G-Snoop: Enhancing TCP Performance over Wireless Networks

Kui-Fai Leung and Kwan L. Yeung

Department of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong, PRC.
Tel: (852) 2857-8493 Fax: (852) 2559-8738 E-mail: kyeung@eee.hku.hk

Abstract: *Focusing on a general wireless network where a wireless link can be at any link along the sender-to-receiver path, a new TCP enhancement scheme, called Generalized-Snoop (G-Snoop), is proposed. Since many existing applications are built on top of TCP, it is essential that any TCP enhancement scheme should be transparent to the end-systems as well as the fixed networks. To achieve this, G-Snoop only needs to be implemented at the wireless gateways, no other parts of the network require modifications. With G-Snoop, TCP senders are shielded from non-congestion packet loss and thus no unnecessary congestion control mechanisms will be performed. Simulation results show that significant throughout gain can be obtained with G-Snoop.*

1. Introduction

Wireless data applications such as e-mail, web browsing, mobile computing, etc., are gaining increased attention due to rapid advances in the areas of wireless communications and the Internet. Transmission Control Protocol (TCP), an end-to-end reliable transport protocol in the Internet Protocol (IP) suite, is widely used in popular applications like telnet, ftp, http, etc. TCP has been designed, improved and tuned to work efficiently in the wired network where the bit-error-rate is very low. Whenever a packet is lost, it is reasonable to assume that congestion has occurred on the connection path. Hence congestion recovery algorithms are triggered to recover the loss efficiently. In other words, both congestion/flow control and error control functions are carried out simultaneously based on the simple sliding window mechanism. This enables various implementations of TCP very efficient.

Unfortunately this original design beauty becomes a hurdle in providing TCP transportation over wireless links. In a wireless network with both wired and wireless links, the assumption that packet loss infers congestion is flawed because wireless link has a much higher bit-error-rate, and a TCP connection might be temporally broken as a result of handoff or signal fading. As a result, TCP congestion recovery algorithms will be incorrectly triggered. This results in low link utilization and poor TCP performance.

Since many network applications are built on top of TCP, it is necessary to enhance TCP performance without (or with minimal) modifications to end-to-end TCP protocols. Various TCP enhancement schemes, also known as performance enhancement proxies [1], have been proposed. Most of them are designed based on the following ideas.

- Shield the sender from the peculiarities in the wireless links. Local loss recovery is usually performed at the gateway between wireless and wired networks.
- Decouple the combined error control and congestion control functions of the original TCP. This can be achieved by either designing an explicit loss notification mechanism, or an explicit congestion notification mechanism.

A brief review of the existing TCP enhancement schemes is presented in the next section.

Most of previous studies consider the scenario that the last hop/link (i.e. directly connected to the receiver) of an end-to-end TCP path is wireless. In other words, the receiver is a mobile terminal. This corresponds to the popular web-browsing applications via mobile terminals. There is also a growing interest in studying the case that the first link is wireless, or the sender is a mobile terminal. This can represent, e.g. a mobile streaming video/audio server for a live broadcast event. Relatively little attention has been paid to the general case that a wireless link can be at any link along the sender-to-receiver path, as shown in Fig. 1. In this case, the end-to-end TCP path consists of three segments, upstream network from the sender to wireless gateway A, wireless link from gateway A to gateway B, and downstream network from gateway B to the receiver. A gateway is a router that can send and receive data from a wireless link. When either upstream network or downstream network is not available, the network degenerates into the special cases we mentioned earlier.

