Effect of Local Retransmission at Wireless Access Points on the Round Trip Time Estimation of TCP*

KARUNAHARAN RATNAM
Dept. of Electrical and Computer Engineering
Northeastern University
Boston MA 02115
karu@ece.neu.edu

IBRAHIM MATTA
College of Computer Science
Northeastern University
Boston, MA 02115
matta@ccs.neu.edu

Abstract

The Transmission Control Protocol (TCP) has been mainly designed assuming a relatively reliable wireline network. It is known to perform poorly in the presence of wireless links because of its basic assumption that any *loss of a data segment is due to congestion and consequently it invokes congestion control measures. However, on wireless access links, a large number of segment losses will occur more often because of wireless link errors or host mobility. For this reason, many proposals have recently appeared to improve TCP performance in such environment. They usually rely on the wireless access points (base stations) to locally retransmit the data in order to hide wireless losses from TCP. In this paper, we use extensive simulations to evaluate TCP performance in the presence of congestion and wireless losses when the base station employs one of two proposals, namely Snoop [2] and WTCP [8]. Our results show that WTCP significantly improves the throughput of TCP connections due to its unique feature of hiding the time spent by the base station to locally recover from wireless link errors so that TCP's round trip time estimation at the source is not affected. This proved to be critical since otherwise the ability of the source to effectively detect congestion in the fixed wireline network is hindered.*

1 Introduction

The development of portable communication devices in recent years has made it possible for a mobile user to access the services provided by the Internet (*e.g.* e-mail, WWW) through wireless radio technology. Such wireless access is provided through *base stations* connected to the (wired) Internet. The design of such system requires special attention to its wireless and mobile nature. However, transport protocols not tailored for such system are often used, specifically the Transmission Control Protocol (TCP) [4]. TCP was mainly developed for fixed (wired) networks with relatively reliable links and low bit error rates (BER). However, a wireless channel is generally characterized by its high BER. When used in a wireless environment, TCP *wrongly* assumes data segment loss due to wireless link errors to be a sign of network congestion and invokes congestion control mechanisms that curb the flow of segments on that connection.[1] However, for better throughput, a segment loss due to wireless link errors must be detected and retransmitted as quickly as possible.

There are many proposals to improve TCP performance over networks with wireless links. Invariably, most of them, in one form or another, rely on the base station to improve performance. We discuss two proposals, *Snoop* [2] and *WTCP* [8], that improve TCP performance while maintaining end-to-end TCP semantics[2], and require no modification to the TCP code that runs on the end-systems (*i.e.* hosts). In both of these proposals, the base station buffers TCP segments destined for the mobile host and retransmits a segment if the segment is deemed to be lost. That is, the base station hides the wireless error losses from the TCP source.[3] But, local retransmissions by a base sta-

*This work was supported in part by NSF CAREER grant NCR-9701988 and NEU grant RSDF-377090.

[1] The same problem occurs with TCP when a segment is lost due to mobility handoff from one base station to another. We do not however consider mobility in this paper.

[2] By end-to-end semantics we mean that an acknowledgment to a data segment comes only from the destination host and not from any other intermediate system.

[3] We point out that congestion due to contention on the wireless link is resolved by a lower layer (MAC) protocol and losses detected at the higher/transport layer of the base station are assumed to be due to wireless errors.

tion can affect the round trip time (RTT) estimation at the TCP source, which in turn can impair its ability to detect congestion losses (due to buffer overflow) in the wired network. In this paper, we investigate the effect of hiding the time spent by a TCP segment in the base station buffer, and compare the performance with the traditional approach in which local recovery time is not hidden.

Next we briefly describe Snoop and WTCP schemes. We assume a TCP connection between a source host in the fixed network and a mobile host (sink). See Figure 1. The mobile host is connected to a base station by a wireless link, and the base station is connected to the fixed network by wired links.

