End-to-End Bandwidth Estimation in TCP to Improve Wireless Link Utilization

Invited paper

S. Mascolo (1), A. Grieco (2), G. Pau (3), M. Gerla (3), C. Casetti (4)

(1) *Dipartimento di Elettrotecnica ed Elettronica,* Politecnico di Bari, Via Orabona 4, 70125
Italy, Ph.: + 39 080 5963621, Fax: + 39 0805963410, e-mail: mascolo@poliba.it
(2) *Dipartimento d'Ingegneria dell'Innovazione*, Università di Lecce, Italy
(3) *Computer Science Department,* University of California, Los Angeles, USA
(4) *Dipartimento di Elettronica,* Politecnico di Torino, Italy

ABSTRACT

TCP Westwood (TCPW) is a sender-side only modification of TCP Reno congestion control that exploits an end-to-end bandwidth estimation mechanism to set the values of Slow-Start-Threshold and Congestion-Window after a congestion episode, that is after three duplicate acknowledgments or a timeout. TCP Westwood represents an innovative variant of the AIMD mechanism that we call Additive Increase Adaptive Decrease (AIAD). The aim of this paper is to investigate TCP Westwood performances in the most common wireless scenarios. We will focus on utilization of wireless links by comparing Westwood vs Reno throughput. Moreover we study the friendliness between TCP Westwood and TCP Reno in wireless scenarios. Simulations and measurements show that Westwood utilizes wireless links much better than Reno.

We choose three common scenarios including respectively a mobile client, a mobile server and a GEO satellite link. Simulation results show that Westwood obtains a remarkable throughput improvement up to 578% with respect to Reno over wireless links. Moreover, TCPW does not degrade TCP Reno performances. Finally, measurements carried out across a NASA high-sped network test-bed show an improvement of TCPW throughput up to 185% with respect to Reno.

1. INTRODUCTION

Packet switching networks require sophisticated mechanisms of flow and congestion control in order to share resources and avoid congestion. Congestion control functions were introduced into the TCP in 1988 and have been of crucial importance in preventing congestion collapse [1].

The congestion control algorithm described in [1], which eventually led to the Tahoe version of the TCP congestion control algorithm, includes two phases: slow-start and congestion avoidance. Enhanced recovery from sporadic errors is provided by Fast Retransmission and Fast Recovery mechanisms that form what is known as the TCP Reno congestion control algorithm [2].

While end-to-end TCP congestion control can insure that network capacity is not exceeded, it cannot insure fair sharing of that capacity [1]. Furthermore, TCP Reno is not well suited for wireless lossy links since sporadic losses due to radio channel problems are often misinterpreted as a symptom of congestion by current TCP schemes and thus lead to an unnecessary window and transmission rate reduction. Thus, TCP Reno requires supplementary link layer protocols such as reliable link-layer or split-connections approach to efficiently operate over wireless links [15].

The key idea of TCP Westwood (TCPW) is to exploit additional information available from the flow of TCP acknowledgment packets [3]. A TCPW source performs an end-to-end estimate of the bandwidth available along a TCP connection by measuring and low-pass filtering the rate of returning ACKs. The estimate is then used to compute the congestion window and slow start threshold after a congestion episode, that is, after three duplicate acknowledgments or after a timeout. The rationale of this strategy is simple: in contrast with TCP Reno, which implements a multiplicative decrease algorithm after congestion, TCPW sets a slow start threshold and a congestion window which are aware of, and somewhat related to, the effective bandwidth used at the time congestion is experienced. We call this mechanism *adaptive decrease.* The probing phase of network capacity is left as in Reno so that it can be said that TCP Westwood implements an *Additive Increase Adaptive Decrease* (AIAD) mechanism.

The aim of this paper is to investigate TCP Westwood performances in the most common wireless scenarios. We will focus on the utilization of a wireless link by comparing Westwood throughput to Reno throughput. Moreover, we study the friendliness between TCP Westwood and TCP Reno in wireless scenarios.

The paper is organized as follows. Section 2 provides an overview of TCP Westwood, Section 3 collects simulation results including three common scenarios, which are a mobile client, a mobile server and a GEO satellite link. They show that Westwood obtains a remarkable throughput improvement up to 578% with respect to Reno over wireless links. Moreover, Westwood does not degrade Reno performance while effectively using the wireless link. Section 4 presents a set of experiments across a NASA High-Speed network test-bed with both wireless and wired links. Measurements show a throughput improvement up to 185 %. Finally, Section 5 draws the conclusions.

