Design of SNACK Mechanism for Wireless TCP with New Snoop [1]

Fanglei Sun and Victor O.K. Li
Dept. of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong, China
Email: {flsun, vli}@eee.hku.hk

Soung C. Liew
Dept. of Information Engineering
The Chinese University of Hong Kong
Hong Kong, China
Email: soung@ie.cuhk.edu.hk

***Abstract*—TCP is the most widely adopted transport layer communication protocol. In heterogeneous wired/wireless networks, however, the high packet loss rate over wireless links can trigger unnecessary execution of TCP congestion control algorithms, resulting in performance degradation. TCP performs poorly on wireless links with bursty losses, when it is forced to rely on limited information available from batched acknowledgements, (i.e., multiple packets are acknowledged with one acknowledgment packet). In this paper, a Selective Negative Acknowledgement (SNACK) mechanism is designed to overcome the limitation of batched acknowledgments. A new link layer retransmission protocol, called, SNACK-NS (New Snoop), is proposed. Through the detection and retransmission functions that are provided by the two protocol components of SNACK-NS, namely, SNACK-Snoop and SNACK-TCP, the transmission performance of TCP over wireless network is greatly enhanced in both fixed host (FH) to mobile host (MH) and MH to FH transmissions.**

I. INTRODUCTION

During the past few years, the increase in the number of competing technologies and service network models available to the public has accelerated the growth of the wireless Internet. The congestion control algorithms embedded in TCP work well in wired networks in preventing congestion collapse. One problem of TCP congestion control in heterogeneous wired/wireless networks is that TCP regards both wired and wireless packet losses as indications of network congestion. Even when packet losses are due to noise in the wireless medium rather than congestion, TCP and its extended versions, such as TCP Reno, TCP Newreno, TCP SACK, will begin executing their congestion control algorithms, by blindly reducing the congestion window size. How to allow TCP to distinguish between the losses due to congestion and those due to packet corruption in a timely fashion is a major research issue for wireless TCP. Several approaches [1] to address this problem have been proposed. These approaches can be divided into end-to-end mechanisms like Veno [2], split connections mechanisms like M-TCP [3] and localized link layer mechanisms like Snoop [4,5]. Performance comparisons of some of these mechanisms were given in [1], in which the localized link layer solution was demonstrated to be superior in terms of throughput performance when the packet loss rate is high [1]. In addition, these mechanisms can be implemented easily and have fast response to wireless random losses.

In this paper, we consider the transmission between a fixed host (FH) and a mobile host (MH) relayed through a base station (BS), as shown in Figure 1. The main advantage of employing a link-layer protocol for loss recovery is that it fits naturally into the layered structure of network protocols. At the same time, it has more control over the physical layer protocols. The protocol that runs on top of the physical layer has immediate knowledge of dropped frames, and thus can respond faster than higher layer protocols. The Snoop protocol (Snoop) installed at the link layer of a BS monitors the packets and ACKs in both MH to FH and FH to MH directions.

For transmission from FH to MH, Snoop caches the packets arriving at BS. When packets are lost in the link from BS to MH, BS arranges local retransmissions based on the type of ACKs from MH and local timers. For packets from MH to FH, Snoop at BS adds explicit loss notification (ELN) [6] in the ACKs to the MH, setting the value of one bit in the six reserved bits included in a TCP header, thus allowing MH to distinguish congestion losses from wireless random losses. However, Snoop can only provide single packet loss information within one local RTT (round-trip-times). Under high loss rate wireless network environment, Snoop does not work well because it mimics the TCP error recovery mechanism, which is not very robust under harsh error conditions. In bursty traffic networks, the lack of explicit and accurate information in Snoop degrades the bandwidth utilization sharply. Furthermore, Snoop offers great improvement in the model of wired-cum-wireless networks. But when used in wireless-cum-wired or wireless-cum-wireless networks, Snoop is regarded as ineffective [7].