Snoop

In this scheme [2], the base station sniffs the link interface for any TCP segments destined for the mobile host, and buffers them if buffer space is available. The base station also sniffs into the acknowledgments from the mobile host. If the base station sees a duplicate acknowledgment[4], it detects a segment loss and locally retransmits the lost segment if it is buffered and starts a timer. If the retransmitted segment is not acknowledged within twice the RTT of the wireless link, the base station times out and again retransmits the segment.

WTCP

In WTCP [8], the TCP connection from the source is terminated at the base station, and another TCP connection is created from the base station to the mobile host. However, the base station acknowledges a TCP segment to the source only after that segment is acknowledged by the mobile host. This way, TCP's end-to-end semantics are preserved.

The base station retransmits TCP segments to the mobile host based on timeout *as well as* duplicate acknowledgment. In case of timeout, the transmission window for the wireless connection is reduced to just *one* segment assuming a typical burst loss on the wireless link is going to follow. After receiving an acknowledgment, WTCP opens the wireless transmission window completely. In the case of duplicate acknowledgment, WTCP does *not* alter the wireless transmission window assuming that the reception of the duplicate acknowledgment is an indication that the wireless link is in good state, and the base station immediately retransmits the lost segment. The base station continues to transmit the remaining segments that are within the wireless transmission window if they have not already been transmitted. However, until the mobile host receives the lost segment, the reception of each out-of-order segment will generate a duplicate acknowledgment. The number of these additional duplicate acknowledgments can be determined by the base station, which ceases to retransmit during their reception.

Furthermore, WTCP implements a mechanism to hide the time spent by a TCP segment in the base station buffer. This mechanism is explained in Section 3.1.

Organization of the paper

The rest of the paper is organized as follows. Section 2 describes our simulation model and parameters. Our results are discussed in Section 3. We elaborate in Section 3.1 on WTCP's feature of hiding the local recovery time at the base station and its effect on TCP performance. Section 4 concludes the paper.

2 Simulation Model

In our evaluations, we used the network simulator *ns* v.1.2a3 [7], a public domain software from Lawrence Berkeley National Lab. We extended *ns* to incorporate wireless link characteristics and to simulate WTCP and Snoop. We also simulated I-TCP [1], a scheme that splits the TCP connection into two as in WTCP *but* does not preserve TCP's end-to-end semantics.[5] No changes are made to the fixed network hosts or the mobile hosts. However, the base station node is modified to implement the protocols we consider. For all protocols, we change the clock resolution of the base station to 10 ms while maintaining the clock resolution of the fixed host at 100 ms. The reason is that in most multi-hop networks, the round trip time (RTT) is typically in the order of few hundred milliseconds. This is why many TCP implementations use a coarse clock granularity. For example, 4.4BSD-Lite uses a 500ms clock [10]. On the other hand, the base station–mobile host (wireless) link is single-hop, and its RTT is typically in the order of tens of milliseconds. Hence, in order to accurately estimate the wireless link RTT and to trigger timeouts in the millisecond range, we assume that the protocol at the base station uses a 10 ms clock granularity. Furthermore, WTCP does not use

[4] Because TCP uses cumulative acknowledgments (i.e. an acknowledgment indicates that all earlier data has been received correctly), a mobile host would generate a duplicate acknowledgment if it receives out-of-order data.

[5] We only consider I-TCP to provide an upper bound on throughput since in general the source receives acknowledgments faster as they are received from the base station and not from the mobile host.

the smooth averaging used in regular TCP to estimate the RTT of the wireless link since RTT variation is relatively small.

