A New Method to Improve the Performance of TCP SACK over Wireless Links

Jeng-Ji Huang
Department of Electrical Engineering
National Taiwan University
Taipei 106, Taiwan
E-mail: hjj@santos.ee.ntu.edu.tw

Jin-Fu Chang
Department of Electrical Engineering
National Chi Nan University
Puli, Nantou 545, Taiwan
E-mail: jfchang@ncnu.edu.tw

Abstract— **In this paper, a new method to improve TCP SACK's performance over wireless links is presented. The proposed method attempts to differentiate congestion and corruption loss by the use of tagged segments. Whenever a congestion loss occurs at a router, a tagged segment is dropped at the router and a notification is generated. Loss of the tagged segment is eventually detected by the TCP to serve an indication of congestion. When the tagged segment is not lost due to congestion, its survival then further serves as a trial signal for the TCP to detect corruption loss. Contrasting to most existing approaches employing explicit notification, our method is attractive in that it requires no modification to the receiving end. Simulation results show that our method significantly improves the performance of TCP SACK over both low and long delay noisy links.**

I. Introduction

Extending the TCP (Transmission Control Protocol) into the wireless segments has attracted enormous research attention in recent years [1]–[18]. One major effort treats the performance degradation problem of wireless TCP which is caused mainly by TCP's unable to discern between network congestion and (wireless) link corruption losses, and the predominant assumption that TCP segment loss is a sign of network congestion.

Among the many approaches to cure this problem, one category falls into providing a robust wireless link using more powerful link-layer protocols, e.g., FEC (Forward Error Correction) or ARQ (Automatic Repeat reQuest) [1], [2], or using a performance enhancing proxy (PEP) at the boundary of the Internet, e.g., the snoop protocol [3] or the split-connection protocol [4]. Another category is to change TCP to let it become capable of distinguishing these two kinds of loss [7]–[11]. In this category, side information must be supplied and should be firm and clear enough to let a sender correctly tell if a loss is due to corruption and to avoid triggering unnecessary congestion avoidance mechanism. But, if side information has ambiguity, the sender should be rather conservative in preventing network from congestion collapse [19].

There have been several previous works [9]–[11] on providing clear side information. Their design is to incorporate an explicit notification signal into the congestion detection mechanism. However, most of them require modifications at both the sender and receiver TCP, and at routers or base stations as well. Other approaches, e.g., [5]–[7], although affect only TCP implementation at the sender, the information provided is less clear.

In this paper, a new method to differentiate these two types of segment loss is proposed. When a router is in congestion, a notification is generated by forcefully dropping a pretagged segment. Based on whether a congestion notification is detected, the TCP is able to differentiate congestion losses from corruption losses. The proposed congestion notification involves changes at the sender and routers only, no modification is required at the receiving end.

The rest of this paper is organized as follows. Section II gives relevant backgrounds, including related works that use side information for differentiation, and an overview of TCP SACK (TCP with Selective Acknowledgement). We introduce the proposed method in Section III, and compare its performance with TCP SACK through simulations in Section IV. Finally, concluding remarks are given in Section V.

II. Backgrounds

A. Related Side-Information Based Works

Side information appears in one of the following forms.

- RTT (round trip time) [5], [6]: The sender deduces that a router may currently encounter congestion from the observation of an increasing RTT. Segment losses are then assumed to be due to congestion in this case.
- ECN (explicit congestion notification) [20]: For a TCP connection that is ECN-capable, if a sender receives three duplicate ACKs with the CE (Congestion Experienced) bit set, the sender infers that a segment loss due to congestion has occurred [7].
- ELN (explicit loss notification) [8]: An ELN bit in a new form of acknowledgement called ACK_{ELN} is judged at the base station, and is used to explicitly tell that a segment has been lost before it comes to the base station.
- Lhack (last hop acknowledgement) [9]: A lhack is returned to the source for every message received at the

The work reported in this paper is supported by the National Science Council, Taiwan (NSC 91-2219-E-260-001).

0-7803-7757-5/03/$17.00 ©2003 IEEE.

base station, and is used to indicate to the source that the corresponding message must have been lost due to corruption if the sender does not receive an ACK from the receiving end for this message.