When multiple packets are lost in a TCP window and within one RTT, with the limited information available from cumulative acknowledgments (ACKs) by TCP, the congestion window size will be reduced continuously, degrading the throughput nearly to zero. To overcome this limitation, a selective acknowledgment (SACK) mechanism was proposed in RFC 2801 [8]. In TCP SACK, several SACK blocks are used to inform the sender about all the segments that have

1 This research is supported in part by the Areas of Excellence Scheme established by the University Grants Committee of the Hong Kong Special Administrative Region, China, under project No. AoE/E-01/99.

been received successfully, which allows the sender to retransmit only the lost segments. Each SACK block consists of the beginning and the ending sequence number of a consecutive packet block received by the sender, and thus the holes between the SACK blocks are regarded as lost packets. TCP SACK and Snoop have been combined in many papers to enhance the TCP performance over wireless links with burtsy losses. However, apart from other limitations of SACK itself, such as redundant transmissions caused by using TCP options, aggressive retransmissions in the presence of congestion and unnecessary retransmissions when a number of successive ACK packets are dropped in the network during a fast recovery period [9], the mechanisms of TCP SACK and Snoop may interfere with each other in both directions when combating bursty wireless losses. The detailed analyses will be presented in Section II. So, it is impractical to solve the problem of bursty losses over wireless networks by using the combination of TCP SACK and Snoop. Recently, much research has been focused on designing a new ACK [10] for wireless TCP. Unfortunately, it encounters the same problems as TCP SACK.

In this paper, SNACK mechanism running on BS and MH (note: as opposed to SACK running on FH and MH) is designed to provide explicit information on multiple packet losses over wireless links. Then, based on the SNACK mechanism, two protocol components are proposed to overcome the above problems. They are the new Snoop protocol deployed on BS (SNACK-Snoop) and TCP deployed on MH (SNACK-TCP). When handling the wireless losses, by performing the detection and retransmission functions, faster recovery and more effective congestion avoidance over wireless links can be provided. In addition, in the two data transmission directions, SNACK-Snoop and SNACK-TCP operate similarly, differing only in the way the detection and retransmission functions are divided. Compared with other protocols, the analyses in Section II and the simulation results in Section IV show that the two protocol components can improve the network throughput and enhance the TCP performance over wireless networks greatly.

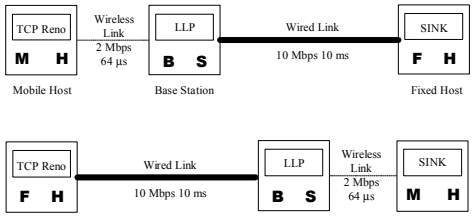
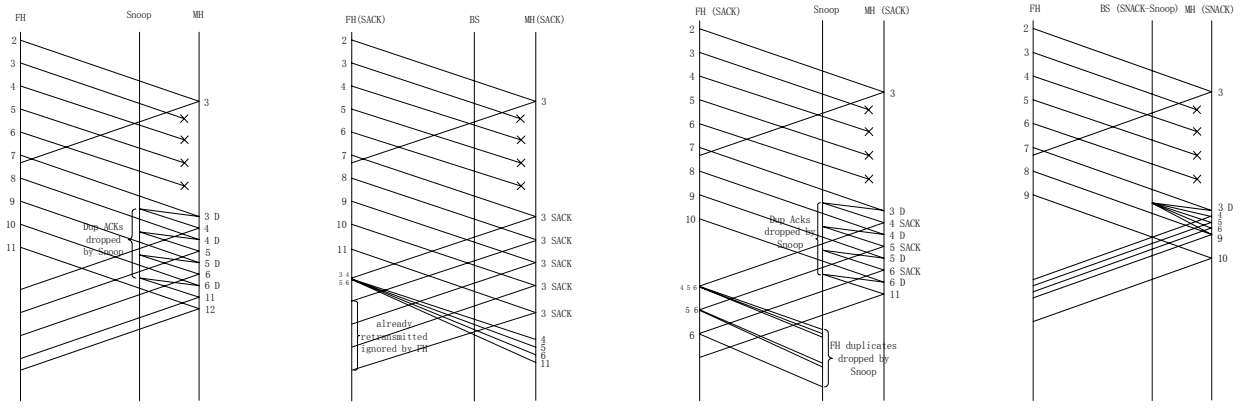