2. TCP WESTWOOD OVERVIEW

A detailed description of Westwood TCP is reported in [3]. In this section, we briefly summarize its features:

a) End-to-end, sender-side estimate of the bandwidth *B* available to a TCP connection and seen at the receiver, obtained by measuring and low-pass filtering the rate of returning ACKs.

b) When 3 DUPACKs are received:
ssthresh = (*B** RTTmin) / seg_size;
cwnd = ssthresh;

c) When a coarse timeout expires:
ssthresh = (*B** RTTmin) / seg_size;
cwnd = 1;

d) When ACKs are successfully received, TCPW increases *cwnd* according to Reno's congestion control algorithm.

As has been noted in [1], [2], and [27], the stability of the Internet does not require that flows reduce their sending rate by half in response to a single congestion indication. In particular, the prevention of congestion collapse simply requires that flows use some form of end-to-end congestion control to avoid a high sending rate in the presence of high packet drop rate. In the case of TCPW the sending rate is reduced by taking into account a measurement of the rate actually achieved by the connection (i.e., the bandwidth made available to the connection) at the time congestion is experienced. Therefore, in the case of sudden increase in bottleneck load, this reduction can be even more drastic than a reduction by half and it can be less drastic in other cases. This feature clearly improves network stability and utilization in comparison with the “blind” window halving performed by Reno.

3. PERFORMANCE EVALUATION IN WIRELESS SCENARIOS

In this Section, we evaluate the impact of TCP Westwood in the most common wireless scenarios. We will consider three cases: (1) mobile client connected through a last hop wireless link to the Internet (Figure 1); (2) mobile server connected through a last hop wireless link to the Internet (Figure 1); (3) GEO satellite bottleneck link shared by TCP connections (Figure 2).

We consider for each scenario both an independent error model and a burst error model. Errors are assumed to occur on both link directions.

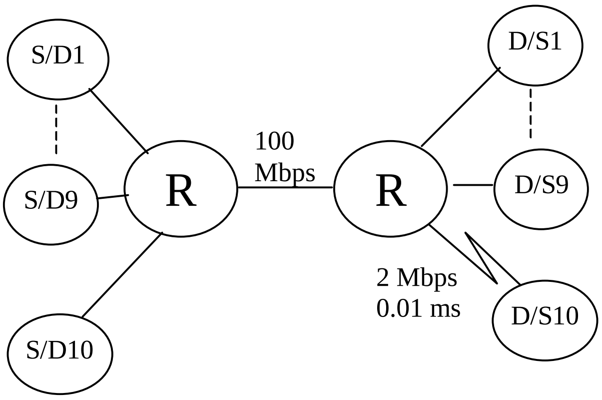


Figure 1. Mobile client or mobile server

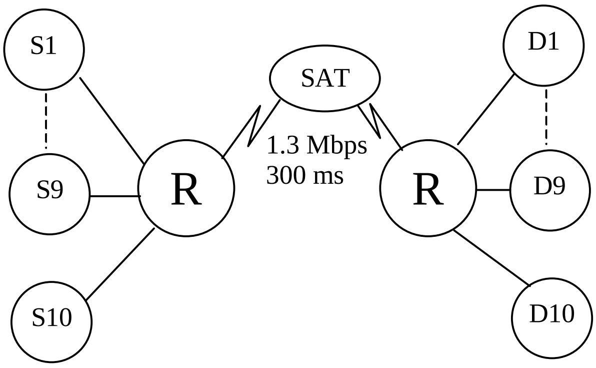


Figure 2. GEO satellite scenario.

3.1 Mobile Client

In order to test the impact of TCP Westwood on the performance of a mobile client, we test two scenarios: (a) a single-connection going through a wired portion including a 100 Mbps link between a source node and a base station with a propagation time of 62ms. The wireless portion is a very short 2Mbps wireless link with a propagation time of 0.01ms (see Figure 1). (b) a single bottleneck topology with 9 wired Reno connections sharing a 100Mbps bottleneck with a wireless connection with the same characteristics as the ones described above, (see Figure 1); RTTs of wired connections range from a minimum of 25ms to a maximum of 250 ms. Errors occur in both directions of the wireless link.

