Performance Study of TCP Veno over WLAN and RED Router

Qixiang Pang[1], Soung C. Liew[1], Cheng Peng Fu[2], Wei Wang[1], Victor O.K. Li[3]

[1]Department of Information Engineering
The Chinese University of Hong Kong
Hong Kong

[2]Division of Computer Communications
School of Computer Engineering
Nanyang Technological University
Singapore

[3]Department of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong

Abstract—**This paper examines the impact of RED on two versions of TCP – traditional TCP Reno and a newly proposed variant, TCP Veno - over 802.11b WLAN. TCP Reno was originally designed for wired networks where packet losses are primarily due to network congestion. This assumption is not always true in wireless networks, in which packet losses can be due to transmission errors on the noisy wireless link. TCP Veno refines the algorithms in Reno by distinguishing between noncongestive and congestive states, and avoids the unnecessary reduction of TCP congestion window when packet losses are not due to congestion. Our results show that TCP Veno can achieve up to 30% more throughput than TCP Reno when link quality is poor. Our results also show that TCP Veno is compatible with RED. In addition, although RED does not help to further improve the throughput in Veno, it can improve fairness among co-existing TCP flows.**

Keywords-TCP; 802.11; RED; performance

I. INTRODUCTION

Several wireless technologies, such as Bluetooth, 3G, and WLAN, have been targeted for data transmission. In particular, WLAN equipment based on the IEEE 802.11 standards has seen astounding success due to its many advantages, including easy installation, low cost and high data rates [1][2].

Wireless networking technologies share the common problem that they generally suffer from much higher link error rate than wired networks. Reliable transport protocols such as TCP assume congestion in the network to be the primary cause of packet losses [3]. TCP reacts to all packet losses by reducing the rate at which it injects traffic into the network. Unfortunately, when packets are lost for reasons other than congestion - e.g. due to transmission error in the wireless link - this congestion control measure results in unnecessary end-to-end throughput reduction.

In [4], a new TCP variant called TCP Veno was proposed to solve the problem. TCP Veno defines and distinguishes between two states of operation: congestive and noncongestive states. When a packet loss is detected while the TCP connection is in the noncongestive state, the input traffic rate is not reduced as much as when the loss is detected during the congestive state. Compared with other TCP variants that also improve throughput performance [5][6][7], a distinguishing feature of TCP Veno is that it only requires modification of the sender-side protocol stack, making it easier to deploy over the current Internet.

However, the performance of Random Early Detection (RED) when combined with TCP Veno was not studied in the earlier work in [4].

The traditional queue management algorithm, "tail drop", drops packets only when the router buffer is full. On the other hand, the RED algorithm drops arriving packets probabilistically when the buffer is about to overflow. The goal is to give the TCP connections an early indication of impending congestion. It is claimed in [8] and RFC2309 [9] that RED can avoid lock-out and global synchronization phenomena, thereby improve fairness and throughput.

An interesting issue is whether RED works well with TCP Veno traffic in the environment of 802.11b WLAN. Although many papers have investigated RED in wired networks, there have been few experiments investigating the compatibility between RED and TCP Veno, particularly in the WLAN environment. This paper is an attempt toward this direction.

This paper also presents TCP Veno results obtained under different experimental settings than in [4], in which TCP Veno was originally proposed. In particular, the results in this paper are obtained based on an 802.11b (up to 11 Mbps) experimental WLAN platform, as opposed to [4], in which earlier-generation WLAN (2 Mbps data rate) was used.

The remainder of this paper is organized as follows. In Section II, we review 802.11 WLAN, TCP Reno and Veno, and the RED algorithm. In Section III, the set-up for our WLAN experiments is described. In Section IV, the experimental results are presented and interpreted. Section V concludes this paper.

II. TCP OVER WLAN AND RED ROUTER

A. *802.11 WLAN*

The IEEE 802.11 standards specify the Media Access Control (MAC) and physical (PHY) layers for WLAN.

One reason why WLAN has been receiving so much attention is that it works at the unlicensed ISM frequency bands. Within 802.11, the most successful standard so far is 802.11b, which operates at 2.4 GHz and provides up to 11 Mbps data rate.

