck link speed times round trip time, in units of packets. Figures 4 and 5 are for NewReno, and Figures 6 and 7 are for TCPW. For both protocols, RED helps them to gain more fairness over Droptail. This is because when RED is present, it prevents some early-start connections from capturing the link.

Table 1. 10 TCP connections with and without RED

10 TCP	Newreno		Westwood	
	T-put (Mbps)	Jain's Index	T-put (Mbps)	Jain's Index
RED	4.486	0.960	4.439	0.999
DropTail	4.490	0.722	4.491	0.984

Table 1 shows summary of results for 10 TCPW connections and 10 NewReno connections. The throughput was taken from average among the connections. Since we are using 45Mbps bottleneck, so the optimal throughput for each connection should be 4.5Mbps. As we can see, TCPW has a bit less throughput than NewReno with RED, but it has better fairness than NewReno.

5. TCPW AND ECN

In this section we describe the configuration of our experiment and present the simulation results. The setup is similar to the simulation in RED, with multiple senders and receivers sharing a bottleneck link.

We setup three simulation scenarios:

- (1) Two connections, having the same TCP variant, but with different round trip propagation times. We measure the one-way packet delay for both TCPW and NewReno with and without ECN.
- (2) We study the friendliness of TCPW-ABSE to NewReno with and without ECN. Multiple TCP connections are run (10 to 40), all having the same round trip propagation times, 100ms.
- (3) We investigate a scenario in which there is burst errors. We believe that in such condition, using ECN would help TCPW-ABSE make better estimation about the congestion level.

The ECN parameters we used in these two simulations are minth = 50%, maxth = 80%, and max_p = 0.1.

5.1. Packet Delay

There are two connections, the longer RTT is 80ms and the shorter one is 20ms. We measure the one-way packet delay for TCPW-BE with and without ECN. Then we repeat the simulation using NewReno. We define one-way packet delay as the time between when a packet leaves the TCP source and when it arrives at the TCP sink. Figure 8 shows the relations between the average one-way packet delay of TCPW-BE and the buffer size at the bottleneck router. From the graph, we see that ECN significantly reduces the individual packet delay of both long-delay and short-delay TCP connections. In fact, the packet delay of a long-delay connection with ECN is comparable to, and sometimes even lower than (e.g. at a buffer size of 675 packets) the packet delay of a short-delay connection without ECN. Similar observations can be made in Figure 9, which shows the same delay-versus-buffer graph for TCP NewReno. By comparing these two graphs, we can see that TCP NewReno and TCPW get almost the same amount of delay reduction from the use of ECN.

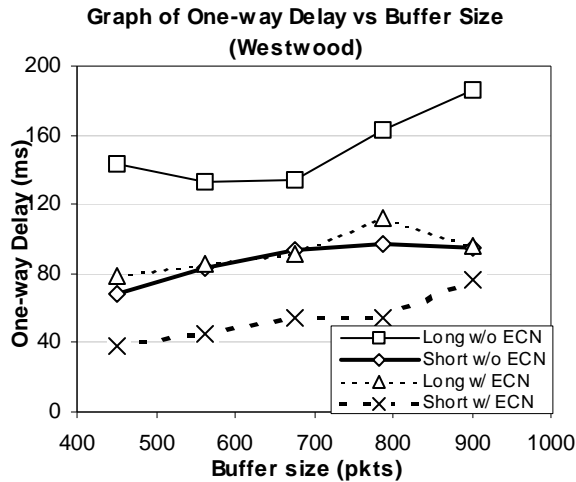


Figure 8. Graph of one-way delay against buffer size for TCP Westwood

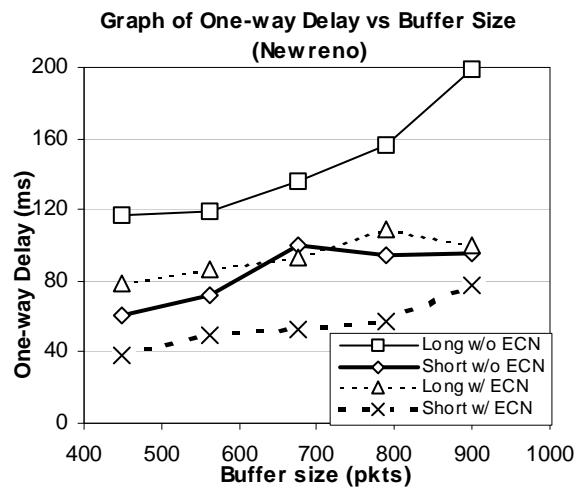


Figure 9. Graph of one-way delay against buffer size for TCP NewReno

5.2. TCPW friendliness to NewReno

The fair share of each NewReno connection can be determined by running a simulation session consisting of only NewReno connections. One way of analyzing the friendliness of TCPW to NewReno is to calculate the percentage change of fair share for each NewReno when half of the connections in the simulation are TCPW [20]. Following this procedure, we obtain the graph in Figure 10, which plots the percentage difference of bandwidth fair share of each NewReno connection against the number of connections in the simulation. We also plot Figure 11, which shows the percentage difference of bandwidth share by each TCPW connection over NewReno when they co-exist in the network. Positive percentage means that the connection has gained bandwidth when it coexists with the other type. The results in the two graphs are encouraging since the presence of ECN has improved friendliness. Without ECN, TCPW-ABSE was “too” friendly to NewReno that NewReno has gained up to 14% in bandwidth share. With ECN, the excessive friendliness has been

decreased. From these two graphs, we may conclude that ECN helps improve TCPW and NewReno friendliness.