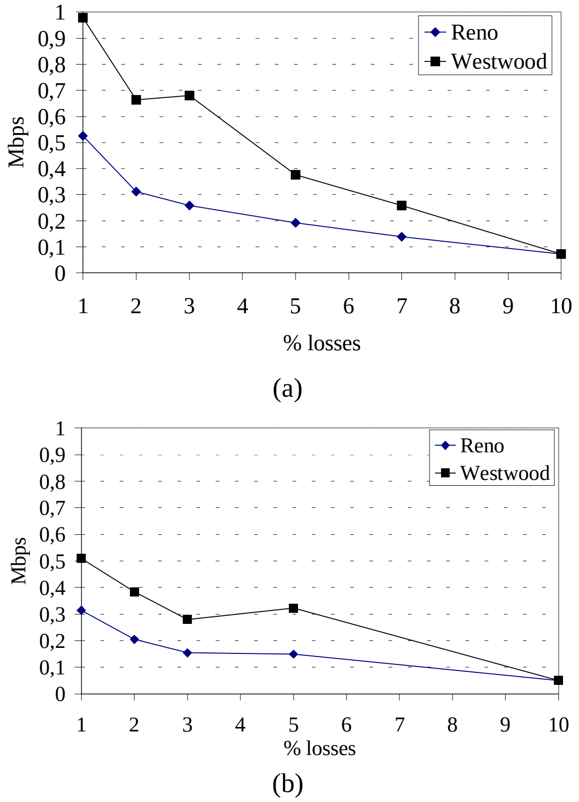
3.1.1 Independent Error Model

We assume independent (Bernoulli) errors ranging from 1% to 10% packet loss probability. The error model assumed here is equivalent to the “exponential error” model in which the time between successive errors is exponentially distributed [15]. Figures 3 (a) and (b) show the throughput of the wireless connection as a function of the loss rate in the case of single connection and, in the case of multiple connections, respectively. Two curves are shown that refer to TCP Reno and TCP Westwood. Figure 3 (a) shows that TCPW improves the throughput up to 163% with respect to TCP Reno. Figure 3 (b) shows an improvement of up to 116% of TCPW over TCP Reno.

To give a further insight, Figures 4 (a) and (b) show congestion window and slow start threshold behavior when the loss rate is 2% and Westwood or Reno are used in the single-connection scenario. These figures clearly show the reason why Westwood is more efficient than Reno in the use of wireless links: namely, losses due to unreliable links, and not to congestion, keep the values of *cwnd* and *ssthresh* for Reno much lower than those of Westwood.

3.1.2 Burst Error Models

To study TCPW performance with correlated errors, we use a 2-state Markov model (see Figure 5 and [26]). In such model, burst errors occur at a high rate due to a variety of conditions mostly associated with terminal mobility. The wireless link is assumed to be in one of two states: Good or Bad. In the Good state, a Bernoulli model is assumed for packet error. Intervals between packet errors are thus exponentially distributed (memory less channel errors). In addition, a link is assumed to stay in the Good or Bad state for a time interval that is exponentially distributed. In the Bad state we assume that errors are still Bernoulli; however, the rates of errors in the Bad state are much higher. In the simulation experiments below we vary the error rate in the bad state depending on the specific link conditions we want to study. To represent fading conditions, the segment error rate is assumed to range from 0 to 30%.

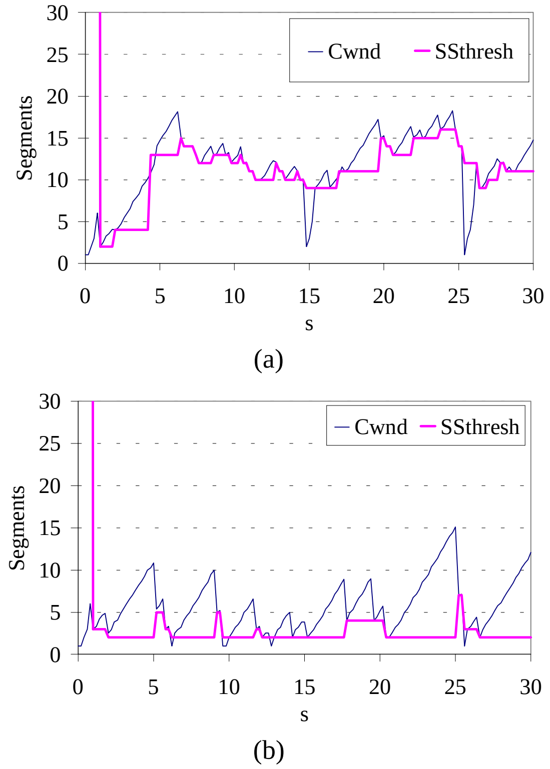


(a)

(b)

Figure 3. Average throughput under independent lossy condition. (a) Single connection. (b) Multiple connections

Let the Bad state represent fading conditions, and let the mean duration of Good and Bad states be 8*s* and 4*s*, respectively. In the Good state a 0.1% packet loss is assumed, whereas in the Bad state the loss is varied from 0 to 30%. Results in Figures 6 (a) and (b) show the throughput of the wireless connection as a function of the loss rate in the Bad state when topology (1) or (2) is assumed, respectively. Figure 6(a) shows that the improvement of TCPW with respect to Reno ranges from 66%, when the loss rate is 7%, to 578% when the loss rate is 25%. For greater loss rates TCPW and Reno tend to the same throughput. Figure 6(b) shows that the improvement of TCPW with respect to Reno ranges from 16%, when the loss rate is 20%, to 118% when the loss rate is 3%. Again, for loss rates greater than 20%, TCPW and Reno tend to the same throughput.