To handle timeout based retransmissions over the wireless link, WTCP modifies the backoff algorithm of TCP as follows. Initially, the timeout value at the base station is set to 1.5 times the wireless link RTT. However, after the first timeout, WTCP backs off the next timeout by thrice the wireless link RTT. But, additional timeouts do *not* increase the timeout value any further. After receiving an acknowledgment, the timeout value is reset to 1.5 times the wireless link RTT. The value of 1.5 is chosen so that any minor variation in the RTT do not trigger unnecessary timeouts. The selection of the backoff value depends on the nature of the wireless link. The first retransmission occurs at $1.5 \times$ RTT. If the exact duration of the burst error (bad) state of the wireless link is known (see below), the second retransmission can be made to occur immediately after the link returns to a good state. However, since the exact burst error duration is not known, the second retransmission can be made to occur after the mean burst duration by selecting a suitable backoff period. For example, if the mean burst duration is around $4 \times$ RTT, then the backoff duration for the second retransmission can be set to $3 \times$ RTT so that the total time for the second retransmission to take place is ($1.5 \times$ RTT + $3 \times$ RTT =) $4.5 \times$ RTT, which covers the mean bad state duration.

We simulated the Snoop protocol based on [2]. Snoop uses negative acknowledgments at the mobile host to improve performance. Here the sink keeps track of the segments that are lost in a transmission window, and sends negative acknowledgements to those lost segments. However, because this requires modification to the TCP code at the mobile host, we did not implement negative acknowledgments.

Wireless channel error model

Unlike the wired link, wireless links are characterized by high bit error rates. The errors are generally bursty in nature with varying burst duration. We model the wireless link using a two-state Markov channel [3], where the channel is in either good state or bad state. In the good state the bit error rate (BER) of the channel is low, and in the bad state BER is high. We simulated the wireless link with bit error rates of 10^{-6} and 10^{-2} in good and bad states, respectively. The time spent in each state is modeled by an exponential distribution with mean good state duration fixed at 1 second. The mean bad state duration is varied from 10 ms to 100 ms.

Parameter	Value
Maximum window size	10 Kbytes
Buffer size at base station	20 Kbytes
Segment size (including header)	1 Kbyte
BER in good state	10^{-2}
BER in bad state	10^{-6}
Mean good period	1 sec

Table 1. Simulation parameters.

Simulated network

We have simulated the network shown in Figure 1. The wired link bandwidth is 10 Mbps with 100 ms latency. The wireless side has an 800 Kbps link with 10 ms latency. Table 1 shows the parameters used in the simulation. We note that the maximum window size of the TCP connection on both sides of the base station is chosen to be smaller than its maximum buffer size, which would avoid buffer overflow at the base station for both Snoop and WTCP. This is because the base station buffer cannot become full as the transport connection is *end-to-end* and so the number of unacknowledged segments sent out by the source is bounded by its transmission window. However, in the case of I-TCP, buffer overflow can easily happen due to the split nature of the connection. TCP handles the receiver buffer overflow situation by using receiver window size advertisements. Thus for the I-TCP simulation, we modified the *ns* code at the base station to advertise available buffer size to the source as the window size in every acknowledgement. When there is no buffer space available, the base station advertises a window size of zero, and the source refrains from transmitting any more data until the window size is increased. The base station will increase the window size only if it can accept at least one full TCP segment [9].

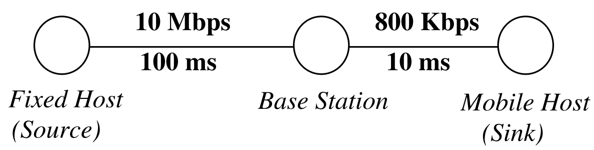

Figure 1. Simulated network.

3 Results and Discussion

We measured the throughput of I-TCP, Snoop, WTCP and TCP-tahoe[6], and the results are shown in Figure 2. Throughput is measured as the average number of bits successfully transmitted to the mobile host in a second. Each simulation is run for 200 seconds and the results are obtained with a confidence interval of 95%. We assume no losses on the wired link. For a good quality wireless link, the throughput of I-TCP is better than other protocols. However, when the wireless link quality degrades WTCP yields better throughput mainly because of its aggressive retransmission policy over the wireless link. WTCP achieves throughput values 4-8 times higher than TCP-tahoe. However, WTCP's aggressive retransmission policy slightly degrades the wireless link utilization than other schemes [8]. However, throughput gain outweighs the small loss in link utilization.

