The TCP SACK-Aware Snoop Protocol for TCP over Wireless Networks

Sarma Vangala and Miguel A. Labrador
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
Email: {vangala,labrador}@csee.usf.edu

***Abstract*— TCP continues to be the most important transport layer communication protocol. Several solutions have been proposed to address the known problems that TCP faces when running over wireless networks. Of these solutions, the Snoop protocol, a link layer retransmission strategy, has been shown to be the most effective. However, not all TCP versions have been analyzed using the Snoop protocol. In fact, we recently showed that TCP Vegas and TCP SACK exhibit opposite performance results when utilized with and without the Snoop protocol. In this paper we analyze this behavior and introduce the TCP SACK-Aware Snoop protocol for wired cum wireless networks. We show how to make the Snoop protocol TCP SACK-Aware and why using the Snoop protocol is in fact worse than not using any mechanism at all. The TCP SACK-Aware protocol improves the performance of TCP SACK by around 30% compared to the plain Snoop protocol and by about 8% in an environment where no TCP enhancing mechanism is in place.**

I. Introduction

Since its inception 30 years ago, the Transmission Control Protocol (TCP) has grown to be the most important communication protocol and the responsible for the stability of the Internet [15]. Over the years, TCP has been modified several times to improve its performance and as a result, several important TCP versions have emerged, such as TCP Tahoe [15], TCP Reno [15], TCP Newreno [13], TCP SACK [18] and TCP Vegas [8]. However, all these new mechanisms and versions don't work the same when called to work on the diverse environments TCP has been called to work on, such as satellite networks, wireless, and wireless ad hoc networks. All these environments have peculiarities that make TCP's performance to be not as good as in wired networks.

When TCP works over wireless environments several well-known problems affect its performance. In wireless networks packets are lost due to high Bit Error Rates (BERs), signal fading, user mobility, hand-off procedures, channel asymmetries, and others, and not due to network congestion. As a result, TCP misinterprets these losses to be due to congestion and applies its congestion control algorithms unnecessarily, yielding low throughputs.

Several performance enhancing solutions have been proposed to help TCP differentiate congestion related losses from wireless losses. These solutions have been proposed at various layers of the protocol stack and can be mainly classified as link layer mechanisms, transport layer mechanisms, and also newer versions of TCP. Snoop [3], Delayed Duplicate Acknowledgments [23] and TULIP [19] are examples of link layer mechanisms. I-TCP [9] and M-TCP [4], Explicit Loss Notification [6], Explicit Bad State Notification [5] and Explicit Congestion Notification [22] are examples of transport layer mechanisms. Although newer TCP versions have been proposed to address specific issues related to wireless problems [20], [2], the main interest is in improving the performance of widely used TCP versions, such as TCP Newreno and SACK.

Of the above mechanisms, the Snoop protocol has been shown to be the best performing solution [7] [10]. In our earlier paper [24], we investigated the performance of the most important TCP versions with the Snoop protocol and found that surprisingly, TCP Vegas and TCP SACK presented opposite behaviors when run over a wired cum wireless scenario with and without the Snoop protocol. We found that the Snoop protocol improved the performance of TCP Vegas considerably but in the case of TCP SACK, the effect of using the Snoop protocol was actually negative. TCP SACK which was the best performing version and TCP Vegas which was the worst performing version without the Snoop protocol showed completely opposite behaviors when Snoop was used. In this paper this interesting behavior of TCP SACK and TCP Vegas is analyzed and the TCP SACK-Aware Snoop protocol is proposed and compared with existing solutions.

The remaining of the paper is organized as follows. Section II reviews past work in this area. Section III describes our simulation environment and the model utilized to generate the errors in the wireless channel. Section IV describes the problem that TCP SACK faces in the presence of Snoop and introduces the TCP SACK-Aware Snoop protocol. Finally, Section V concludes the paper.