This work is sponsored by the Areas of Excellence scheme established under the University Grant Committee of the Hong Kong Special Administrative Region, China (Project Number AoE/E-01/99).

802.11 defines two types of network architectures: independent or ad hoc networks, and infrastructure or client/server networks. In infrastructure networks, an access point relays the frames between the stations, or between the stations and wired backbone. Due to overhead, link error, and station collisions, the actual effective bandwidth of 802.11b is much lower than 11 Mbps.

B. TCP Reno and Veno

TCP is a reliable connection-oriented protocol that implements flow control by means of a sliding window algorithm [3]. TCP Reno, which makes use of slow start and congestion avoidance algorithms to adjust the window size, is widely deployed in the Internet. During the slow start phase, its window is incremented for each *ack* received until packet loss is experienced, at which point the window is halved and then a linear increase algorithm takes over until further packet loss is experienced. This additive increase and multiplicative decrease mechanism leads to periodic oscillations in the congestion window, round trip delay and queue length of the bottleneck buffer in the path.

However, the assumption in TCP Reno that packet loss implies network congestion may not apply to wireless networks, in which packet loss may be induced by noise, link error or reasons other than network congestion. Not making an attempt to distinguish between random and congestion losses, TCP Reno is equally sensitive to both of them. This may lead to significant but unnecessary end-to-end throughput degradation.

In [4] an end-to-end congestion control mechanism, called TCP Veno, was proposed. It integrates the advantages of both TCP Reno and Vegas. In contrast to many other complex solutions described in [6], TCP Veno only requires modification to the TCP sender. This interoperability with the legacy TCP makes it easier to deploy TCP Veno in the current Internet. The algorithm differs from the conventional TCP in two ways: 1) it dynamically adjusts the slow start threshold (*ssthresh*) according to the perceived state of a connection – congestive or noncongestive, as opposed to using a fixed drop-factor when packet loss is encountered; 2) it uses a refined linear additive increase algorithm to adjust the congestion window evolution for congestion avoidance.

In order to distinguish between congestive and noncongestive states, TCP Veno borrows the idea of monitoring the difference between the measured and the expected throughput from TCP Vegas, namely,

$$DIFF = (Expected - Actual) \qquad (1)$$

In the above, *Expected = cwnd/BaseRTT*, where *BaseRTT* is the minimum of all measured RTT (round trip times). It is usually the RTT of the first segment sent by a connection; *Actual* is the measured throughput at the sender given by *cwnd/RTT*, where RTT is the actual round-trip time of a tagged packet. Strictly speaking, *Expected* as defined is the best possible throughput, since *BaseRTT* is the minimum of all measured RTT.

In Veno, *DIFF*BaseRTT* is used to estimate the number of packets accumulated at the bottleneck buffer. If there are more than an upper threshold (β) of packets queuing for processing, the TCP connection is said to have evolved into a congestive state. Otherwise, it is in the non-congestive state. As in TCP Reno, packet loss in the congestive state will cause the window to be halved. However, packet loss in the non-congestive state will only cause the window size to be decreased by a factor of 1/5.

Veno also refines the additive increase phase of Reno by forcing the TCP connection to stay longer at the operating region.

The enhanced algorithms are described in Fig. 1 and Fig. 2.

C. TD and RED Routers

Historically, Internet routers use a TD (Tail Drop) discipline as the buffer management mechanism: the TD router serves incoming packets in order of their arrivals and simply discards newly arriving packets when the buffer is full.

The RED (Random Early Detection) algorithm is designed to detect the beginning of congestion by monitoring the average queue size at the router (the average number of packets in the router buffer) and signals to the TCP senders that congestion has occurred by intentionally dropping packets in a probabilistic manner [8].The RED algorithm sets the packet dropping probability as a function of the average queue size. It uses a low-pass filter with an exponentially weighted moving average to calculate the average queue size avg:

$$avg \leftarrow (1 - w_q)avg + w_q q \qquad (2)$$

where q is the instantaneous queue size.