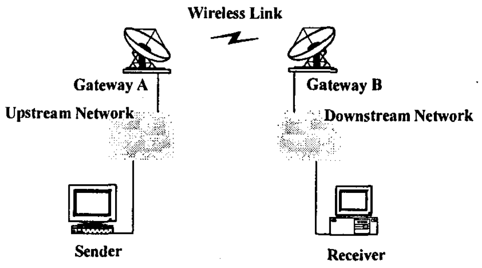


Fig. 1 A general wireless network

In this paper, we focus on designing an efficient TCP enhancement scheme called G-Snoop that can cater for the general case where a wireless link can be at any link along the sender-to-receiver path. It is essential that any TCP enhancement scheme should be transparent to the end-systems as well as the fixed networks. To achieve this, G-Snoop only needs to be implemented at the wireless gateways, i.e. gateways A and B in Fig. 1, no other parts of the network require modifications. In Section 3, the proposed scheme, G-Snoop, is presented in details. A data flow analysis is then carried out in Section 4 to demonstrate how G-Snoop can be used to handle packet loss at various locations of the network. In Section 5, simulations are conducted for performance evaluations. Finally, we conclude the paper in Section 6.

2. Existing TCP Enhancement Schemes

A. Split Connection

The Indirect-TCP (I-TCP) [2] protocol was one of the first protocols to use this method. It involves splitting a TCP connection between a fixed and mobile host into two separate connections at the mobile base station. Since the second connection is over a one-hop wireless link, an optimized wireless link-specific protocol tuned for better performance can be used. The advantage of the split connection approach is that the flow and congestion controls of the wired and wireless links are separated. However, this approach suffers from breaking the end-to-end TCP semantics, requiring application relinking and higher software overhead.

B. Preventing Sender Timeout

This approach focuses on addressing the TCP performance when communication resumes after a temporary disconnection, e.g. handoff. When a timeout occurs, the system enters slow start and the window size is reduced to 1. Usually handoffs complete relatively fast, and long waits are required by the mobile host before timeouts occur at the sender.

To reduce this waiting period, [3] makes the mobile host retransmit 3 copies of the ACK for the last data segment it received priori to the disconnection, immediately after completing the handoff. This causes TCP at the sender to immediately reduce its window size and retransmit packets starting from the first missing one (for which the duplicate acknowledgment was sent). Freeze-TCP [4] uses idea of "shrinking its window size to zero" together with that of [3]. But like other protocols following this approach, they are incapable to handle packet corruption at the wireless links.

C. Link-level Retransmission

In this approach, the wireless link implements a retransmission protocol coupled with forward error correction at the data-link level [5,6]. The advantage is that it improves the reliability of communications independent of the higher-level protocol. However, TCP implements its own end-to-end retransmission protocol. This duplicates the effort. Studies have shown that independent retransmission protocols can lead to degraded performance, especially when error rates become significant.

D. TCP-aware Loss Recovery

The enhancement schemes are designed using the knowledge obtained from the TCP layer by snooping. In [7], it is proposed to delay the duplicate ACKs (acknowledgments) for a missing packet in order to allow any special local retransmissions on the wireless links to work. In [8], an explicit bad-state notification (EBSN) scheme is proposed. The idea is to prevent the sender from dropping congestion window (when packet corruption occurs) by generating explicit notifications at the base station.

Recently, Snoop protocol [9] started a new trend of performing smart local retransmission based on the snooped TCP information at the base station. A snoop agent monitors every packet that passes through the TCP connection and maintains a cache of TCP packets sent across the wireless link that have not yet been acknowledged by the mobile receiver. A packet loss is detected by the arrival of a small number of duplicate ACKs from the receiver or by a local timeout. The snoop agent retransmits the lost packet if it has it cached and suppresses the duplicate ACKs. Like other local retransmission schemes, Snoop also suffers from not being able to completely shield the sender from wireless losses. In [10], New Snoop has been proposed by us to overcome this problem. Combining with a two-layer hierarchical cache architecture, we showed that New Snoop handles both local loss recovery and seamless handoff efficiently.

Due to limited bandwidth and high-bit error rates, wireless links usually become the bottleneck in an end-to-end connection path. In [11], we designed an efficient flow control scheme at the base station, called FDA. FDA should be functioned together with either Snoop or New Snoop. Implementing FDA also only needs code modifications at the base station.