Figure 1. One wireless link simulation network topologies

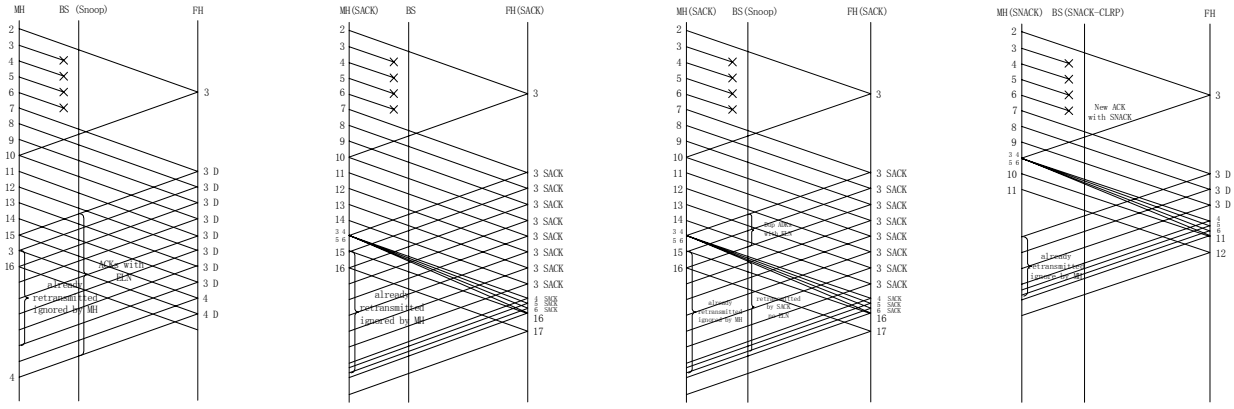
II. RELATED WORK

With random noise, multi-path fading and mutual user interference in wireless channels, bursty losses may become a tough problem for wireless TCP. The simulation-based performance of several TCP versions over wireless networks with and without Snoop has been analyzed on heterogeneous networks in the transmission direction from FH to MH [11,12]. It has been found that TCP SACK is the best performing version without Snoop, but it is the worst version with the aid of Snoop. This result is identical with our simulation result under a comparable network environment. Furthermore, we find that except for TCP Vegas, the enhancement of TCP Tahoe, Reno, Newreno and TCP SACK is limited when implemented on networks primarily with bursty losses. In addition, we also study the performance of the TCP versions with and without Snoop in the direction from MH to FH. It is puzzling that TCP SACK does not achieve additional improvements with the help of Snoop, as shown in Figure 6. Certainly, there are some problems with Snoop or the interactions between Snoop and TCP SACK in heterogeneous networks with bursty losses. So in the remainder of this section we describe our analyses of a scenario with four packets dropped from a window of data in both directions.



(a) Snoop (b) SACK (c) SACK with Snoop (d) SNACK-NS

Figure 2. Recovery from four drops in FH to MH direction



(a) Snoop (b) SACK (c) SACK with Snoop (d) SNACK-NS

Figure 3. Recovery from four drops in MH to FH direction

From FH to MH: Figure 2 (a) shows the Snoop recovery steps after the drops of Packets 3 to 6. Snoop maintains a cache of unacknowledged data segments destined to MH. When Snoop receives a duplicate acknowledgement from MH destined for FH, it retransmits the missing segments and suppresses the duplicate acknowledgment instead of forwarding it to FH. Snoop also uses timeouts to locally retransmit segments, if needed. These timeouts are less coarse then the de facto TCP timeouts, and therefore expire sooner, leading to a local retransmission within the time span of a TCP timeout. It can be seen that Snoop also uses duplicate ACKs and timeouts to retransmit the wireless losses, which mimics the recovery mechanism of TCP. Therefore when handling bursty losses, it is also inefficient. This also explains why TCP Tahoe, Reno, Newreno and TCP SACK only have limited improvements over links with bursty losses. Different from Snoop, in Figure 2 (b), TCP SACK can retransmit the four losses in one RTT by getting the loss information piggy-backed by the SACK. Therefore, it performs better than other TCP versions without Snoop. However, when combining TCP SACK with Snoop, as shown in Figure 2 (c), Snoop retransmits the lost packets indicated by the duplicate ACKs and suppresses those duplicate ACKs on which the SACK information is also piggy-backed, and thus FH cannot respond to the SACK in a timely fashion. After the delayed SACK arrives at FH, the sender retransmits the losses that have already been retransmitted by Snoop earlier. Finally, when the redundant retransmission packets arrive at BS, Snoop drops them all. In other words, Snoop delays the multiple loss information piggy-backed in SACK and TCP SACK causes redundant transmissions. Therefore, TCP performs worst with Snoop under the same network environment.