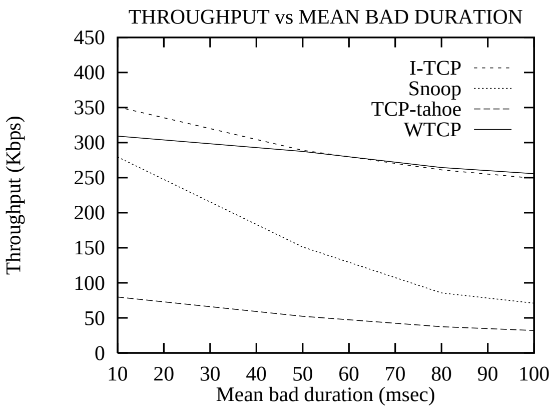

Figure 2. Throughput of different protocols.

Though Snoop achieves throughput values comparable to I-TCP and WTCP at low loss situations, the throughput of Snoop is poor when the wireless link is very bursty (in bad error state) for long durations. The reason is that Snoop triggers retransmission only after the base station receives a duplicate acknowledgment. However, when the wireless link is in bad error state, it is likely that acknowledgments be lost. In this situation, the base station fails to trigger duplicate acknowledgment based retransmission, and the throughput degrades.

[6] We do not show results for TCP-reno as it was found to be less robust than TCP-tahoe in wireless applications [6].

3.1 WTCP: On Hiding the Effect of Local Retransmission on RTT Estimation of TCP

In the older version of TCP specified in RFC 1185 [5], the (end-to-end) round trip time (RTT) is computed once per transmission window, and none when there is a timeout. Since RTT is not frequently updated, the RTT estimate may not be accurate. Inaccurate RTT estimation can cause unnecessary retransmissions by triggering timeouts too soon or can degrade the throughput by triggering timeouts too late. The newer implementation of TCP specified in RFC 1323 [4] uses the timestamp option in the TCP header to better estimate the RTT. The sender writes the current timestamp in the header, and the receiver echoes back this timestamp without modifying it. Since each acknowledgment contains the timestamp of the segment that generated it, RTT can be computed for every acknowledgment, even in the presence of timeout in a transmission window. However, when an intermediate host (base station) buffers the segments and locally tries to retransmit them, the RTT computation at the source can be affected. For example, if a segment is successfully retransmitted after one or more timeouts at the base station, the RTT value computed at the source for this segment will overshoot by the residence time of this segment at the base station buffer. If the burst error duration of the wireless link is significantly long, the residence time of the segment in the WTCP buffer may span a few source clock ticks and affect the RTT computation, especially its variance. Even after successful transmission resumes on the wireless link and acknowledgments are sent back to source, the RTT computation at the source may take a long time to settle down to the actual value. This problem exists in all the proposals that use local retransmission by the base station. To hide the wireless link errors from the source, ideally the residence time of the segment at the WTCP buffer could be added to the timestamp value in the TCP header before the acknowledgment is forwarded back to the source. This has the effect of subtracting the residence time at the base station from the RTT value computed by the source.

This approach requires the base station to know the clock granularity of the source since the timestamp field in the TCP header contains a clock tick value, not the real-time clock value. In particular, when a system is booted, the clock tick is set to 0 and is incremented by one every 500 ms in 4.4BSD-Lite. To get around the problem of not knowing the clock granularity, an *approximation* to subtracting the exact residence time at the base station is as follows: when a segment is sent to the mobile host the base station can replace the timestamp of that segment with the timestamp of the *most recent segment* received from the source.[7] When the acknowledgment comes back from the mobile host, it is forwarded to the source without any modification. When there are no source timeouts, the base station receives TCP segments from the source at least once every RTT interval. Hence the maximum error in the RTT computation using the most recent timestamp is bounded by one RTT (see Figure 3) when there are no source timeouts.