The packet dropping probability is determined in different ways according to *avg*:

- If $avg < min_{th}$, all arriving packets are accepted.
- If $min_{th} < avg < max_{th}$, arriving packets are dropped with probability $P_{\mathrm{red}}(avg)$, which is defined as follows: $P_{\mathrm{red}}(avg) = max_p * (avg - min_{th})/(max_{th} - min_{th})$
- If $avg > max_{th}$, all arriving packets are dropped.

In recent work [11][12][13] however, many researchers have pointed out that it is not easy to choose the control parameters of RED (max_{th}, min_{th}, max_p, w_q) to work well, and even when the choice of the parameters are optimized, the performance of RED routers is doubtful. The experimental results presented in the following sections help to understand RED performance more thoroughly.

In addition, the observant readers may note that both TCP Veno and RED monitor the buffer occupancy, and make use of that knowledge to attempt to improve throughput performance. One may wonder if the actions of these two mechanisms are complementary, independent, or incompatible. A goal of this paper is to answer this question. Note that while TCP Veno

```
When packet loss is detected by fast retransmit:
   if (DIFF*BaseRTT < β)     //most likely it is a random loss
      ssthresh = cwnd_loss * (4/5)
   else                      //most likely it is a congestion loss
      ssthresh = cwnd_loss / 2

When packet loss is detected by retransmit timeout timer:
    ssthresh is set to half the current window ;
    slow start is performed; //performs the same action as in Reno
```

Figure 1. TCP Veno - ssthresh adjusting algorithm

```
During the additive increase period:
  if (DIFF*BaseRTT < β)    //available bandwidth is underutilized
    cwnd=cwnd+1/cwnd when every new ack is received
  else                     //available bandwidth is fully utilized
    cwnd=cwnd+1/cwnd when every other new ack is received
```

Figure 2. TCP Veno - refined additive increase algorithm.

estimates the backlog of an end-to-end connection at the bottleneck, RED actually monitors the aggregate of the backlogs from all connections traversing the bottleneck.

III. Experimental Network

An experimental network is established to evaluate the performance of TCP over RED router and WLAN. The experimental network makes use of a commercially available 802.11b access point and PCMCIA cards. The components of the experimental network are a data server, a router, an AP, and several laptop stations, as shown in Fig. 3.

A. Server

The server serves as the data sender of both TCP Reno and Veno. The operating system of the server is FreeBSD 4.3. A testing program called *TCPSuiteserver* is implemented in the server to send TCP data to the clients. Correspondingly, the clients run a program called *TCPSuiteclient* to receive the data and collect the experimental results.

B. Router

The router machine runs the Dummynet software on the FreeBSD 4.3 OS (Fig. 4). The operating system has an embedded *ipfw* command to configure the parameters of the forward and the reverse buffers and pipes:

- the forward buffer size, B_f,
- the forward bandwidth, μ_f,
- the forward propagation delay, τ_f,
- the reverse buffer size, B_r,
- the reverse bandwidth, μ_r, and,
- the reverse propagation delay, τ_r.

Random packet drop rates at the router can be artificially induced by the command. A drop-tail or RED buffer management mechanism can be selected using *ipfw*. The RED control parameters, max_{th}, min_{th}, max_p and w_q can be set up by the command as well.

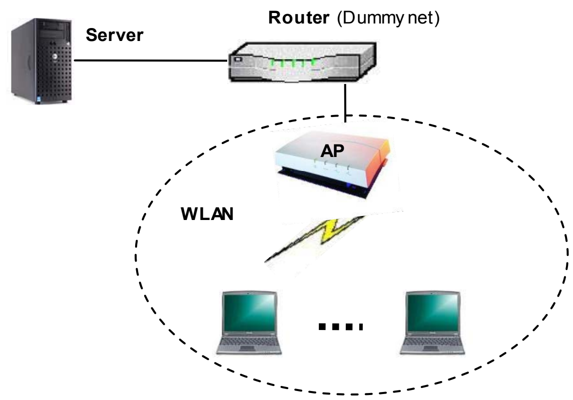

Figure 3. Experimental network.