(a)

(b)

Figure 4. Congestion Window and Slow Start Threshold behaviors. (a) Westwood. (b) Reno

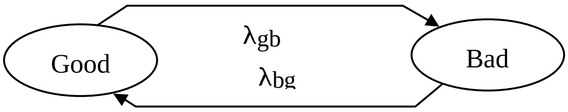


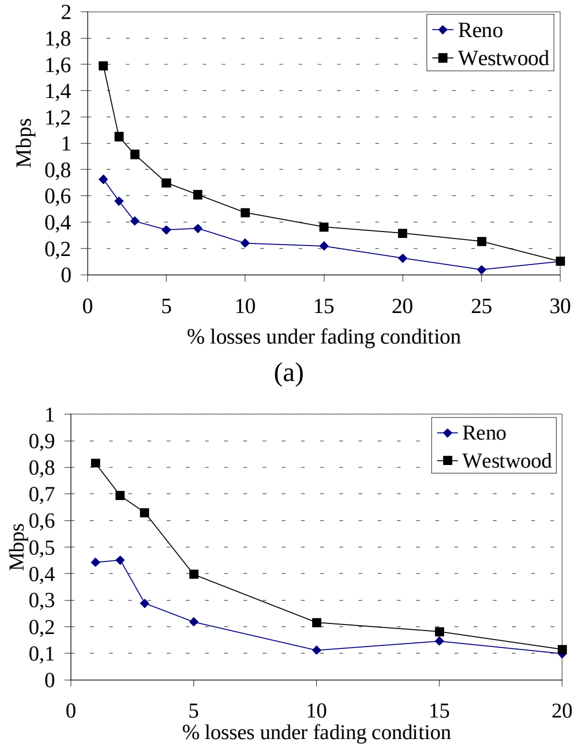
Figure 5. Two-State Markov Model for Burst Error Characterization

3.2 Mobile Server

In this Subsection we test the impact of TCP Westwood on the performance of a mobile server. We simulate a single connection and a multiple connection scenario as described in the previous Subsection. The only difference is that the mobile node is now the server. Again, both independent and correlated error models are assumed. Errors occur in both directions of the wireless link.

3.2.1 Independent Error Model

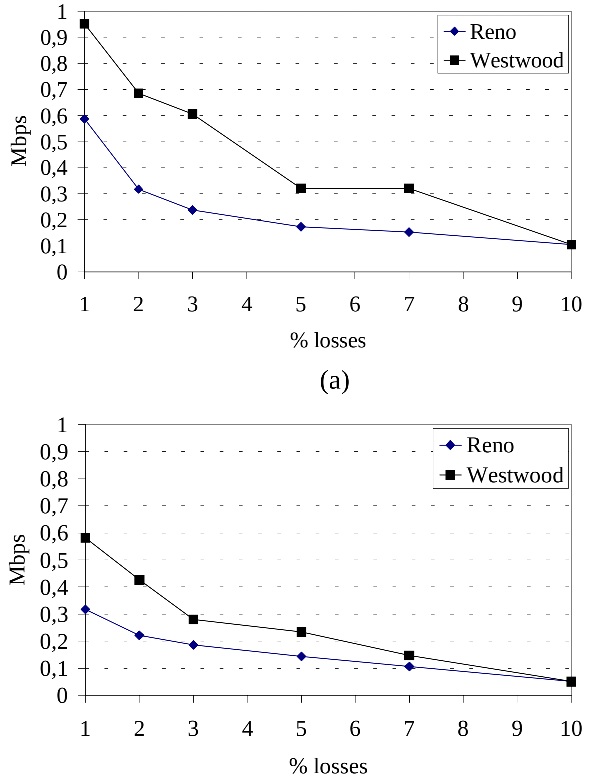
We use the same error model employed in Section 3.1.1. Figures 7 (a) and (b) show the wireless connection throughput as a function of the loss rate in the case of single and multiple connection scenarios, respectively. Two curves are shown that refer to a wireless TCP Reno connection and to a wireless TCP Westwood connection. Figure 7 (a) shows that TCPW improves the throughput up to 155% with respect to TCP Reno, whereas a 93% improvement is shown in Figure 7 (b)



(a)

(b)

Figure 6. Throughput vs. loss rate of the Bad state. (a) Single connection. (b) Multiple connections