E. Adopted Protocols

There are a few protocols that are originally designed for other purposes, but nevertheless, they can be utilized/adopted for improving the TCP performance over wireless links. These include TCP-SACK (selective acknowledgment) [12] and ECN (explicit congestion notification) [13]. TCP-SACK was originally proposed to recover quickly from multiple packet losses within a single transmission window. Later studies [14] showed that SACK can also be used to improve the performance over the wireless link. ECN scheme was originally proposed for minimizing the packet loss due to congestion, thereby without triggering time-consuming end-to-end loss recovery too frequently. Recently, an Internet Draft [15] proposed to apply ECN for improving TCP performance over wireless links so that a TCP sender could distinguish the cause of packet lost.

F. Mobile TCP Sender

All the above five approaches (except [9]) assume that the sender is inside the fixed network, and the receiver is a mobile host. This is a reasonable assumption in most cases. But there is also a growing need to cater for the vice versa scenario. For example, a live broadcast from a crime scene over the Internet via a first-hop wireless link. In [9], a mechanism based on SACK was proposed when the TCP sender is a mobile host. In [16], the same authors proposed a refined solution called Explicit Loss Notification (ELN). The idea is to explicitly detect and inform the sender which packet loss is caused by wireless link errors.

3. Our Approach

Unlike previous approaches, we consider a general network where a wireless link can be at any link along the sender-to-receiver path as shown in Fig. 1. To shield the sender as well as the receiver from the peculiarities of the wireless links, G-Snoop is implemented at the two gateways connected by the wireless link. Refer to Fig. 1. The main function of gateway A is to cache the packets that have been transmitted to gateway B. In case of packet corruption on the wireless link, as reported by gateway B, gateway A carries out local loss recovery if a copy of the corrupted packet can be found in its cache.

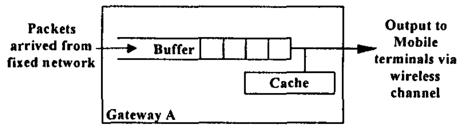
Gateway B has two main tasks, detecting and reporting packet corruption, and suppressing duplicate ACKs (acknowledgements) generated by the receiver for already correctly recovered packets. The resulting scheme is totally transparent to the TCP implementation at the two end-systems as well as other parts of the network. The detailed protocol mechanisms are discussed below.

A. At Gateway A

For each flow/connection, gateway A keeps track of its incoming packet sequence numbers. Any missing packet creates a "hole" in its sequence number table. In case of an out-of-order packet arrival, the hole will be filled up very quickly. Otherwise, the hole indicates the packet with the corresponding sequence number is lost.

The buffer at the gateway A is shown in Fig. 2. When a packet arrives, it waits in the FIFO buffer for transmission over the wireless link. When it has been sent, a copy of it will be stored in the FIFO cache for possible local loss recovery later on. The cache size should be large enough to store at least the maximum number of packets can be transmitted on one wireless link round-trip-time. Otherwise, cached copies of the transmitted packets will be prematurely flushed (i.e. pushed out) before the retransmission request arrives.

When a NACK (negative ACK) generated by gateway B is received, gateway A checks its hole table. (NACK is a special ACK used only between the two gateways.) If there is no match, gateway A checks its cache for a cached copy of the requested packet. If the cache size is large enough, we can always find a cached copy of the requested packet. Then gateway A carries out the local packet retransmission. In case of a cache-miss, the request is ignored and no further action will be taken. This happens when the cached copy of the requested packet has been pre-maturely flushed.



Packets arrived from fixed network
Buffer
Cache
Gateway A
Output to Mobile terminals via wireless channel

Fig. 2 Buffer design at gateway A.