From MH to FH: In Figure 3 (a), for the ELN information mainly piggy-backed in duplicate ACK by Snoop, MH can identify the wireless random losses, allowing MH to retransmit the wireless losses without blindly starting up the congestion control. Different from transmission in the FH to MH direction, however, reliable transmission is provided by a wired channel, and therefore the corresponding retransmissions are implemented by MH. Thus the recovery of multiple losses should experience a longer delay. In Figure 3 (b), for the same reason as presented in the above paragraph, TCP SACK is still the best when compared with other TCP versions without Snoop in the MH to FH direction. However, in Figure 3 (c), when the duplicate ACK with SACK information arrives at MH, TCP SACK immediately retransmits all four losses, of course including the wireless losses indicated by the ELN flag. Therefore, Snoop does not provide any performance improvement over TCP SACK, and of course leads to similar simulation results shown in Figure 6.

III. IMPLEMENTATION OF SNACK

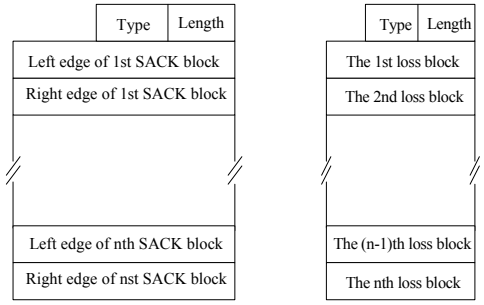
Based on the above analyses, in this section, the SNACK mechanism is designed to provide explicit information on multiple packet losses on wireless links. Combined with SNACK, a new link layer retransmission protocol, named SNACK-NS, is proposed to overcome the conflict between TCP SACK and Snoop, and to enhance the TCP performance over wireless links with high packet loss rates and stubborn bursty losses.

A. SNACK Mechanism

Like TCP SACK, SNACK also uses the TCP option. Figure 4 compares their structures. In SNACK, explicit loss information is conveyed by several loss blocks and each block stores the sequence number of the most recent wireless loss judged by the associated detecting protocol. According to the network model and the utilization of other TCP options, such as the timestamp option in RTTM [10], the maximum number of loss blocks accessed by an ACK is decided (The value 6 is used in our simulations). According to the current loss condition, the number of loss blocks to be piggy-backed by an ACK is also decided. Unlike TCP SACK, SNACK has the following virtues:

- SNACK can provide more effective explicit multiple packet loss information.

- SNACK utilizes parts of the TCP options in a small number of ACKs and is conveyed only between BS and MH, thus incurring only a small transmission cost.
- According to the type of wireless link in heterogeneous networks, the SNACK mechanism can be embedded in other protocols and have its parameters set flexibly.
- SNACK does not require any modifications to the protocol stack on FH.



(a) SACK (b) SNACK

Figure 4. The block structures

B. *SNACK-SN*

In this section, we propose two protocol components, SNACK-Snoop and SNACK-TCP, to implement SNACK-NS. From FH to MH, SNACK-Snoop performs the functions of detecting wireless multiple losses and piggy-backing the SNACK information, while SNACK-TCP performs the function of processing the ACKs with SNACK information and rapidly retransmitting the losses. From MH to FH, SNACK-Snoop performs the functions performed by SNACK-TCP in the direction from FH to MH, but SNACK-TCP performs the function performed by SNACK-Snoop in the direction from MH to FH. In addition, SNACK-NS follows the local timeout mechanism used in Snoop for fast retransmissions if needed. The following is the detailed implementation of the two protocol components in both directions. The network topologies are illustrated in Figure 1.

1) Transmissions from MH to FH

For the transmissions in this direction, SNACK-NS discards the ELN mechanism used in Snoop, and also introduces the fast retransmission mechanism over wireless links to enhance TCP performance.