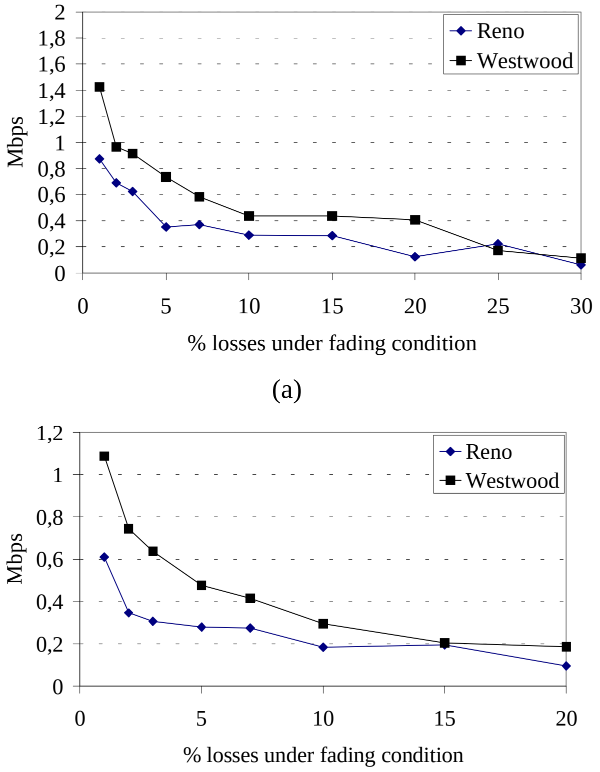
(a)

(b)

Figure 7. Average throughput under independent lossy condition. (a) Single connection. (b) Multiple connections

3.2.2 Burst Error Models

We use the same error model employed in Section 3*.1.2.* Results in Figures 8 (a) and (b) show the throughput of the wireless connection as a function of the loss rate in the Bad state in the case of single and multiple connection scenario, respectively. Figure 8 (a) shows that the improvement of TCPW over Reno ranges from 40%, when the loss rate is 2%, to 222% when the loss rate is 20%. For greater loss rate TCPW and Reno tend to the same throughput. In Figure 8(b) the improvement of TCPW over Reno ranges from 60%, when the loss rate is 10%, to 115% when the loss rate is 2%. For loss rate greater than 20% TCPW and Reno converge to the same throughput.



(a)

(b)

Figure 8. Average throughput vs. loss rate of the bad state. (a) Single connection. (b) Multiple connections

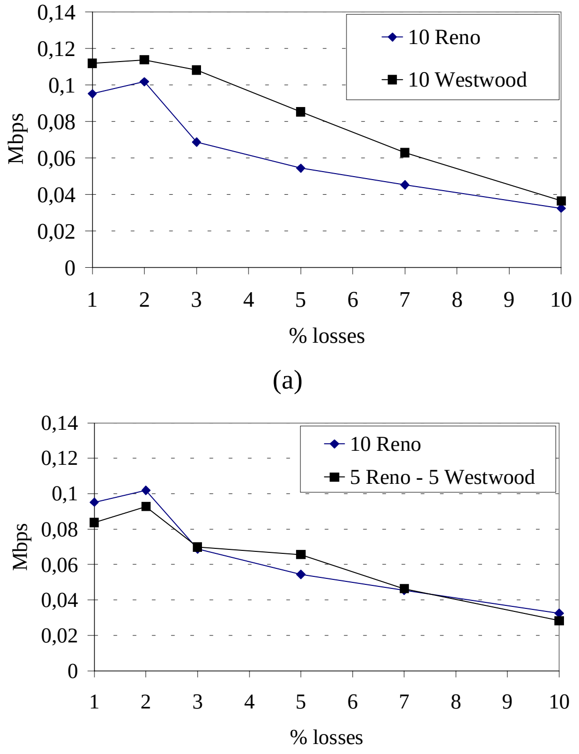
3.3 Geo Satellite scenario

In this Subsection we investigate Westwood performance over a GEO satellite link. We simulate a single bottleneck topology in which 10 TCP sources are sharing the GEO satellite link. We model the GEO satellite link with a bandwidth of 1.3 Mbps and Round Trip Time of 600ms, see Figure 2. We compare Reno and Westwood performances by measuring the mean throughput of 10 Westwood sources, 10 Reno sources and finally of 5 Westwood and 5 Reno sources sharing the GEO satellite link at the same time. Errors occur in both directions of the satellite link.

3.3.1 Independent Error Model

We use the same error model employed in Section 3*.1.1.* Figure 9(a) shows the mean throughput of the 10 TCP connections as a function of the loss rate. Two curves are shown that refer to 10 TCP Reno connections and 10 TCP Westwood connections. Figure 9(a) shows that TCPW improves the throughput up to 57% over TCP Reno.