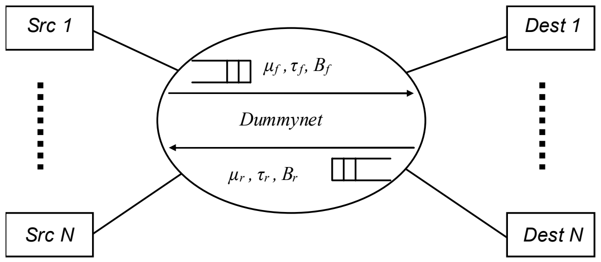

Figure 4. Dummynet topology.

C. Access Point

The 802.11b AP is an Orinoco AP 1000 from Lucent Technologies (now Agere Systems).

D. Stations

The laptop computers with Orinoco 802.11b WLAN cards serve as the data receivers. The operating system on the laptops is Red Hat Linux. No TCP protocol modification is needed for the laptops because TCP Veno only requires sender side modification.

The data rate of links connecting the server and the router, the router and the AP is 100 Mbps.

IV. Experimental Results and Discussions

The purposes of the experiments are to evaluate the benefits of TCP Veno and RED. Two major performance metrics are measured: throughput and fairness.

The throughput mentioned in this paper is the effective throughput. Fairness is defined as the extent to which each connection receives an equal share of the bandwidth when multiple connections share a link. Several fairness indices can be used to measure the fairness, such as Jain fair index, min-max ratio, and variance [10]. In our work, the standard deviation (STD) of the throughputs is used as the fairness measure.

By putting laptops at different locations, we create two different WLAN link conditions, low loss (LL) case and high loss (HL) case, to evaluate the performance. In the low loss case, the measured TCP segment (MAC frame) loss ratio is about 10^{-4}. In the high loss case, the ratio is 10^{-2}.

Two types of pipes and propagation delays are set up exclusively at the router. In the thin pipe, $\mu_f = \mu_r = 1.6$ Mbps, $\tau_f = \tau_r = 60$ ms, and the buffer size $B = 12$. In the fat pipe, $\mu_f = \mu_r = 16$ Mbps, $\tau_f = \tau_r = 30$ ms, and the buffer size $B = 60$. With the buffer settings, both the fat and the thin pipes have the same normalized buffer size[1], 0.73. No random loss is induced at the router. In the case of the 1.6 Mbps thin pipe, the bottleneck is the router or core network. In the 16 Mbps fat pipe, the bottleneck is the wireless access network, i.e. the AP.

The maximum TCP segment size is 1460 bytes. The β value of TCP Veno in the server is set to 3 in all experiments.

A. Experiment 1

In the first set of experiments, we assume all the TCP connections are homogeneous. The servers either all run TCP Veno or TCP Reno. The RED in the router is disabled and the drop-tail mechanism is used. Four TCP connections are established from the server to the clients.

The comparisons of throughputs are given in TABLE I. When the WLAN link quality is good, i.e. low loss rate, the throughput of TCP Veno is close to that of TCP Reno. When the wireless link quality is poor, TCP Veno provides higher throughputs than TCP Reno in both the thin and fat pipes. This is as expected, since Veno will provide improvement only when the transmission errors are not negligible. In the high loss cases, TCP Veno provides up to 30% more throughput than TCP Reno.

Note that even in the case of the fat pipe, the TCP throughput is far below 11 Mbps. This is because of WLAN overhead, link error, station contention, and confliction of TCP data and TCP ack segments.

B. Experiment 2

The second set of experiments is to evaluate the TCP performance when the co-existing connections are a mix of Reno and Veno, running over a mix of HL and LL wireless links.

In each experiment as listed below, four co-existing TCP connections are established. Each case corresponds to a different combination of TCP mechanisms and link qualities:

- Case 1: two HL Reno and two LL Reno connections
- Case 2: two HL Veno and two LL Veno connections
- Case 3: two LL Reno and two LL Veno connections
- Case 4: two HL Reno and two HL Veno connections

The experimental results for the fat pipe are shown in Fig. 5 and TABLE II. The results for the thin pipe are similar.

The results show that TCP Veno achieves higher throughput in the high loss case. It is also observed in the cases of mixed link conditions (Case 1 and Case 2), TCP Veno

[1] The normalized buffer size is defined to be the ratio of the buffer size to bandwidth-delay product.