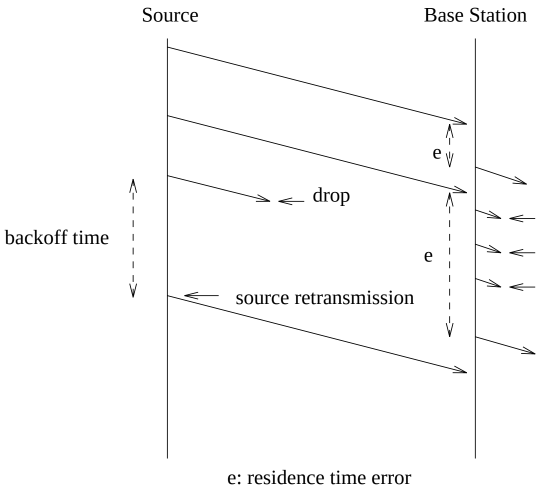

Figure 3. Diagram showing the error in RTT using the most recent timestamp.

The above approach works fine as long as the base station frequently receives TCP segments from the source. However, if the source times out due to a prolong burst error state of the wireless link or due to congestion in the wired portion of the network, the inter-arrival time between source TCP segments at the base station can be large. During this phase, the error in excluding residence time using the timestamp of the most recent segment received by the base station can be as large as the source backoff time (see Figure 3). Any acknowledgment sent to the source using this timestamp will obviously affect the RTT estimate maintained at the source by a large factor. To avoid this problem, before sending a segment over the wireless link, the base station can remove its timestamp option field if there has been any retransmission from the base station *after* receiving the most recent segment from the source. Timestamps are again used only after receiving a TCP segment from the source.

In Figure 4 we plot the source timeout values when WTCP is employed at the base station, sampled every second, as a function of time. The plotted timeout value is the initial timeout computed by the TCP source, which is close to the smooth (average) RTT between the fixed host and mobile host when the variance is small. The granularity of the source clock is 100 ms. The mean bad period of the wireless link is 100 ms, and there are no wired link losses. Figure 4 shows the effect of (a) not hiding the residence time; (b) excluding actual residence time assuming an ideal scenario where the granularity of the source clock is known to the base station; and (c) using the timestamp of the most recent segment received from the source with timestamp removal option (discussed earlier). When the residence time at the base station is not excluded, the timeout value reaches as high as 2.4 seconds, a five fold jump from the timeout value when the residence time is excluded.

In the latter case, the timeout value varies between 500–700 ms. However, when the exact value of the residence time is excluded, the timeout value varies more slowly compared to when most recent timestamp is used. Variations in the timeout value in the former case are due to queuing delays. In particular, if there is a retransmission at the base station, WTCP closes the wireless transmission window to just one segment. However, when an acknowledgment is received, the transmission window is opened fully, and the packets are queued on the wireless link for transmission. For an 800 Kbps link, the maximum queuing delay is 90 ms if ten 1 Kbyte packets are queued.

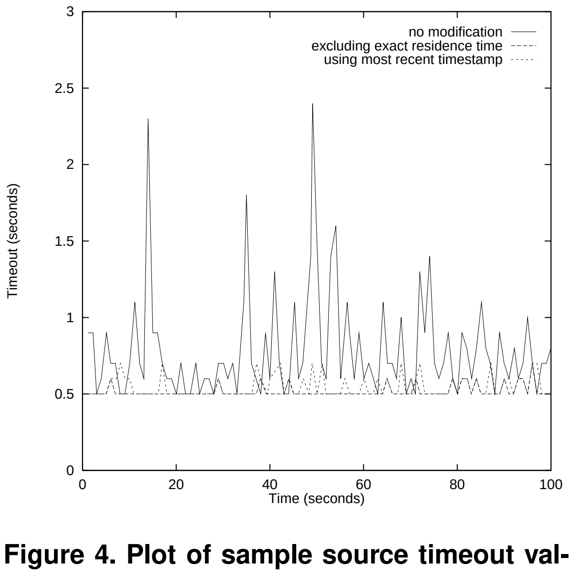

Figure 4. Plot of sample source timeout values with 100 ms clock.