SNACK-Snoop does not require storing the arriving packets and retransmitting any lost packets because wired networks provide reliable transmission. It only stores the sequence numbers of the received packets in a list to determine the sort of losses. For example, a hole between consecutive packets, which persists after several packets have arrived, will be regarded as a wireless loss. Whereas, if the sequence number of a lost packet indicated by some duplicate ACKs (Several duplicate ACKs whose sequence numbers are n indicate that the packet whose sequence number is n+1 has been lost.) is identical with one of sequence numbers stored in the above list, it shows that the packet has been transmitted to BS successfully but lost in the later transmission between BS and FH, so the loss will be regarded as a wired congestion loss. However, if the sequence number of a lost packet indicated by duplicate ACKs is not identical with any one of the sequence numbers in the list, the loss should be regarded as a wireless loss. When an ACK arrives at the BS, whether a new or a duplicate one, by using the above rules, if SNACK-Snoop detects some wireless losses, then it will piggy-back all their sequence numbers to the ACK as primary multiple wireless loss information. In existing TCP versions, wireless loss information is piggy-backed mainly by the duplicate ACKs produced by the arrival of packets behind the lost ones, but in SNACK-Snoop, loss information is piggy-backed by not only duplicate ACKs but also new ACKs, so long as when the ACKs arrive, there is detected loss information needed to be sent out. Thus, SNACK has responds faster to wireless multiple losses. Moreover, if SNACK-Snoop does not detect any wireless losses, the ACK will be transmitted to MH untouched. Therefore, apart from the recovery mechanism to wireless losses provided by SNACK-TCP, we can still utilize the congestion avoidance mechanism of TCP to handle wired losses, and thus recovery from network losses can be provided.

SNACK-TCP mainly performs the function of retransmission control, because the packet loss rate over wireless links is high. Minor modifications to TCP deployed on MH are required to take care of the explicit multiple wireless loss information piggy-backed in SNACK and to perform fast retransmissions of multiple wireless losses. Therefore, a structure list is used in SNACK-TCP. It consists of many structure cells and each structure cell is composed of the sequence number of a lost packet and the total times that MH has received the sequence information. When an ACK arrives in MH, according to information attached in the ACK, the list is updated, deleting the records of the packets that have been acknowledged. If it is a SNACK, then SNACK-TCP adds the records for the most recent lost packets or updates the numbers of times of notifications of lost packets. After updating, if the recorded times of any lost packet exceeds the retransmission threshold, the sender will retransmit the lost packet promptly without starting up TCP congestion control. After packet retransmission, the times record will be set to a negative value, say –1. If MH still receives loss information, the times record will remain negative. Generally speaking, by properly setting the value of the retransmission threshold, aggressive retransmissions can be avoided. In our simulation, we set the parameter as 2. Figure 5 shows the flowchart for ACK processing.

Figure 3 (d) analyzes the cooperation of the two protocol components to recover from the four continuous packet losses. When ACK 3 arrives at BS, SNACK-Snoop has detected all four packet losses, so it piggy-backs multiple loss information immediately on ACK 3, and then SNACK-TCP provides faster

retransmissions to all the four losses. It is obvious that the sender can recover from the busty losses within a shorter period.

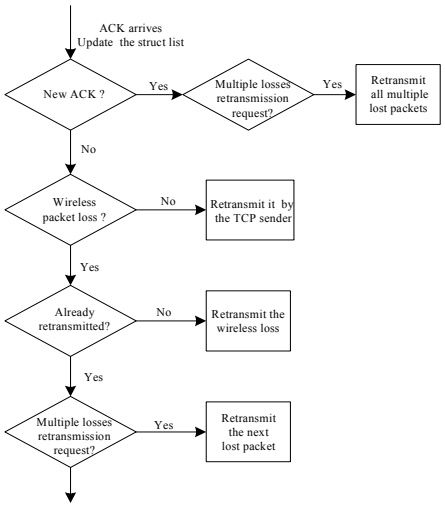


Figure 5. Flowchart for ACK processing

2) Transmissions from FH to MH

For data transmissions in this direction, SNACK-Snoop stores all packets received by BS. On the one hand, these packets are used to judge whether a loss is due to network congestion or wireless loss. On the other hand, they support fast local link layer retransmissions of the losses over wireless links. Besides, SNACK-Snoop performs the functions of processing ACKs with SNACK information and controlling fast retransmissions. However, TCP-MH performs the functions of detecting multiple packet losses and piggy-backing the SNACK information. Therefore, the two protocol components can also work well in the transmission from FH to MH. The implementation details of the two protocol components are the same as mentioned above in this section.