Figure 9(b) compares the mean throughput of 10 Reno connections as a function of the loss rate (the same curve shown in the Figure 9(a)) and the mean throughput of just 5 Reno connections, sharing the bottleneck with 5 Westwood connections, whose throughput is not reported. The two curves show that Westwood does not reduce the throughput of Reno sources, i.e. replacing 5 TCP Reno connections (out of 10 TCP Reno) with 5 TCPW connections does not affect the throughput of those remaining 5 Reno connections.



(a)

(b)

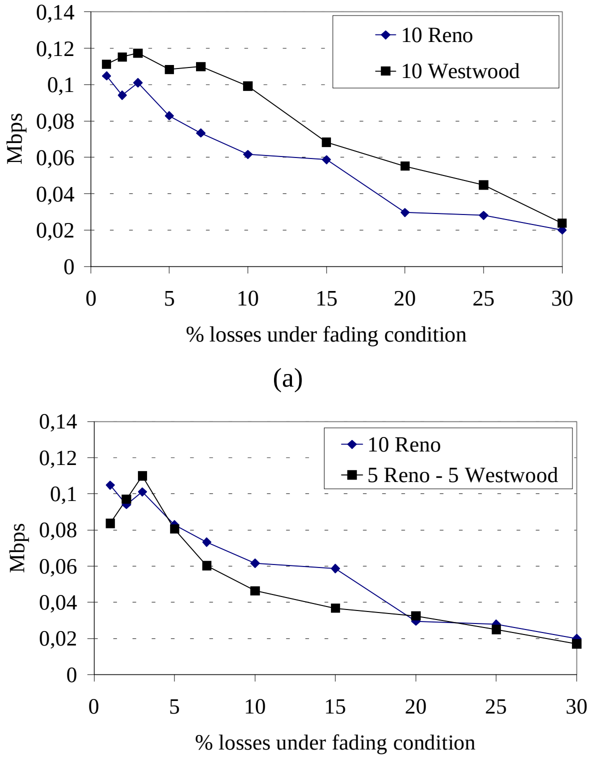
Figure 9. Average throughput under lossy condition. (a) Reno vs Westwood. (b) Friendliness evaluation

3.3.2 Burst error Model

By assuming the same error model employed in the section 3*.1.2* results similar to the ones in the previous Subsection are found. Figure 10(a) highlights a throughput improvement by TCPW of up to 87% with respect to TCP Reno. Similarly, Figure 10(b) shows that Westwood does not reduce the throughput of Reno sources.

4. INTERNET MEASUREMENTS

In this Section, we present a set of experiments across a high-speed network with both wireless and wired links. The first set of experiments was carried out using the California Research Network (CALREN), the NASA Research Network (NREN) and a dedicated high speed (45 Mbps) satellite connection also provided by NASA. A second set of Internet experiments with Brazil, Italy and Taiwan was performed using a source in the UCLA Network Research Laboratory test-bed.



(a)

(b)

Figure 10. Average throughput vs. loss rate of the bad state. (a) Reno vs Westwood. (b) Friendliness evaluation

4.1 Experiments over the NASA network

The network topology used to test TCP-Westwood across the CALREN and NREN networks is depicted in Figure 11. The path has an average round trip time of 650ms and the available bandwidth measured at different times during the day is on average 26.7 Mbps A number of tests using one single TCP Westwood connection and one single TCP Reno connection were performed. In order to emulate a fairly large file transfer both connections were kept running for 2 minutes. Each test was repeated at different times during the day. The experiments were conducted using the tool IPERF [28] developed by NLNAR. IPERF is a client-server application specifically developed to measure TCP performance across a given Internet path. In addition, the connections were traced using tcpdump [29], an application that sniffs (using the Ethernet interface promiscuous mode) and saves on disk all packets sent or received from a given network interface. The traces acquired using tcpdump were then analyzed using the tcptrace tool [34].

Table I shows the max, min and average statistics of experiments collected at different times. The loss is expressed as percentage of lost packets. The throughput is measured by dividing the total received bits by the simulation interval which amounts to two minutes.

Both Reno and Westwood connections never fill the pipe since the ideal window is 2.1 Mbytes, whereas the max *cwnd* measured in our experiments was 96360 Bytes. The reason for the small *cwnd* in spite of the use of the "large window" TCP option was the presence of frequent timeouts (caused by line errors) and the frequent resetting of the slow start threshold. It is worth noting that TCPW achieves on average twice the throughput of Reno. This is due to the more efficient resetting of *cwnd* and *ssthresh*. Moreover, TCPW throughput is practically the same over all the experiments, while Reno throughput shows large fluctuations.