[7] A trivial case is when the base station buffer is empty. When a TCP segment arrives, it is obviously the most recent segment received at the base station. Thus, in this case, the new timestamp will be the same as the original timestamp in this segment's header yielding an RTT value that indeed does not include any residence time.

If, as in most TCP implementations, a coarser clock granularity is used at the source, this queuing effect is indiscernible. In Figure 5 we show the plot of source timeout values when the source clock granularity is 300 ms. When the timestamp of the most recent segment is used or the exact residence time is excluded, the timeout value is always 900 ms (i.e. 3 clock ticks). However, if the timestamp is not modified (i.e. residence time is not excluded) the timeout value reaches as high as 2.4 seconds.

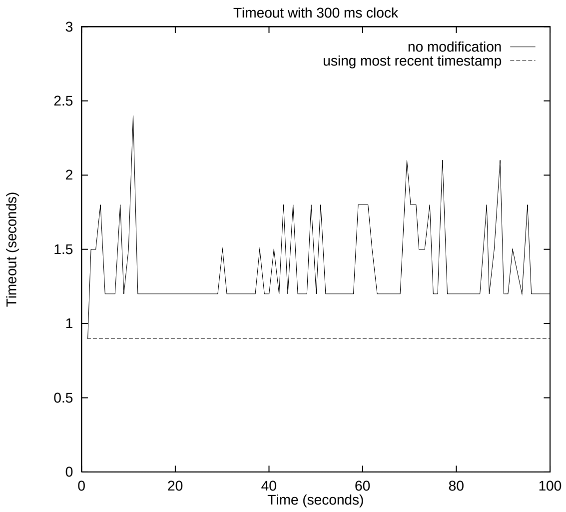

Figure 5. Plot of sample source timeout values with 300 ms clock.

In summary, if the timestamp is not modified to exclude the residence time in the base station buffer, the timeout value maintained at the source will be high. As pointed out earlier, this affects the TCP source's ability to effectively detect any congestion losses in the wired network portion of the connection. To see this effect we modeled a lossy wired link which randomly looses data packets with probability P. Figure 6 shows the throughput for WTCP without modifying the timestamp and with modifying the timestamp to exclude the residence time (using the timestamp of the most recent segment), as well as the throughput for Snoop and TCP-tahoe. The wireless link is modeled by a two-state error channel as before, and the mean good period is 1 second. The effect of residence time on the TCP source's ability to detect congestion losses in the wired link can be clearly seen in Figure 6(a) where the wired loss probability is 0.01. The advantage of hiding the residence time at the base station becomes even more pronounced for higher wired (congestion) loss probabilities. However, when there are only few wired link losses (Figure 6(b)), as expected, the number of data segments lost is low, and hence number of retransmissions is low and excluding the residence time does not provide significant improvement in throughput.