II. Related Work

In this section, the most important solutions at the transport and data link layer are outlined along with their main advantages and disadvantages. Transport layer approaches include split connection mechanisms, such as I-TCP [4] and M-TCP [9], and end-to-end explicit notification mechanisms, such as Explicit Loss Notification (ELN) [3], Explicit Bad State Notification (EBSN) [5], and Explicit Congestion Notification (ECN) [22]. In general, split connection mechanisms

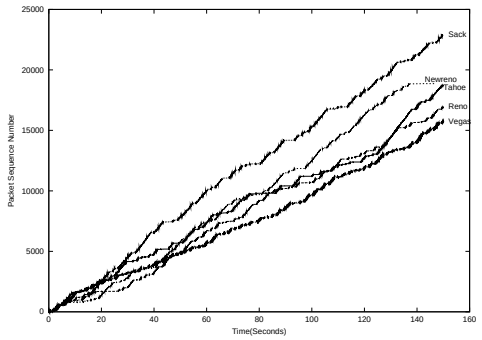

Fig. 1. TCP Performance over Wireless Networks

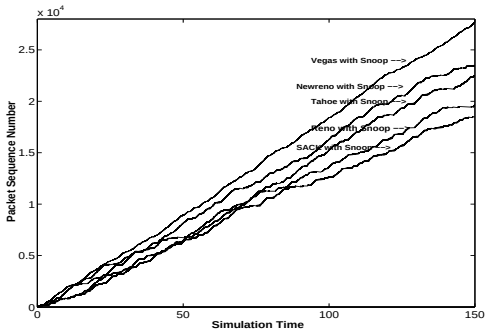

Fig. 2. TCP Performance over Wireless Networks with Snoop

have not been used because they break the end-to-end semantics of TCP. In addition, the generation of independent acks can lead to cases where the acks from the base station reach the wired sender but the packet never being delivered to the mobile receiver [9]. Explicit notification mechanisms have been shown to improve TCP's performance because they let TCP know whether the packet has been lost due to errors or congestion. These mechanisms don't improve TCP's performance as much as link layer mechanisms because of the longer round trip delays (we demonstrate this later).

Another approach to improve the performance of TCP over wireless networks is to modify the actual protocol itself. Different versions of TCP behave differently when used over error prone wireless channels. Earlier comparison studies have shown that among TCP Tahoe, Reno and Newreno, TCP Reno performs the worst due to its inability to deal with multiple loss of packets in a window of data [16]. Tahoe usually performs worse than Newreno but in some cases it can obtain similar performance values [25], [12]. Vegas and SACK were included in the comparison in [24] where we showed that Vegas performs the worst and SACK the best. TCP SACK on the other hand, is the best performing of all the TCP versions due to its ability to recover multiple losses efficiently. Figures 1 and 2 show performance results of all these TCP versions over a wired cum wireless environment with and without the Snoop protocol. These results, taken from [24] clearly show the improvements related to the use of the Snoop protocol and also the strange reverse behavior experienced by TCP Vegas and SACK.

Link layer schemes retransmit lost packets at the data link layer of the wireless hop. Since the propagation delay of the wireless last hop is considerably shorter compared to the end-to-end delay, link layer solutions have immediate knowledge of the packets dropped and can thus respond faster than higher layers. However, link layer retransmission mechanisms can sometimes adversely affect the operation of other layers leading to excessive retransmissions, as suggested in [10]. The most important link layer solutions include Snoop [3], the TCP Unaware Approach [23], and TULIP (Transport Unaware Link Improvement Protocol) [19]. TULIP is a link layer solution for half duplex links. This protocol is still in its infancy and undeveloped [19]. The TCP Unaware Approach relies on delaying duplicate acknowledgments and it has been shown that in cases of low mobility and fixed wireless, similar to that of ours, Snoop works better [21]. In the following subsections, we provide more details about the Snoop protocol and TCP SACK since they are the main mechanisms analyzed in this paper.

A. The Snoop Protocol

The Snoop Protocol [3] provides a reliable solution while maintaining the end-to-end semantics of the transport layer connection. It performs local recovery using link level buffers at the base station where passing packets across the wired-wireless link are stored. Snoop uses these cached packets to retransmit unacknowledged packets and also suppresses duplicate acknowledgments (dupacks) from the receiver to avoid unnecessary timeouts at the sender. Snoop works using two important functions, snoop_data() and snoop_ack(). The snoop_data() function processes and caches the packets going to the mobile host. It takes care of retransmitted packets as well as out-of-sequence packets. When a packet arrives in sequence and has a sequence number greater than the previous packet, Snoop caches it at the base station and forwards the packet to the mobile host. The round-trip time for this packet is also calculated by using a local clock. If an out-of-sequence packet with sequence number less than the one already acknowledged is obtained, this means a retransmitted packet form the sender has reached the base station. If this packet has already been sent to the mobile host, Snoop sends an ack back to the sender so that the sender does not timeout, otherwise, it forwards the packet to the mobile host and caches this packet as being retransmitted by the sender. The snoop_ack() function processes the acknowledgments received from the mobile host and performs retransmissions. Whenever an ack is received from the mobile host, Snoop distinguishes it as either genuine, spurious or dupack. If it is a genuine ack, i.e. an ack that is in order, it clears the local buffers, estimates the RTT, and forwards the ack to the source. If it is an ack with a sequence number smaller than the one previously received, it is a spurious ack and is discarded. If a dupack is