If the requested packet by NACK corresponds to a hole in the hole table, that means this packet was lost in upstream network (most likely due to congestion in upstream network), *not* at the wireless link. An incipient congestion notification (ICN) function is then activated. The purpose is to inform the corresponding sender to slow down its transmission rate at an earlier time. To achieve this, gateway A makes three copies of the next received ACK, which belongs to the same TCP connection of the requested packet by NACK, and sends them to the corresponding sender. When the sender receives three duplicate ACKs, it resends the "lost" packet and slows down its transmission rate.

It should be noticed that this retransmitted packet is *not* the packet requested by NACK and is thus *not* really a lost packet. Besides, gateway A cannot generate three duplicate ACKs for the requested packet because ACK in TCP is cumulative. That would cause misunderstanding to the sender that all packets with sequence numbers smaller than that of the received ACK have been correctly received.

The downside of the ICN function is that the receiver is not aware gateway A has already throttled the sender's transfer rate. As a result, when duplicate ACKs generated by the receiver (for the same packet as requested by NACK) arrive at the sender later on, they will trigger the sender to go through another round of rate reduction. If the round trip time on downstream network and wireless link is significant, the gain brought by incipient congestion notification would outperform the potential loss. We can see this from the simulation results in Section 4.

B. At Gateway B

Gateway B carriers out two main functions: (i) detecting and generating NACK for packets that might be corrupted on the wireless link, and (ii) suppressing unnecessary duplicate ACKs generated by receivers for already local recovered packets.

Gateway B also keeps track of the sequence number of incoming data packets for each TCP connection. When a sequence number gap is detected and the gap cannot be filled up by the next three (out-of-order) packet arrivals, a special NACK will be generated and forwarded to gateway A. The threshold of 3 out-of-order packets is chosen in order to be in line with sender's Fast Retransmit triggering mechanism. This is to ensure the missing packet is not due to the possible out-of-order packet delivery.

When a NACK is sent, gateway B initializes a counter, denoted as O3_counter, for counting the number of out-of-order packet arrivals before the requested repair is received. If the packet requested by NACK is lost in upstream network instead of at the wireless link (i.e. a hole-hit at gateway A), gateway B will not get the retransmitted packet from gateway A within a wireless link RTT (plus some guard time). Then the associated O3_counter is reset.

The following example shows how O3_counter can be used to suppress the duplicate ACKs for a corrupted, but already locally retransmitted/recovered packet. Suppose packet 4 is corrupted at the wireless link. When packets 5, 6 & 7 arrive at gateway B, gateway B realizes that packet 4 is lost and it generates an NACK to gateway A. Upon receiving this NACK, gateway A performs the local loss retransmission of packet 4 (assuming there is a cached copy at gateway A). When the retransmitted packet 4 arrives at gateway B, gateway B has already received packets 8, 9, & 10. In other words, the packet receiving sequence at gateway B (which is also the packet forwarding sequence to the downstream network) is "...3, 5, 6, 7, 8, 9, 10, 4 ...". When those packets arrive at the receiver subsequently, the receiver will generate 6 duplicate ACKs for packet 4 before packet 4 is received. If those duplicate ACKs cannot be suppressed by gateway B, they will trigger unnecessary Fast retransmit procedure at the sender.

With the ACK suppression function, when the retransmitted packet 4 arrives at gateway B, 6 packets have been received by gateway B as out-of-order packets, so O3_counter = 6. When the first duplicate ACK for packet 4 arrives at gateway B subsequently, gateway B retrieves the associated O3_counter, and the next 6 duplicate ACKs (including the current one) will be suppressed.

Note that if the retransmitted packet 4 is lost again in downstream network, the number of duplicate ACKs received by gateway B will be larger than 6 (its O3_counter), so gateway B will not suppress useful duplicate ACKs.

3. Data Flow Analysis

In this section, we describe in details about the series of actions to be taken by the two gateways in case of (1) congestion loss in upstream network, (2) packet corruption on the wireless link, (3) cache-miss at gateway A, and (4) congestion loss in downstream network.