TABLE I. THROUGHPUT COMPARISON – HOMOGENEOUS

	Thin pipe (router bottleneck)		Fat pipe (AP bottleneck)	
	LL	HL	LL	HL
Pure Reno	1408Kbps	1083Kbps	2768Kbps	1995Kbps
Pure Veno	1441Kbps	1383Kbps	2775Kbps	2323Kbps

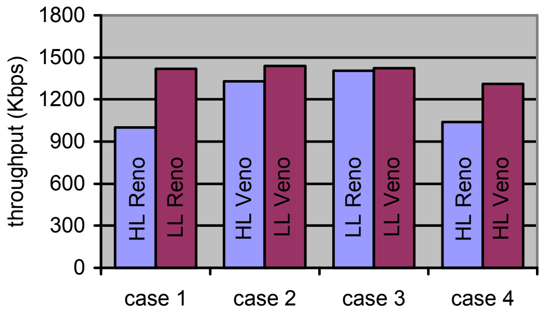

Figure 5. Throughput comparison – heterogeneous.

TABLE II. FAIRNESS COMPARISON – RENO VS VENO

	Case 1 (Mixed Reno)	Case 2 (Mixed Veno)
STD of Throughput(Kbps)	112	66

achieves higher fairness among the connections sharing one pipe than TCP Reno.

Another important observation from Case 3 and Case 4 is that TCP Veno does not grab bandwidth from TCP Reno when they co-exist. Comparing the results of Experiment 2 with Experiment 1, we find that the throughput of TCP Reno does not decrease. TCP Veno connections achieve higher throughput mainly by utilizing the available bandwidth neglected by TCP Reno connections. This means that while providing higher throughput, TCP Veno is friendly to the existing TCP Reno version.

C. Experiment 3

In this set of experiments, the RED algorithm in the router is enabled and the benefits of RED are examined for both TCP Veno and TCP Reno.

Fig. 6 and 7 show the results. In each experiment, six homogenous connections share a thin pipe, where the bandwidth is 1.6 Mbps and the buffer size is 12. The parameters for RED control are, min_{th}=5, max_{th}=10, max_p=0.1, and w_q=0.005. The experiments on the fat pipe are conducted with similar results obtained.

The experimental results show that the RED mechanism does not enhance the throughput. No obvious difference in throughput is observed. Other RED parameters (e.g. min_{th}, max_{th}) have been tried as well, with similar results obtained.

However, it is observed that the throughput variance is lowered when RED is used. This means RED helps to improve fairness among different TCP connections. Our results obtained

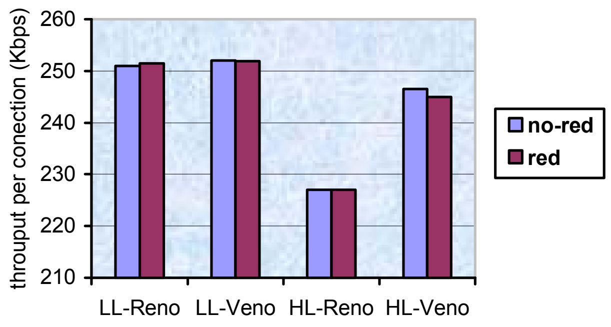

Figure 6. Average throughput comparison – RED vs no RED

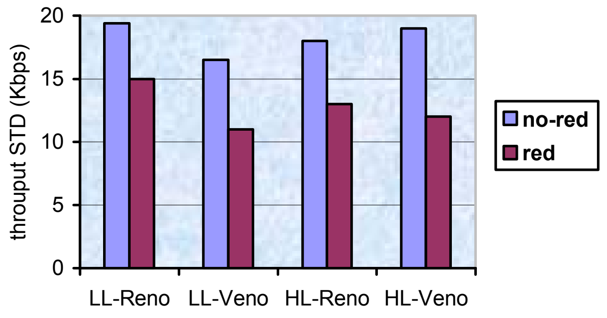

Figure 7. Throughput deviation comparison – RED vs no RED

on WLAN are consistent with the statements in [13][14], that is RED reduces the dispersion of TCP window sizes, and therefore improves the fairness. The balance of all connections is better maintained by RED.