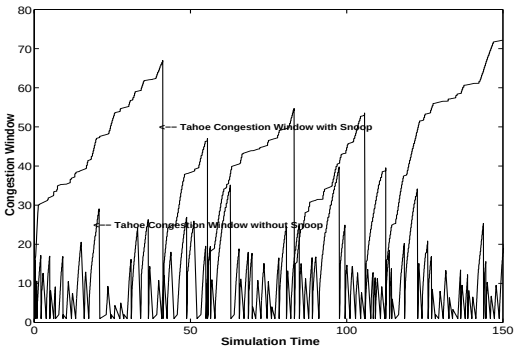

Fig. 3. Variation of Tahoe Congestion Window with and without Snoop

received for a packet not in the snoop cache or as marked for retransmission for the sender, then it is routed to the fixed host. If a dupack for a packet in the snoop cache is obtained, it retransmits the packet to the mobile host. If a dupack for a missing packet (know to snoop as missing because of a sender retransmission) is obtained, it is discarded, thus, stopping unnecessary retransmissions. Comparing Figures 1 and 2 we show that the Snoop protocol in fact, improves TCP's performance in agreement with conclusions found in other papers. The figures also show that this is true for all TCP versions except TCP SACK, which had not been analyzed before. Figure 3 also shows the positive effect of the Snoop protocol in TCP Tahoe's performance in a different way. It can be easily seen that the Snoop protocol avoids time outs considerably and maintains a larger value of TCP's congestion window at all times. These two factors are the responsible for the better throughputs obtained.

B. TCP SACK

TCP SACK was developed by Floyd and Fall [11] to take care of the inefficiency of TCP Reno in handling multiple losses in a window of data. As indicated in RFC 2018 [18] a SACK receiver is able to exactly indicate the sender using Selective Acknowledgments which packets in a sequence of data have been received correctly and which have been not. This is done using a special type of Selective Acknowledgment segment called SACK block. A SACK block provides the sender with all the necessary information needed to retransmit the exact packets that are missing. It is able to thus efficiently cope with packet losses in the wireless channel and retransmit all the missing packets in one RTT, hence reducing TCP timeouts.

The establishment of a SACK connection is done at the connection establishment phase when SYN packets are exchanged with the kind bit of the options field set to 4 (SACK enabled option). The receiver in turn acknowledges the sender of its SACK capabilities setting the kind bit to 5. Once the connection is established, the receiver sends SACK blocks in case of multiple losses in a window of data. Each SACK block reports the most recently received segments and the next acknowledgment expected in a series of data. A SACK block

Fig. 4. Simulation Topology

can indicate this information for up to three non-contiguous sequences of missing data. When received by the sender, it is able to exactly know the series of data missing and retransmits only those packets based on the availability of the congestion window.

both the sender and the receiver need to agree in implementing SACK.

III. The Simulation Environment

A simulation approach is taken to study the performance of TCP over wireless networks. We used the ns-2 simulator [17] and the wired-wireless topology shown in Figure 4. The wired node is connected to a base station using a 10 Mbps 20 msec delay link. The wireless channel is of 2 Mbps capacity with negligible delay. The packet size is fixed at 1000 bytes and the intermediate queues are assumed to be large enough so that no losses due to congestion take place and losses only occur due to the wireless errors. It is widely known that errors in wireless channels occur in bursts unlike those in wired channels where they are more random in nature. These errors can be modeled as a continuous time two-state Markov model consisting of a good state and a bad state as analyzed in [25], [26]. In our simulations, we use the error model developed in [1] which generates a slow and moderate fading channel suitable for the work done here. When the chain is in the good state packets are transmitted completely and whenever it is in the bad state they are dropped. The chain stays in each state for certain period of time transmitting many good packets and also dropping several packets together. For our study, we used an error model where the chain stays in the good state for an average of 97 msec and in the bad state for 3 msec. This model generates packet errors of up to 5% and an average burst size of about 5 packets which is reasonable for a realistic scenario. These have been common assumptions in several papers and were actually corroborated in a real wireless environment in [14]. A single TCP connection from the wired node to the wireless node is assumed. The results collected include packet sequence numbers and throughputs.