A. Congestion loss in upstream network

If congestion occurs in upstream network (including gateway A), some packets will be dropped as a result of buffer overflow (or action by active queue management scheme such as RED). A missing data packet creates a sequence number gap and will be detected by gateway A as a "hole" in its hole-table. Similarly, gateway B detects the missing packet and generates a retransmission request (NACK) to gateway A after three out-of-order packets have been received.

When gateway A receives the NACK, it checks against its hole-table. If the sequence number of the packet specified in the NACK matches a hole, this confirms that the requested packet is lost in upstream network as a result of congestion. The incipient congestion notification (ICN) function is activated. Gateway B informs the associated sender by generating three duplicate ACKs of the next arrived ACK of the same connection. This can slow down the sender earlier than waiting for the duplicate ACKs generated by the receiver to arrive.

B. A packet is corrupted at the wireless link

When gateway A receives a NACK generated by gateway B, it checks its hole-table. If there is no hole-hit, the requested packet is corrupted in the wireless link. Then gateway A checks its cache for a repair. If the cache size is large enough, a copy of the corrupted packet can always be found and local retransmission can then take place.

C. Cache-miss at gateway A

If the cache size at gateway A is not large enough, a copy of the requested packet will be prematurely flushed. This results in a cache-miss. In this case, no attempt will be made by gateway A to retrieve a copy of the corrupted packet from the associated sender. This is because the loss recovery in this case is not urgent and there is no congestion being detected in the network. Therefore we do not want to put extra processing burden at gateway A.

D. Congestion loss in downstream network

If congestion occurs in downstream network, some packets will be lost as a result of buffer overflow or random discarding. The receiver will detect the sequence number gap in the received packet stream. Duplicate ACKs will be generated by a standard TCP receiver. Those duplicate ACKs will be forwarded by gateways B and A all the way to the sender. Triggered by three duplicate ACKs, the sender carries out the standard Fast Retransmit and Fast Recovery procedures.

4. Performance Evaluations

Since there is no other schemes that target at a general scenario as us in this paper, the performance of G-Snoop is not compared with others directly. Instead, we focus on the performance of TCP Reno that deploys our proposed scheme or not.

The G-Snoop is implement using the LBNL network simulator ns version 2.1b7a [17]. The simulated network is shown in Fig. 3. The network consists of five TCP senders, five TCP receivers, two intermediate nodes and two intermediate transmission gateways where G-Snoop is implemented. The buffer size at all intermediate routers and gateways is 140 packets and is managed by drop-tail queue management policy.

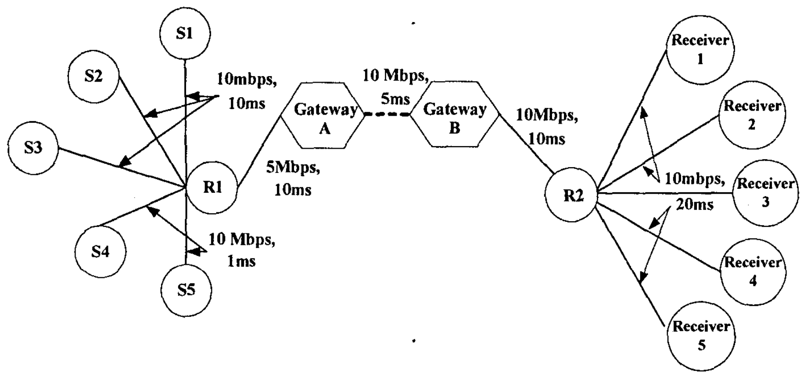


Fig. 3 Simulation network topology.

In the simulation, a large file is sent from each sender to the receiver by using FTP. All links (both wired and wireless) are of 10 Mbps except the wired link connected to gateway A. It has a capacity of 5 Mbps and thus it is the system bottleneck. The packet loss rate at the wireless link is uniformly distributed. Simulation for the wireless link with two different propagation delays, 5ms for a terrestrial wireless link and 275 ms for a typical satellite link, are conducted.