As shown in Figure 2 (d), when the ACKs with SNACK information arrive at BS, after updating the retransmission list, SNACK-Snoop immediately locally retransmits the multiple wireless losses and suppresses duplicate ACKs that may cause redundant transmissions. Thus, the conflict between TCP SACK and Snoop is avoided successfully and the performance of TCP over wireless networks is enhanced.

IV. NUMERICAL RESULTS

A. Simulation Topologies

All simulations in this paper were performed in Network Simulator (NS-2) [13]. Figure 1 shows the topologies of heterogeneous wired/wireless networks used. The system consists of a 10 Mbps, 10 ms propagation delay wired channel and a 2 Mbps wireless channel with a negligible propagation delay of 64 μs. The maximum congestion window size of the sender is 30 segments. The packet size is fixed at 1000 bytes.

It is well known that losses in wireless channels usually occur in a bursty fashion. These losses can be modeled as a two-state Markov error model consisting of a good state and a bad state as analyzed in [14]. In our simulations, we choose the two-state Markov error model used in [15] which have been adopted in several papers. In order to simulate a realistic network scenario with a bursty wireless link with different packet loss rates, we set the periods of the good and bad states as 6s and 0.2s respectively, fix the high packet loss rate in Bad state at 50% and vary the packet loss rate in Good state from 0.01% to 10%, then test the variation of the throughput of several different protocols.

B. Simulation Results

In the following, we show our simulation results for different transmission directions. Since the performance of TCP Tahoe, Reno and Newreno under the same network environment are similar, we only show TCP Reno results.

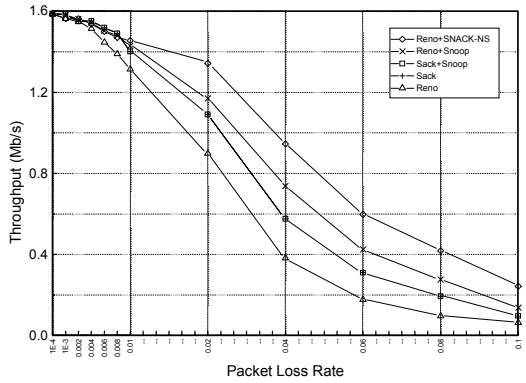


Figure 6. Transmission from MH to FH

Figure 6 shows the simulation results of the transmission in the direction from MH to FH over a 300 second period. Without Snoop, TCP SACK's performance is better than Reno, while, with Snoop, TCP SACK's performance is worse than Reno. In addition, the throughput of TCP SACK at each packet loss rate (PLR) is identical with that of TCP SACK with Snoop. These results prove the correctness of the analyses in Section II. Compared with Reno with the aid of SNACK-NS, however, when PLR in good state is varied from 0.01% to 1%, the throughput of Reno with SNACK-NS is close to the two TCP versions with Snoop, but has certain improvements over those without Snoop. When PLR is above 1%, with SNACK-NS, Reno has distinct improvements over the others. Compared with Reno with the aid of Snoop, Reno with SNACK-NS has an enhancement of 14.8% to 77.5%. Compared with TCP SACK with Snoop, the enhancement is around 23.2% to 152.7% and compared with Reno, the enhancement is around 49.5% to 286.9%.