- W-ECN (wireless-ECN) [10]: The W-ECN is used as a prompt notification to retransmit lost segments whenever congestion loss occurs. Thus, it can be used as an indicative signal of congestion.
- HACK (HeAder CheckSum) [11]: The HACK proposes to add a separate checksum for the header portion of a TCP segment so that TCP receiver is able to check the integrity of header. The receiver is able to return a signal ACK back to the sender indicating corruption.

In comparison, RTT or ECN provides less clear information than the other four. But, implementation of ELN, W-ECN, or HACK requires changes at both sender and receiver, and at routers or base stations as well; while implementation of RTT or ECN affects the software at sender only. Lhack though is able to provide a clear signal and requires no change at the receiver, the signal is subject to loss as it travels back to the sender.

B. TCP SACK

The SACK option of TCP has been standardized and widely deployed in the Internet [12], [21]. In TCP SACK, the SACK option should be included in duplicate ACKs [22]. When three consecutive duplicate ACKs are collected at the sender, fast recovery is triggered and transmit window is halved.

Using SACK, a sender can be informed of up to three noncontiguous blocks of data that have been received and queued at the receiver. With this information, the sender is able to retransmit the segments that comprise the holes in the sequence space, whenever the number of flying segments in the pipe is less than the congestion window. If there are no holes, new data is sent. When an ACK acknowledges everything sent before fast recovery was entered, the TCP leaves fast recovery.

Due to the ability to recover from multiple lost segments in one RTT, it has been shown that TCP SACK provides a better performance than other versions of TCP [4], [14]. In addition, experiment results also show that TCP SACK is more efficient on noisy large BDP (bandwidth-delay-product) connections [15].

III. The Proposed Method

The proposed method is a variant of TCP SACK. It attempts to differentiate losses by the use of tagged segments. Differentiation is facilitated by a new congestion detection mechanism, whereas segment losses are no longer directly interpreted as congestion warning unless a tagged segment is declared lost.

A. Tagged Segments

Tagged segments are equivalent to normal TCP segments except that the priority bit in their IP header is set low. Like normal segments, payload information is also carried by tagged segments; thus, they must be recovered if they are lost. To TCP receiver, a tagged segment looks exactly the same as a normal segment, since its IP header would have been removed.

Use of low-priority segments can also be found in [13], [16], and [23]. They are used as probing packets to measure the available bandwidth in the network [13], [23], or are used to carry information to the receiver more rapidly without harming other flows [16]. In this paper, tagged (low-priority) segments are used to provide the sender with the information of network congestion.

In the proposed method, one segment is tagged at the source per every window worth of segments. When an ACK returns to acknowledge successful reception of the previously tagged segment, a next segment is tagged, even when the TCP is undergoing a fast recovery.

B. Dropping Policy at Routers

The proposed method requires the support of a new dropping policy at routers. Under this new dropping policy, whenever congestion loss occurs at a router, a notification is generated by forcefully dropping one tagged segment at the router. If no such tagged segment is available, the next arriving tagged segment will be dropped, even though the router may become not congested as it arrives. Dropping a tagged segment serves as a notification to cover congestion losses that have occurred since the drop of the last tagged segment. Loss of a tagged segment will be eventually detected at the sender via SACKs to serve as a signal indicative of congestion.

It should be noted that a tagged segment may be forcefully dropped when the buffer is actually not full. The impact of such extra drop can be either of the following two. One is that this extra drop may lead the TCP to reenter fast recovery. We shall later in Sec. III.D.1 illustrate that fast recovery reentering is vital for the proposed method. The other is that TCP may take more time to recover losses. It is because that, although TCP SACK is able to retransmit multiple losses in one RTT, it in no way says that no matter how many segments are lost all can be retransmitted in one RTT. Thus, the TCP may take an additional RTT to continue to recover losses due to the extra drop [24]. We have performed simulations to examine its impact on performance in Sec. IV.B; as will be seen, degradation turns out to be very light.