	Min	**Max**	**Avg**
RTT	630 ms	960ms	644.3ms
RTO Events	0.00 %	0.48 %	0.22 %
Triple Dup Acks	0.01 %	0.37 %	0.17%
RENO Throughput **bit/s**	264664 bit/s	595488 bit/s	440050 bit/s
Westwood Throughput **bit/s**	752792 bit/s	778040 bit/s	764968 bit/s

Table 1: NASA Experiment Summary

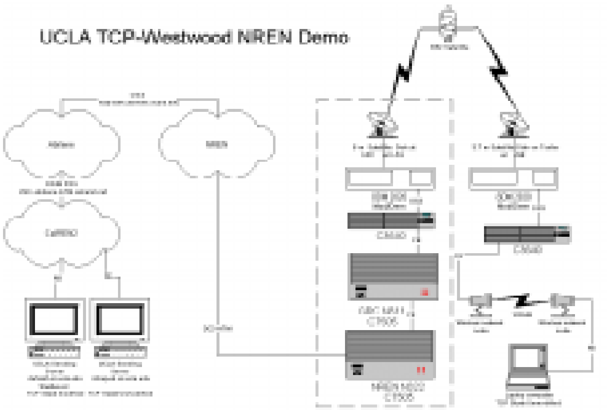


Figure 11: Experiment Scenario

4.2 Internet Measurements

To test TCP Westwood in the Internet environment, we have carried out a set of Internet experiments using the configuration depicted in figure 12. The sources (i.e., the host sending data) are at UCLA, while the destinations (i.e. the host acknowledging the data) are chosen in three different continents (Europe, South America, and Asia). The destination hosts are, of course, unaware whether the source host runs TCPW or Reno.

Tests were scheduled during normal operations, i.e., during normal working hours at the destination sites. Throughput results were obtained by averaging several single file transfers. A rather large file size was used (10 Mbytes) to capture only steady state behavior. We used a standard FTP client (ncftp-3.0.2) as testing software with additional code for obtaining detailed logging at 1-second intervals. We measured application throughput in terms of user data/second as reported by ftp. The average throughput achieved by Reno and TCPW on each intercontinental connection is shown in Table 2. Tests were repeated about 200 times throughout the day. The results show that TCPW performs marginally better than Reno on the Italy and Taiwan connections. It performs significantly better on the Brazil connection.

This result motivated further examination of the paths involved in the experiments using the well-known `traceroute` tool.

We found that Italy and Taiwan are connected using standard wired technology. In this case, link errors are expected to be minimal, thus TCPW does not introduce much improvement over Reno. On the other hand, the Brazil path has a "lossy" satellite link provided by Teleglobe. The lossy link accounts for the TCPW improved performance.

Destination RTT	Throughput (Kbit/s)	
	TCPW	Reno
Italy 170 ms	629.28	591.44
Taiwan 250 ms	1339.04	1216
Brazil 450 ms	177.28	123.2

Table 2: Internet Experiment Summary

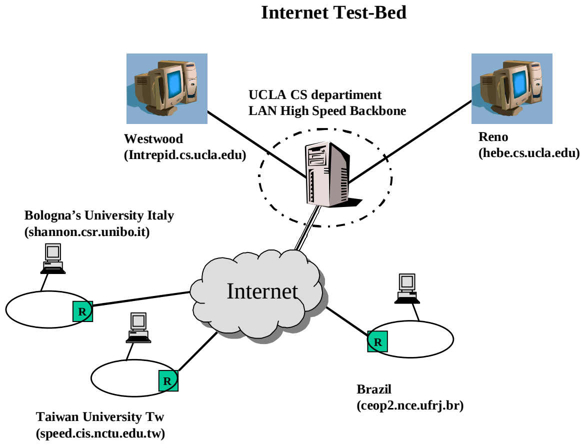


Figure 12: Internet Experiment Scenario

5. CONCLUSIONS

We have investigated TCP Westwood performances over the most common wireless scenarios. Analyses are based on simulations and measurements and show that Westwood manages to use wireless links much better than Reno. Simulation results show throughput improvement up to 578%. Moreover simulations show that TCP Westwood is friendly to Reno in wireless scenarios. Measurements in the NASA testbed show throughput improvements ranging from 31% to 185%. Measurements carried out in Internet scenarios show a throughput improvement up to 47% when the TCP connection includes a satellite link.

REFERENCES

[1] V. Jacobson, "Congestion Avoidance and Control," ACM Computer Communications Review, 18(4): 314 - 329, August 1988.

[2] V. Jacobson, "Berkeley TCP evolution from 4.3-Tahoe to 4.3 Reno," Proceedings of the 18th

Internet Engineering Task Force, University of British Colombia, Vancouver, BC, Sept. 1990

[3] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, Ren Wang, “TCP Westwood: End-to-End Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks,” ACM Mobicom 2001, July 2001, Rome, Italy.