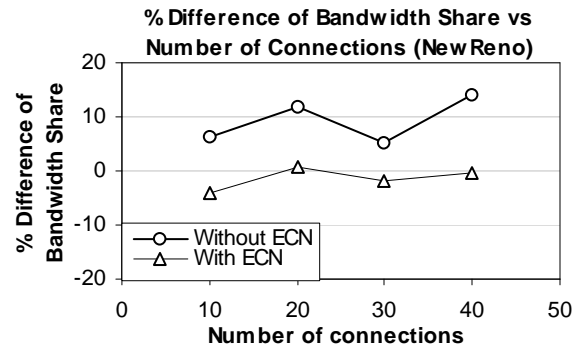


Figure 10. Percentage difference for NewReno

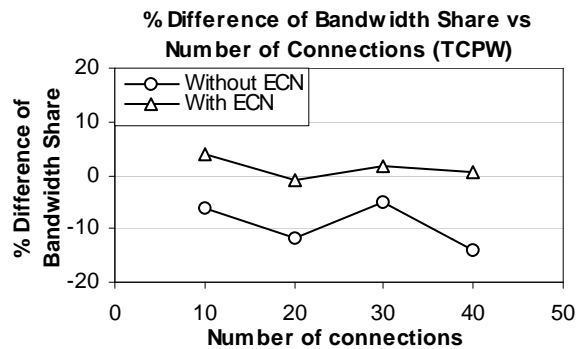


Figure 11. Percentage difference for TCPW-ABSE

5.3. Using ECN to improve TCPW

TCPW-ABSE decides the level of congestion by comparing the values of the expected throughput (Exp) and the achieved throughput (Ach). If $Exp \gg Ach$, then the network is considered congested, and the value of T_k will be larger resulting in a more conservative bandwidth share estimate. ECN provides a different signal to indicate the presence of congestion. One scenario, in which ABSE detects congestion, while ECN indicates the opposite, is in a high error rate or burst error environment. In such cases, a series of dropped packets would cause ABSE to detect congestion (since Exp is much greater than Ach). When no congestion is present, ECN-Echo bits in returning ACKs will indicate no congestion since the packets are actually lost due to errors. We evaluate below a method to modulate the ABSE setting of T_k by the presence of, or lack of presence of ECN echo bits. The method consists of the following modifications to ABSE: if an ACK has ECN-Echo bit set, then the value of T_k produced by ABSE is multiplied by a factor larger than one, since ECN adds to the belief that there is congestion. If there is no ECN-Echo bit, then T_k is multiplied a factor less than one. As a result, in the burst error scenario, although ABSE will detect congestion and set T_k accordingly, the lack of ECN ECHO bits will result in a reduction of T_k bringing the eligible rate estimation higher.

We setup simulation to run 5 connections, all using the same TCP protocol, and measured the total throughput of the five connections. The RTT of the experiment is 70ms, bottleneck link is 45Mbps, and so the ideal utilization would be 45Mbps. In addition,

we varied the error rate of the links to see the effectiveness of using ECN information to modulate ABSE estimates.

Table 2. Total throughput of 5 connections with ECN

error rate (%)	0	0.01	0.1	1
Newreno (ECN)	41.046	41.065	22.642	6.667
ABSE (ECN)	42.692	42.638	42.545	24.072
ABSE (mod)	42.824	42.781	42.746	24.806

Table 2 shows the total throughput of the 5 connections with ECN. The first two rows are the results for running NewReno and ABSE using the original ECN response (cut by half), and the third row is using the modification we have applied on ABSE. Although the improvements are small, the benefit of the modified ABSE becomes more obvious as the error rate goes up. This is due to the more occurrences of the burst error case that caused ABSE to guess wrong, but ECN is able to correct it. The limitation of the improvement is because burst errors lead to timeouts, and thus high utilization of the link is difficult to achieve. Furthermore, in a high-error-rate environment, persistent burst errors would eventually cause the estimation of bandwidth to drop to significantly, and ECN information is into sufficient to address the problem. Other modifications in TCP may still be required for higher error rates than approximately 2% [21].

6. CONCLUSION

We evaluated the interoperability of TCPW and RED and compared it to that of NewReno and RED. We considered as comparison criteria the efficiency (throughput) and fairness (using Jain's Index). We found that TCPW and RED interoperate well in that TCPW behavior is improved with the presence of RED. In comparison to NewReno, we determined that TCPW exhibits better fairness in cases of connection with asymmetric propagation time, as well as for symmetric connections.