IV. The Problem with TCP SACK and the TCP SACK-Aware Snoop Protocol

Figure 2 is an excellent example of a link layer mechanism affecting the performance of a transport layer protocol and points out the importance of establishing these mechanisms very carefully. Figure 5 explains this problem much more elaborately. Assume that packets up to sequence number (seqno)10 have been transmitted successfully and that packets

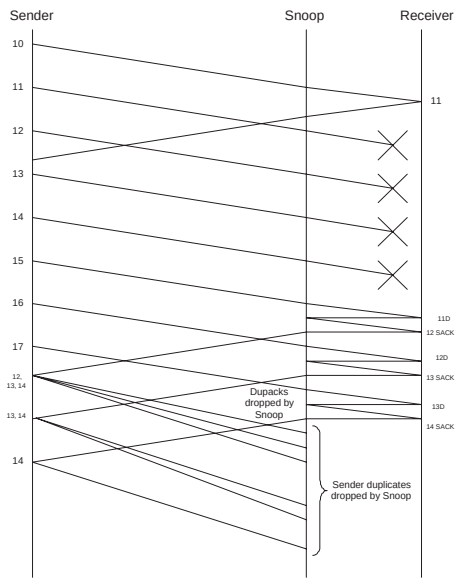

Fig. 5. The problem with TCP SACK

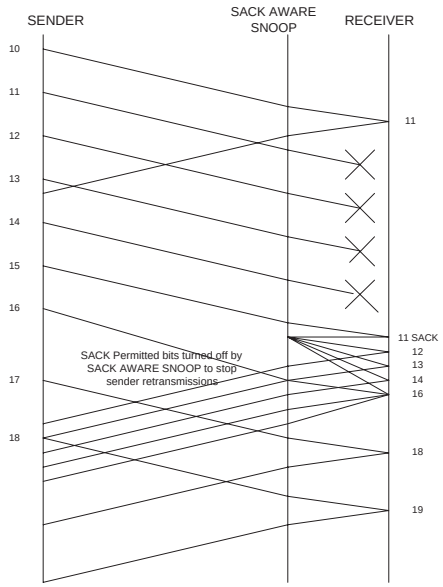

Fig. 6. The TCP SACK-Aware Snoop protocol

11, 12, 13 and 14 have been dropped due to a burst error. When packet 15 is received the receiver sends a dupack for packet 11. This dupack, when received by the Snoop protocol makes it retransmit packet 11 from its cache and drop the duplicate ack for 11. In the meantime packet 16 generates another duplicate ack for packet 11 and this also is dropped by Snoop. When packet 11 is received by the receiver it generates ack 12. These are all actually SACK blocks asking the sender to retransmit packets 11, 12, 13 and 14 in one RTT. In case where no Snoop is present a TCP SACK source would have retransmitted all those packets at once. Due to Snoop's interference only packet 11 is retransmitted and the duplicate acks (SACK acks) are dropped. The SACK sender is thus not able to retransmit all the packets in one RTT and the basic SACK mechanism is thus disrupted. If a SACK block such as that having seqno 12 reaches the SACK source a retransmission of all the packets 12, 13 and 14 is made. These upon reaching the Snoop agent are dropped as Snoop already has them in its cache. This once again leads to unnecessary retransmissions by the transport layer mechanism, this time reducing the network utilization in the wired portion of the network. TCP SACK is therefore the worst performing version with Snoop.