[4] D. Clark, “The design philosophy of the DARPA Internet protocols,” Sigcomm’88 in ACM Computer Communication Review, vol. 18, no. 4, pp. 106 - 114, 1988.

[5] S. Floyd, K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” IEEE/ACM Transactions on Networking, Aug. 1999, vol.7, (no.4):458-72.

[6] K.J. Åström , B.Wittenmark, “Computer controlled systems,” Prentice Hall, Englewood Cliffs, N. J., 1997.

[7] ns-2 network simulator (ver 2). LBL, URL: http://www.isi.edu/nsnam/ns

[8] R. Jain, “The art of computer systems performance analysis,” John Wiley and Sons, 1991.

[9] W. Stevens, “TCP/IP illustrated,”, Addison Wesley, Reading, MA, 1994.

[10] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, “TCP Selective Acknowledgement Options,” RFC 2018, April 1996.

[11] L.S. Brakmo, S.W: O’Malley, L.L. Peterson, "TCP Vegas: End-to-end congestion avoidance on a global Internet," IEEE Journal on Selected Areas in Communications (JSAC), vol. 13, no.8, pp. 1465-1480, 1995.

[12] J., C.Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP", ACM Sigcomm'96, Antibes, France, August 1996, pp. 270-280.

[13] E. L. Hahne, C. R. Kalmanek, S. P. Morgan, “Dynamic window flow control on a high-speed wide-area data network,” Computer Networks and ISDN Systems, 26 (1993), 29-41, North-Holland.

[14] R. Morris “TCP behavior with Many Flows,” IEEE International Conference on Network Protocols, October 1997, Atlanta, Georgia, pages 205-211.

[15] H. Balakrishnan, V.N., Padmanabhan, S. Seshan, R.H.Katz, “A comparison of mechanisms for improving TCP performance over wireless links,” IEEE/ACM Transactions on Networking, Dec. 1997, vol.5, no.6, 756-69.

[16] Keshav, S., “A control-theoretic approach to flow control,” SIGCOMM '91 Conference”, Communications, Architectures and Protocols, Zurich, Switzerland, 3-6 Sept. 1991, Computer Communication Review, Sept. 1991, vol.21, (no.4):3-15.

[17] M. Allman, V. Paxsons, “On Estimating End-to-End Network Path Properties”, ACM Sigcomm 1999, Cambridge, MA, USA, August 1999.

[18] K. Lai, M. Baker, “Measuring Link Bandwidths Using a Deterministic Model of Packet Delay,” ACM Sigcomm 2000 , Stockholm, Sweden, August 2000.

[19] S. Q. Li, C. Hwang, “Link Capacity Allocation and Network Control by Filtered Input Rate in High speed Networks,” IEEE/ACM Trans. Networking, vol. 3, no. 1, Feb. 1995, pp. 10-25.

[20] F. Kelly, “Mathematical modeling of the Internet,” In "Mathematics Unlimited - 2001 and Beyond" (*Editors B. Engquist and W. Schmid*). Springer-Verlag, Berlin, 2001. 685-702.

[21] S. Mascolo, “Congestion control in high-speed communication networks," Automatica, Special Issue on Control Methods for Communication Networks, vol. 35, n. 12, December 1999.

[22] S. Floyd, V. Jacobson, “Random Early Detection gateways for congestion avoidance,” IEEE/ACM Transactions on Networking, 1(4), August 1997.

[23] D. D. Clark, W. Fang, “Explicit allocation of best effort packet delivery service,” ACM/IEEE Transactions on Networking, August 1998.

[24] M. May, J. Bolot, C. Diot, B. Lyles, “Reasons not to deploy RED,” Seventh International Workshop on Quality of Service, IWQoS'99, London, England, June 1999.

[25] J. Padhye, V. Firoiu, Don Towsley, J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation,” ACM Sigcomm 1998, Vancouver, BC, Canada, September 1998

[26] A.A. Abouzeid, , S. Roy, M. Azizoglu, “Stochastic Modeling of TCP over Lossy link,” IEEE Infocom 2000, Tel Aviv, Israel, March 2000.

[27] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, “Equation-Based Congestion Control for Unicast Applications,” ACM Sigcomm 2000 , Stockholm, Sweden, August 2000.

[28] A. Tirumala J. Ferguson, “Iperf performance test,” NLANRhttp://dast.nlanr.net/Projects/Iperf/index.html

[29] V. Jacobson and S. Mc Canne, “The BSD packet Filter: A New Architecture for User-Level Packet Capture,” USENIX winter Conference ’93, San Diego, CA, USA

[30] S. Ostermann, “TCPTRACE software tool,” http://www.tcptrace.org/