We then evaluated the effect of ECN on TCPW. First, TCPW benefits from ECN as packet delays are significantly reduced. The delay reduction is similar to the reduction of NewReno packet delays when ECN is used. Second, we found that TCPW may be "too friendly" to NewReno when the network router is of type Droptail. By switching to ECN, TCPW is no longer "too friendly". Finally, we studied a method in which ECN could help TCPW in making better estimate of eligible rate estimation, ECN does help TCPW throughput as the error rate rises, but the improvement is very limited.

In summary, TCPW interoperability with RED/ECN is shown above to be as beneficial as NewReno interoperability with RED/ECN.

7. FUTURE WORK

To better infer the reason of packet loss (congestion versus random error), we could gather statistics of ECN instances received over a period of time. Since ECN-capable routers set the Congestion Experienced bit probabilistically, such statistics could allow a more fine-grained inference of the degree of congestion. For example, if we receive a lot of ECN-Echoes over a short period of time, then we can claim with a higher degree of confidence that future packet losses are due to congestion.

Furthermore, for both RED and ECN, there does not exist a proper tuning of the parameters [8]. We can see drastic performance difference when we vary the parameters. Hence, it

would be good to establish a standard way of assigning those parameters under various environments.

REFERENCES

- [1] T. Bonald. "Comparison of TCP Reno and TCP Vegas: Efficiency and Fairness", in *Proceedings of PERFORMANCE'99*, Istanbul, Turkey, October 1999.
- [2] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, R. Wang. "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", in *Proceedings of Mobicom 2001, Rome, Italy*. July 2001.
- [3] S. Floyd. "TCP and Explicit Congestion Notification", *ACM Computer Communication Review*, 24(5): 10–23, October 1994.
- [4] V. Firoiu, M. Borden. "A Study of Active Queue Management for Congestion Control", in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Tel Aviv, Israel), Mar. 2000.
- [5] S. Floyd, V. Jacobson. "Random Early Detection gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, V.1 N.4, p. 397-413, August 1993.
- [6] M. Gerla, R. Lo Cigno, S. Mascolo, W. Weng. "Generalized Window Advertising for TCP Congestion Control," UCLA CSD TR#990012, February 1999.
- [7] T. Goff, J. Moronski, D. S. Phatak, V. Gupta. "Freeze-TCP: a True End-to-end TCP Enhancement Mechanism for Mobile Environments", in *Proceedings of IEEE INFOCOM'2000*, Tel Aviv, Israel, March 2000.
- [8] V. Misra, W. Gong, D. Towlsey. "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED", in *Proceedings of SIGCOMM 2000, Stockholm, Sweden*, September 2000, p. 151-160.
- [9] U. Hengartner, J. Bolliger, T. Gross. "TCP Vegas Revisited", in *Proceedings of IEEE INFOCOM'2000*, Tel Aviv, Israel, March 2000.
- [10] V. Jacobson. "Congestion Avoidance and Control", *ACM Computer Communications Review*, 18(4): 314–329, August 1988.
- [11] Jain's fairness index. URL: <http://www.cs.berkeley.edu/~kfall/EE122/lec21/lec21.pdf>.
- [12] L. Kalampoukas, A. Varma, K. K. Ramakrishnan. "Explicit Window Adaptation: A Method to Enhance TCP Performance", in *Proceedings of IEEE INFOCOM'98*, San Francisco, Ca, USA, March/April 1998.
- [13] M. May, T. Bonald, J. Bolot. "Analytic Evaluation of RED Performance", *infocom 2000*: 1415-1424.
- [14] NS-2 network simulator (version 2) LBL, URL: <http://www.isi.edu/nsnam/ns/>.
- [15] B. Braden, et al.. "Recommendations on Queue Management and Congestion Avoidance in the Internet", *RFC2309*, April 1998.
- [16] K. Ramakrishnan, S. Floyd, D. Black. "The Addition of Explicit Congestion Notification (ECN) to IP", *RFC3168*, September 2001.
- [17] TCP Westwood modules for ns-2. URL: <http://www.cs.ucla.edu/NRL/hpi/tcpw/index.html>.
- [18] S. Mascolo, C. Casetti, M. Gerla, S. S. Lee, M. Sanadidi. "TCP Westwood: congestion control with faster recovery", UCLA CSD TR#200017, 2000.
- [19] R. Wang, M. Valla, M. Y. Sanadidi, M. Gerla. "Adaptive Bandwidth Share Estimation in TCP Westwood", in *Proceedings IEEE Globecom 2002*, Taipei, Taiwan, R.O.C., November 17-21, 2002.
- [20] R. Wang, M. Valla, M.Y. Sanadidi, B. K. F. Ng, M. Gerla. "Efficiency/Friendliness Tradeoffs in TCP Westwood", *Seventh IEEE Symposium on Computers and Communications*, Taormina, Italy, 1 - 4 July 2002.
- [21] G. Yang, R. Wang, M. Y. Sanadidi, and M. Gerla, "TCP with Bulk Repeat in Next Generation Wireless networks", in *Proceedings of IEEE ICC'2003*, Anchorage, AK, USA, May, 2003.