ACKs are assumed not experiencing any loss due to its small packet size. Table 1 summarizes other major simulation parameters.

Parameters	Symbols	Values
Buffer size at all intermediate routers	Buf_size	140 pkts
Cache size at gateway A	Cache_size	Variable
The prob. Of packet loss at wireless link	P_s	$10^{-1}\sim10^{-5}$
Transmission rate of wireless link	Rate $_{wl}$	10Mbps
The round trip time of wireless link	RTT $_{wl}$	5ms, 550ms
Initial TCP slow start threshold	SSTHRESH	64 pkts
TCP packet size	Packet_size	512 bytes
ACK size	ACK_size	40 bytes

Table 1 Simulation parameters

Goodput and fairness performance of G-Snoop are evaluated and their definitions are as follows:

$$\text{Goodput} = \frac{\underline{total\_\#\_of\_packet\_received\_successfully}}{Simulation\_time}$$

$$\text{Fairness} = \frac{(\sum_{i=1}^{n} bi)^2}{n(\sum_{i=1}^{n} b_i^2)}$$

where n is the total number of connections and b_i is the fraction of the bandwidth occupied by connection i. The value of Fairness ranges from $1/n$ to 1 with 1 denoting equal bandwidth sharing.

A. Wireless link with 5 ms propagation delay

First we consider the network shown in Fig. 3 with a wireless link of 5ms propagation delay. The initial/basic cache size at gateway A is set to 25 packets, which is slightly larger than the wireless link round-trip-time equivalent bandwidth.

Goodput and fairness performance against wireless link packet loss rate are plotted in Figs. 4 & 5. From Fig. 4, we can see that with G-Snoop, TCP Reno performance is significantly improved (as compared with the pure TCP Reno) when the wireless packet loss rate is from 10^{-1} to 10^{-3}. Besides, by varying the cache size at gateway A from 25 to 75, there is a diminishing gain in goodput using G-Snoop. This is because the majority of packet losses on the wireless link are recovered within one wireless link round trip time, and the excess cache size is of limited use.

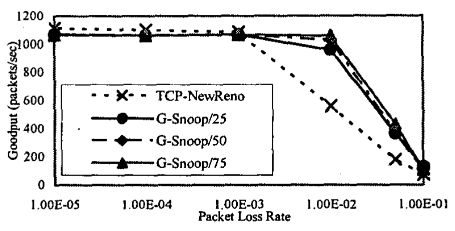


Fig. 4 Goodput vs packet loss rate

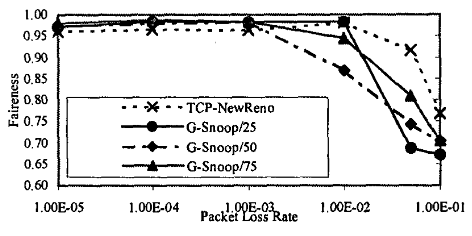


Fig. 5 Fairness vs packet loss rate

When the packet loss rate is less than 10^{-3}, it is interesting to note that with G-Snoop, the goodput is in fact a little bit less than the case without the scheme. This is caused by the incipient congestion notification (ICN) function of G-Snoop. As we mentioned earlier, when the packet loss rate is low and the end-to-end propagation delay is small, the gain in having ICN to inform the sender to slow down in advance is less than the loss due to the duplicate efforts in flow control.

Fig. 5 shows the fairness performance against the packet loss rate. It can be observed that all schemes give pretty good fairness performance with a typical value larger than 0.7.

Referring to Section 3A, we test the effect of function ICN by elongating the propagation delay of downstream network. Simulation parameters keep the same as the previous one except the propagation delay of downstream network is doubled, i.e. 60 ms. Fig. 6 shows the overall goodput performances of using the G-Snoop scheme and without using this scheme. Similar to the previous, a considerable performance gains in goodput are preserved, and the adverse effect in the packet loss rate range of 10^{-5} to 10^{-3} is removed.