In order to rectify the problem of unnecessary retransmissions of SACK and enable Snoop to retransmit all the packets in one RTT, we propose the TCP SACK-Aware Snoop protocol. As the name suggests, this algorithm helps Snoop differentiate between an ordinary ack and a SACK block. In the case of an ordinary ack the TCP SACK-Aware Snoop protocol retransmits only the packet as suggested by the dupack's seqno. However, in the case of a SACK block the protocol retransmits all the packets indicated by the SACK block that are in the cache. The TCP SACK-Aware Snoop protocol checks for the acknowledgment to be a SACK block or not based on the kind bit set in its options field. Once it knows that the ack is a SACK block, it is able to determine the packets that have been lost based on the left and right edges of the 1st, 2nd or 3rd blocks. These packets, if present in the cache, are immediately retransmitted. However, if some of the packets are missing, the packets up to the highest sequence number are retransmitted. This is further explained in Figure 6 where it can be observed that after receiving the first SACK block, the TCP SACK-Aware Snoop protocol retransmits all missing packets locally at once, in a lot shorter round trip time than if the sender had retransmitted them. Following the same example, assume that packets 11, 12, 13 and 14 are lost. Assuming that all these packets are present in the TCP SACK-Aware Snoop cache, a dupack is generated for packet 11 on reception of packet 15. On reception of this SACK block, the TCP SACK-Aware Snoop retransmits the requested packets and drops the dupack after reading the left and right edges of the SACK block. The TCP SACK-Aware protocol also takes care of the unnecessary retransmissions from the SACK sender by setting the SACK permitted bit off in the SACK blocks so that the sender just sees ordinary acknowledgments. With these modifications, the TCP SACK-Aware protocol helps in reducing the sender retransmissions of packets and also enables retransmission of multiple packets in one local (and considerably shorter) RTT thus maintaining a good flow of packets. The improvement that can be achieved in the performance of TCP SACK by using the TCP SACK-Aware Snoop protocol can be seen in Figure 7. As a result, the original idea of Snoop can now also be applied to TCP SACK, providing the best performance of them all. Our results indicate that this combination achieves a throughput of 1.9902 Mbps out of the available 2 Mbps, concluding that the TCP SACK-Aware Snoop protocol hides pretty much all errors from the transport layer, which now sees a very clean channel. In Table I we summarize all our results

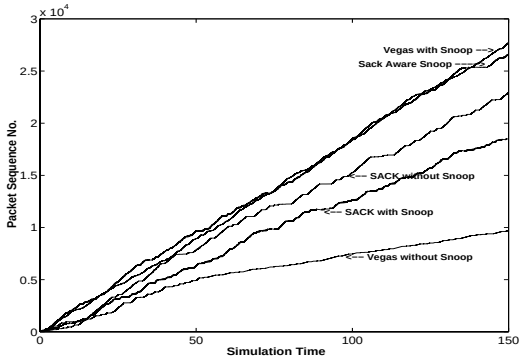

Fig. 7. Performance of TCP SACK and Vegas

TABLE I
SUMMARY OF PERFORMANCE RESULTS

TCP Version	No Enhancing Mechanism	ELN	Snoop	TCP-Aware Snoop
Tahoe	1.5190	1.6892	1.8086	N/A
Reno	1.3734	1.5126	1.6255	N/A
Newreno	1.5259	1.3556	1.8765	N/A
Vegas	1.2735	1.2736	1.9835	N/A
SACK	1.8519	1.8823	1.5394	1.9902

including the performance results obtained using the Explicit Loss Notification mechanism. It can be observed that TCP SACK's performance is improved around 30% over the normal Snoop protocol and around 8% when compared without the use of any enhancement mechanism.

V. CONCLUSIONS

In this paper we analyze the opposite behavior of TCP SACK and TCP Vegas in the presence of the Snoop protocol, one of the best known solutions for improving TCP's performance over wireless networks. In the case of TCP Vegas, the Snoop protocol hides most wireless related losses allowing TCP Vegas to see a clear channel and make good bandwidth estimations. In the case of TCP SACK, the Snoop protocol is not capable of interpreting SACK blocks and interferes negatively in the functionality of the protocol. In order to address this issue, the TCP SACK-Aware Snoop protocol is introduced to help Snoop differentiate between SACK blocks and ordinary acks. We show that this new mechanism improves the performance of TCP SACK by about 30% compared with the presence of the normal Snoop protocol and by about 8% in the case where no performance improvement mechanism is used. We also corroborate the fact that the ELN mechanism improves TCP's performance but not as much as link layer solutions.

REFERENCES

[1] A. Abouzeid, S. Roy, and M. Azizoglu. Stochastic Modelling of TCP over Lossy Links. In *Proceedings of INFOCOM*, pages 1724–1733, 2000.