C. Differentiation Rule

In the proposed method, differentiating losses between congestion and corruption is performed at the end of fast recovery according to whether a tagged segment has been lost in the window upon entering fast recovery. Under our new dropping policy, **if loss of a tagged segment is found there exists at least one segment loss due to congestion; otherwise, all losses are due to corruption.** When congestion loss is detected, the TCP

must slow down its sending rate, even if there is only one segment loss that is due to congestion. On the contrary, when losses are concluded as due to corruption, the transmit window is restored to the value before fast recovery was entered.

Unfortunately, loss of a tagged segment may be due to not only congestion at a router but also corruption on the wireless link. Differentiation is conservative by assuming that a congestion event has occurred in this case. However, when a tagged segment is not lost, the survival would provide TCP with a strong indication that no congestion has occurred on the path; thus, all segment losses can be concluded as due to corruption.

Due to that loss of a tagged segment is used as a congestion notification, our method involves changes at the sender and routers only. No modification of software is required at the receiver. Furthermore, our congestion notification mechanism is robust, since the signal provided is obviously free from any further damage as it loops back to the sender.

D. Reliability of the Proposed Method

The proposed method poses two main threats to reliability. One is that delayed congestion notification may result from delayed drop of a tagged segment. Another is that route may change.

D.1 Delayed Congestion Notification

Under the new dropping policy, as mentioned in Sec. III.B, whenever a congestion loss occurs at a router, one tagged segment needs to be dropped in order to generate a notification. If none is available, the next arriving tagged segment is dropped. This results in the following two scenarios.

1. The tagged segment to be dropped is sent before TCP's entering fast recovery. In this case, the segment would have appeared in the window when fast recovery is entered. Loss of this tagged segment shall be detected at the end of fast recovery, and used to notify of network congestion.
2. The tagged segment to be dropped is a new data segment that is sent during fast recovery. This occurs when there are no holes in the window and the number of segments in the pipe is less than the congestion window, as mentioned in Sec. II.B. If all losses in the window upon entering fast recovery have been recovered, TCP will leave the current fast recovery. In this case, loss of the tagged segment shall be detected in the next fast recovery. According to our differentiation rule, a sender does not reduce its rate until a congestion is notified. Thus, sender's response to a congestion event has to be delayed till when loss of the tagged segment is detected at the end of a later fast recovery. The delay would amount to about one RTT, i.e., the duration of a fast recovery.

To solve the above problem, one possible solution is to inject more tagged segments into every window. This would force the first scenario to occur more often, and would thus lessen the occurrence of the second. However, the increase would raise the ambiguity or vagueness of our proposed signal, due to increasing chance that tagged segments are rather corrupted by channel errors. This would as a result reduce the degree of improvement offered by our method.

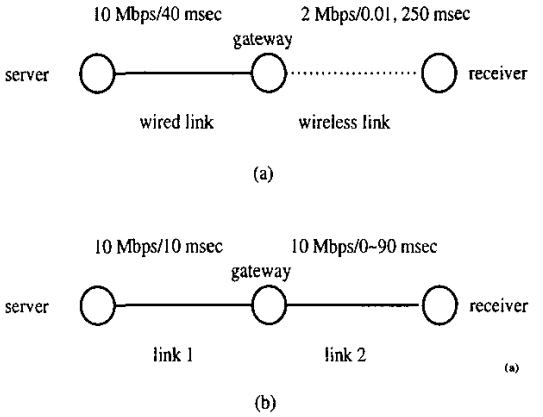


Fig. 1. The topology in the simulations of (a) a TCP connection over a wireless link and (b) a wired connection.

D.2 Route Changes

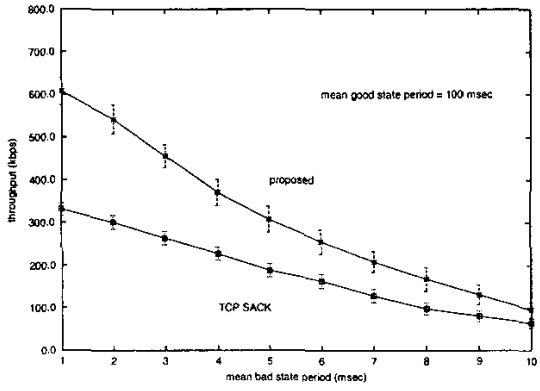
In practice, the route of a TCP connection may undergo changes during its lifetime, although route changes are normally infrequent [25]. If the route has changed while a congestion event occurs in the old path, congestion notification may not be generated by the new path. Without congestion notification, the sender does not reduce its transfer rate. However, this does no harm to the network since sender has no need to reduce its rate if no congestion develops along the new path.