The reason of no obvious throughput difference is as follows. Although RED may help to prevent global synchronization among the TCP connections that tends to degrade throughputs, on the other hand, RED may drop packets at the router even when it is not congested. When that happens, the TCP senders have to re-send the unnecessarily dropped packets, which leads to excessive traffic in the networks and smaller end-to-end throughput. The positive impact of synchronization prevention and the negative impact of unnecessary packet loss results in roughly the same throughput as when RED is not used.

The results also show that RED does not perform differently on TCP Reno and Veno traffic. The same impact is observed. We therefore conclude that RED does not conflict with TCP Veno.

V. CONCLUSIONS

In this paper, two variants of TCP, TCP Reno and Veno, are investigated and compared. The impact of RED on them is also examined.

Experimental results on an 802.11b WLAN show that TCP Veno can significantly improve the throughput of TCP connections over WLAN links when the transmission error rates are high. Up to 30% throughput improvement is observed in our experiments. In addition, fairness is also improved with TCP Veno in the case of mixed link conditions.

When Reno and Veno connections co-exist, it is observed that Veno connections have higher throughput. However, the higher throughput does not come at the expense of the Reno connections, since replacing the Veno connections with Reno connections does not improve the performance of the existing Reno connections. This shows that Veno can effectively make use of the available bandwidth ignored by TCP Reno.

Our experiments on RED indicate that RED does not improve the throughput in either Reno or Veno. However, RED does help to improve fairness. Although both TCP Veno and RED monitor the buffer occupancy at routers and then make use of that information to effect congestion control, our results indicate that the operations of TCP Veno and RED are more or less independent in that TCP Veno throughput is not affected much by RED. We therefore conclude that TCP Veno is compatible with RED.

In our experiments, no UDP traffic is introduced and the number of connections is relatively small. Future work will consider the interactions among a mix of a large number TCP Reno/Veno connections and UDP traffic over WLAN.

REFERENCES

[1] Neeli Prasad, and Anand Prasad, WLAN Systems and Wireless IP for Next Generation Communications, Artech House, 2002.

[2] Matthew S. Gast, 802.11 Wireless Networks: The Definitive Guides, O'Reilly, 2002.

[3] W. Richard Stevens, "TCP/IP illustrated," Addison-Wesley, 1994.

[4] Cheng Peng Fu, and Soung Chang Liew, "TCP Veno: TCP enhancement for wireless access networks," IEEE Journal of Selected Areas in Communications, vol.21, no.2, February 2003.

[5] Cheng Peng Fu, and Soung Chang Liew, "A remedy for performance degradation of TCP Vegas in asymmetric networks," IEEE Comm. Letters, January 2003.

[6] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," IEEE/ACM Transactions on Networking, vol.5, no.6, December 1997.

[7] Claudio Casetti, Mario Gerla, Saverio Mascolo, M.Y. Sanadidi, and Ren Wang, "TCP Westwood: end-to-end congestion control for wired/wireless networks," Wireless Networks, vol.8, no.5, 2002.

[8] Sally Floyd, and Van Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol.1, no.4, August 1993.

[9] RFC 2309, "Recommendations on queue management and congestion avoidance in the Internet," April 1998.

[10] Raj Jain, "Throughput fairness index: an explanation," ATM Forum contribution 99-0045, February 1999.

[11] Martin May, Jean Bolot, Christophe Diot, and Bryan Lyles, "Reasons not to deploy RED," IWQoS'99, June 1999.

[12] Mikkel Christiansen, Keven Jeffay, David Ott, F. Donelson Smith, "Tuning RED for web traffic," ACM SIGCOMM 2000, August 2000.

[13] Thomas Bonald, Martin May, and Jean-Chrysostome Bolot, "Analytic evaluation of RED performance," IEEE INFOCOM 2000, March 2000.

[14] Go Hasegawa, and Masayuki Murata, "Analysis of dynamic behaviors of many TCP connections sharing Tail-Drop/RED routers," IEEE GLOBECOM 2001, November 2001.