[2] I.F. Akyildiz, G. Morabito, and S. Palazzo. TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks. *ACM/IEEE Transactions on Networking*, 9:307–321, June, 2001.

[3] E. Amir, H. Balakrishnan, S. Seshan, and R. Katz. Efficient TCP over Networks with Wireless Links. In *Proceedings of 5th. Workshop on Hot Topics in Operating Systems*, pages 35–41, May 1995.

[4] A. Bakre and B.R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *Proceedings of ICDCS*, pages 136–143, 1995.

[5] B. S. Bakshi, N. Vaidya, and D. K. Pradhan. Improving the Performance of TCP over Wireless Networks. In *Proceedings of 17th. International Conference on DCS*, pages 365–373, July 1997.

[6] H. Balakrishnan and R. Katz. Explicit Loss Notification and Wireless Web Performance. In *Proceedings of GLOBECOM*, November 1998.

[7] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *ACM/IEEE Transactions on Networking*, 5:756–769, December, 1997.

[8] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of SIGCOMM*, pages 24–35, 1994.

[9] K. Brown and S. Singh. M-TCP: TCP for Mobile Cellular Networks. *ACM/SIGCOMM Computer Communications Review*, 27 No.5:19–43, October 1997.

[10] A. DeSimone, M. C. Chuah, and O. C. Yue. Throughput Performance of Transport Layer Protocols over Wireless LANs . In *Proceedings of GLOBECOM*, pages 36–46, 1993.

[11] S. Floyd and K. Fall. Simulation Based Comparisons of Tahoe, Reno and Sack TCP. *ACM Computer Communication Review*, 26, No. 3:5–21, 1996.

[12] Sally Floyd and Tom Henderson. *The Newreno Modification to TCP's Fast Recovery Algorithm*. IETF RFC 2582, 1999.

[13] J. C. Hoe. Improving the Start-up Behavior of a Congestion Control Scheme for TCP . In *Proceedings of ACM SIGCOMM*, volume Vol. 26, No. 4, pages 270–280, 1996.

[14] P. Ikkurthy and M. A. Labrador. Characterization of MPEG-4 Traffic over IEEE 802.11b Wireless LANs. In *Proceedings of IEEE LCN*, pages 421–427, Tampa, FL, November 2002.

[15] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM*, pages 314–329, 1988.

[16] A. Kumar. Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link. *IEEE/ACM Transactions on Networking*, 6, No.4:485–498, 1998.

[17] Lawrence Berkeley National Laboratory. Ns-2.

[18] M. Mathis, J. Mandavi, S. Floyd, and A. Romanov. *TCP Selective Acknowledgment Options*. IETF RFC 2018, 1996.

[19] C. Parsa and J. J. Garcia-Luna-Aceves. Improving TCP Performance over Wireless Networks at the Link Layer. *Mobile Networks and Applications*, 5(1):57–71, 2000.

[20] Christina Parsa and J. J. Garcia-Luna-Aceves. Improving TCP Congestion Control Over Internets with Heterogeneous Transmission Media. In *Proceedings of the 7th IEEE International Conference on Network Protocols (ICNP)*, pages 213–221, Toronto, Canada, 1999.

[21] Kostas Pentikoussis. TCP in Wired-cum-Wireless Environments. *IEEE Communications Surveys*, Vol. 3, No. 4:2–14, 2000.

[22] R. Ramani and A. Karandikar. Explicit Congestion Notification over Wireless Networks. In *Proceedings of ICPWC 2000*, pages 495–499, July 2000.

[23] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro. Delayed Duplicate Acknowledgements: A TCP-Unaware Approach to Improve Performance of TCP over Wireless. *Technical Report, Computer Science Dept., Texas A&M University (citeseer.nj.nec.com/289955.html)*, 1999.

[24] S. Vangala and Miguel A. Labrador. Performance of TCP over Wireless Networks with the Snoop Protocol. In *Proceedings of IEEE LCN*, pages 600–601, Tampa, FL, November 2002.

[25] M. Zorzi, A. Chokalingam, and R. R. Rao. Throughput Analysis of Channels with Memory. *IEEE Journal on Selected Areas in Communications*, 18:1289–1300, July 2000.

[26] M. Zorzi, R. R. Rao, and L. B. Milstein. On the Accuracy of a First order Markov Model for Data Transmission on Fading Channels. In *Proceedings of IEEE ICUPC*, pages 211–215, 1995.