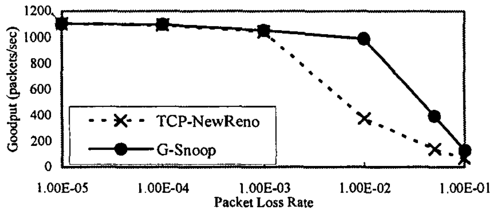


Fig. 6 Goodput vs packet loss rate by elongating the downstream network

5. Conclusion

In this paper, we have considered a general wireless network where a wireless link can occur at any link along the sender-to-receiver path. A new TCP enhancement scheme called G-Snoop was proposed to be transparently implemented at the wireless gateways. This helps to eliminate the adoption problem in a wireless network. With G-Snoop, TCP senders are shielded from non-congestion packet loss due to wireless links and thus no unnecessary congestion control mechanisms will be performed. Simulation results have shown that significant throughout gain can be obtained with G-Snoop.

Acknowledgements

This work was supported by Competitive Earmarked Research Grant HKU 1180/00E.

References:

[1] J. Border, M. Kojo, J. Griner, G. Montenegro and Z. Shelby, "Performance Enhancing Proxies," draft-ietf-pilc-pep-05.txt, Nov. 2000

[2] Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," Proc. 15th International Conf. on Distributed Computing Systems(ICDCS), May 1995.

[3] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE Journal on Selected Areas in Communications, 13(5), June 1995.

[4] T. Goff, J. Moronski, D.S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," IEEE INFOCOM, April 2000.

[5] E. Ayanoglu, S. Paul, et.al, "A Link-Layer Protocol for Wireless Networks," ACM ACM/Baltzer Wireless Networks Journal, 1:47--60, February 1995.

[6] C. Parsa and J.J. Garcia-Luna-Aceves, "Improving TCP performance over wireless networks at the link layer," Mobile Networks and Applications, pp. 57-71, May 2000.

[7] M. Mehta and N.H. Vaidya, "Delayed duplicate acknowledgments: a proposal to improve performance of TCP on wireless links," Texas A&M University, Dec. 1997. (http://www.cs.tamu.edu/faculty/vaidya/Vaidya-mobile.html)

[8] Bakshi, et.al, "Improving performance of TCP over wireless networks," Texas A&M University, Technical Report TR-96-014, May 1996.

[9] H. Balakrishnan, S. Seshan, and R.H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," ACM Wireless Networks, 1(4), December 1995.

[10] J.H. Hu, K.L. Yeung, et al "Hierarchical Cache Design for Enhancing TCP over Heterogeneous Networks with Wired and Wireless Links", IEEE GLOBECOM 2000, San Francisco, Nov. 2000.

[11] J.H. Hu and K.L. Yeung, "FDA: A Novel Base Station Flow Control Scheme for TCP over Heterogeneous Networks," IEEE INFOCOM 2001, Anchorage, Alaska, April 2001.

[12] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and Sack TCP," Computer Communications Review, 1996.

[13] K.K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," Internet draft draft-ietf-tsvwg-ecn-02.txt, February 2001. (being submitted to advance as Proposed Standard)

[14] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," IEEE/ACM Trans. On Networking, pp. 756-769, Dec. 1997.

[15] F. Peng and J. Ma, "A proposal to apply ECN into wireless and mobile networks," Internet Draft draft-fpeng-ecn-03.txt, Jan. 2001.

[16] H. Balakrishnan and R. H. Katz, "Explicit Loss Notification and Wireless Web Performance," IEEE GLOBECOM 98, Internet Mini Conference, Sydney, Nov. 1998.

[17] S. McCanne and S. Floyd. Ns-LBNL Network Simulator. URL: http://www-nrg.ee.lbl.gov/ns/