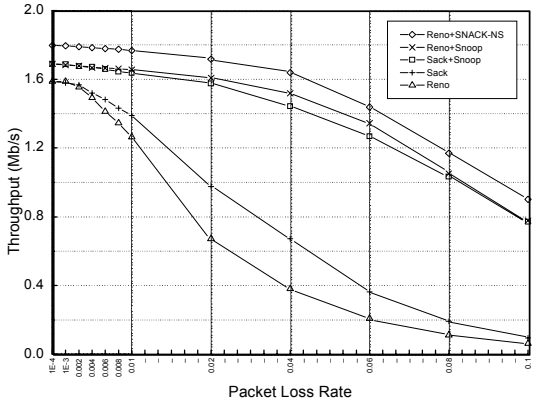


Figure 7. Transmission from FH to MH

Figure 7 shows the simulation results of the transmission in the direction from FH to MH, also in a 300 second period. Comparing the performance of the two TCP versions with and without Snoop, the results are similar to the above discussion. In the same way, the performance of Reno with SNACK-NS is superior to all of them. Deserving special attention is the fact that the performance of Reno with SNACK-NS has a steady improvement over Snoop for the range of loss probabilities under study. Even when the PLR is 0.01%, it has an improvement of 6.3% over Reno and TCP SACK with Snoop. With PLR at 10%, the enhancement is 16.0% over Reno with Snoop, 19.1% over TCP SACK with Snoop, 837.6% over TCP SACK and 1334.2% over Reno.

V. CONCLUTION

The enhancement of TCP over bursty wireless networks is an important research topic in the wireless Internet. This paper has proposed a novel protocol called SNACK-New Snoop to solve this problem. The key idea of the protocol is to introduce the capability to detect bursy losses at the base station (BS) and end mobile host (MH) in a wireless link, and to provide feedback to the source in a speedy manner to effect immediate retransmissions for packet lost in the wireless link. The SNACK mechanism can provide explicit wireless loss information between BS and MH with a small transmission cost. Through changing the functions deployed by the two protocol components, namely SNACK-Snoop and SNACK-TCP, both the MH to FH and FH to MH transmission performance can be greatly enhanced. Our analyses and simulation results show that SNACK-New Snoop can effectively enhance TCP performance over wireless links, particularly in those wireless networks with high packet loss rates and serious bursty losses. In the future, we will attempt to distinguish the wireless losses in details and study different response to them. In addition, we will consider different types of wireless links (e.g., WLAN, cellular networks, etc.) and test the performance of SNACK-New Snoop in such links.

REFERENCES

[1] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Transactions on Networking*, December 1997.

[2] C. P. Fu and S. C. Liew, "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks," *IEEE J. on Selected Areas in Commu*., vol. 21, no. 2, pp. 216-228, Feb. 2003.

[3] K. Brown and S. Singh, "M-TCP: TCP for Mobile Cellular Networks," Computer Communication Review, vol. 27,no.5, Oct. 1997.

[4] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving Reliable Transport and Handoff Performance over Cellular Wireless Networks," *ACM Wireless Networks* ,vol. 1, No. 4, December 1995.

[5] H. Balakrishnan, "Challenges to Reliable Data Transport over Heterogeneous Wireless Networks," Ph.D. thesis, UC Berkeley, May 1998.

[6] H. Balakrishnan and R. H. Katz, "Explicit Loss Notification and Wireless Web Performance," *IEEE CLOBLECOM*, Sydney, Australia, November 1998.

[7] G. Xylomenos and G. C. Polyzos, "Quality of Service Issues in Multi-service Wireless Internet Links," *the International Workshop on QoS in Multi-service IP Networks (QoS-IP)* , pp. 347–365. 2001.

[8] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment and Option," RFC 2801, IETF, October 1996.

[9] S. Floyd, "Issues of TCP with SACK," Technical report, Mar. 1996.

[10] W. Ding and A. Jamalipour, "A New Explicit Loss Notification with Acknowledgment for Wireless TCP," *PIMRC2001*, San Diego, CA, September 30-October 3, 2001.

[11] S. Vangala and M. A. Labrador, "Performance of TCP over wireless Networks with the Snoop Protocol," *LCN 2002. 27th Annual IEEE Conference*, 6-8 Nov. pp. 600-601. 2002.

[12] S. Vangala and M. Labrador, "The TCP SACK-Aware-Snoop Protocol for TCP over Wireless Networks," *IEEE VTC*, Orlando, October 2003.

[13] ns-2, http://www.isi.edu/nsnam/ns/.

[14] A. A. Abouzeid, S. Roy and M. Azizoglu, "Stochastic Modeling of TCP over Lossy Link," *IEEE INFORCOM 2000*, Tel Aviv, Israel, March 2000

[15] M. Gerla, M. Sanadidi, R. Wang, A. Zanella, C. Casetti, and S. Masco, "TCP Westwood: Window Control Using Bandwidth Estimation," *IEEE GLOBECOM*, San Antonio, Texas, USA, November 25-29, 2001.