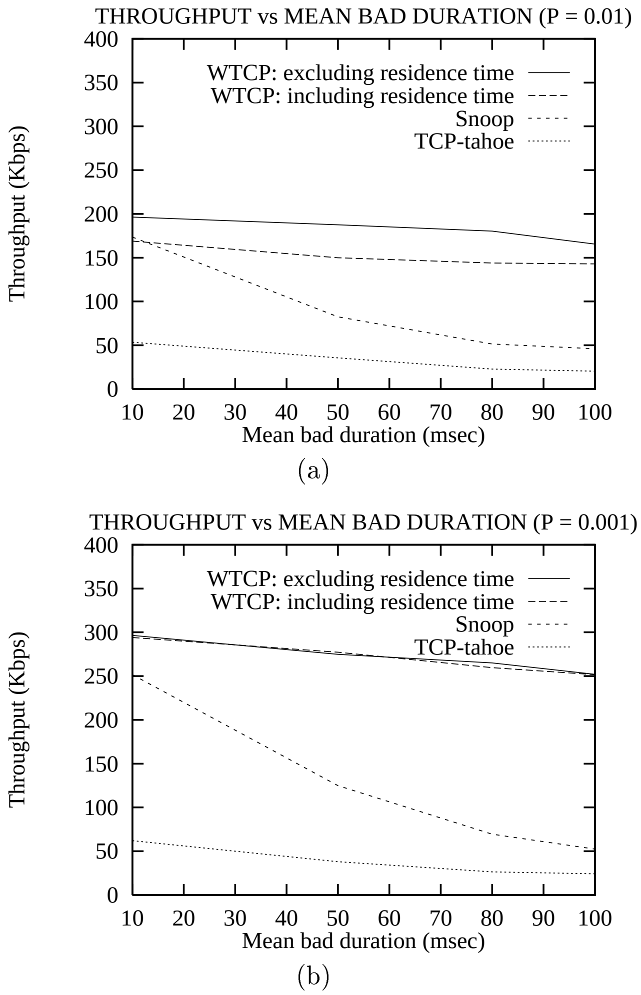

Figure 6. WTCP throughput with wired (congestion) link loss probability (a) $P = 0.01$**; (b)** $P = 0.001$**.**

4 Conclusion

We have compared the performance of transmission control protocols for networks with wireless links. When the base station buffers TCP segments and locally retransmits any lost segment to the mobile host, there can be significant gain in throughput. However, buffering and local retransmission by the base station can affect the round trip time estimation maintained by the TCP source, which in turn can impair its ability to detect congestion losses in the wired network portion of the TCP connection. We have shown that when the wired link is lossy, excluding the residence time of the

TCP segment in the base station buffer from the RTT computation can significantly improve throughput.

References

[1] A. Bakre and B.R. Badrinath. Handoff and systems support for indirect TCP/IP. In USENIX Association, editor, *Proceedings of the second USENIX Symposium on Mobile and Location-Independent Computing: April 10–11, 1995, Ann Arbor, Michigan, USA*, pages 11–24, Berkeley, CA, USA, April 1995.

[2] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz. A comparison of mechanisms for improving TCP performance in wireless networks. In *ACM SIGCOMM Symposium on Communication, Architectures and Protocols*, August 1996.

[3] P. Bhagwat, P. Bhattacharya, A. Krishna, and S.K. Tripathi. Enchancing throughput over wireless LANs using channel state dependent packet scheduling. In *Proc. IEEE INFOCOM'96*, pages 1133–40, March 1996.

[4] D. Borman, R. Braden, and V. Jacobson. RFC 1323: TCP extensions for high performance, May 1992. Obsoletes RFC1185.

[5] R. Braden, V. Jacobson, and L. Zhang. RFC 1185: TCP extension for high-speed paths, October 1990. Obsoleted by RFC1323 [4].

[6] T.V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3), June 1997.

[7] S. McCanne, S. Floyd, and K. Fall. Network simulator. Public domain software, available from http://www-nrg.ee.lbl.gov/ns/.

[8] K. Ratnam and I. Matta. WTCP: An efficient transmission control protocol for networks with wireless links. Technical Report NU-CCS-97-11, Northeastern University, July 1997. http://www.ece.neu.edu/personal/karu/papers/WTCP-NU.ps.gz.

[9] W.R. Stevens. *TCP/IP Illustrated* The Protocols. Addison-Wesley, Reading, MA, USA, 1994.

[10] G. Wright and W.R. Stevens. *TCP/IP Illustrated: The Implementation*. Addison-Wesley, Reading, MA, USA, 1995.