IV. Simulation Results and Discussions

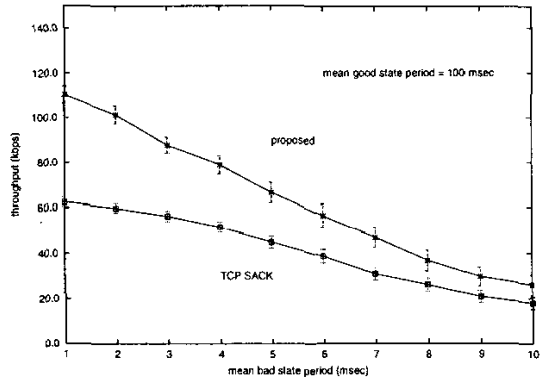
In this section, throughput performances of the proposed method and TCP SACK are compared via simulations. The simulations are conducted using the TCP ns-2 simulator of LBNL (Lawrence Berkeley National Laboratory) [26]. The traffic is assumed to be FTP transfers. The size of each transferred file is 1 Mbytes, and the length of a segment is 1,000 bytes.

A. TCP Connections over Wireless links

In this section, performance of the proposed method is examined over a wireless link. The topology in the simulations is depicted in Fig. 1 (a). It consists of a server, a gateway, and a receiver. The wired link between the server and the gateway can be viewed as a path through routers in the Internet. Its bandwidth is assumed to be 10 Mbps [6], [17], and its delay is assumed to be 40 msec [14], [17]. The bandwidth of the wireless link between the



(a)



(b)

Fig. 2. Comparison of throughput performance over (a) a low delay noisy link and (b) a long delay noisy link.

gateway and the receiver is 2 Mbps [7], [17]. But its delay is taken to be either 0.01 [6], [7] or 250 msec. The former may correspond to a low delay link such as a wireless LAN or cellular system; while the latter may represent a long delay link such as a satellite link.

Correlated channel errors are applied to the wireless link, and are modeled as a two-state, a good state and a bad state, Markov process [18]. The residual time in either state is exponentially distributed. The mean good state period is fixed at 100 msec; while the mean bad state period is varied from 1 to 10 msec to reflect various channel conditions. Segment transmissions are assumed to be successful in a good state and corrupted in a bad state, with probability both equal to one. After 100 independent runs, the results are used to calculate the mean TCP throughput with a 95% confidence interval.

A.1 Low Delay Noisy Links

In this section, the throughput performance of the proposed method is compared with TCP SACK over low delay noisy links. In Fig. 2 (a), our method significantly outperforms TCP SACK, especially for short mean bad state period. The throughput is improved by 83% for mean bad period equal to 1 msec, 63% for 5 msec, and 47% for 10 msec. The substantial improvement is mainly due to the fact that the proposed method avoids unnecessary activation of congestion avoidance mechanism, when segment losses are due to corruption and when they are correctly identified.

The advantage of our method becomes less substantial for long bad state. It is because that burst length is proportional to the duration of the bad state. As burst length increases, the chance of a tagged segment being hit by channel errors increases. Once a tagged segment is lost, even though the loss may be due to corruption, congestion is assumed according to our differentiation rule. This results in unnecessary window reduction, a consequence equivalent to what the original TCP SACK would have; thus, no improvement is seen in this case.

It is clear that improvement offered by our method relies heavily on the survival of tagged segments when channel errors occur on wireless links. In our proposed method, if more protection can be placed on tagged segments at the link level, further improvement is anticipated. Ideally, if tagged segments can be completely shielded from channel errors, the proposed method should perform exactly the same as those employing explicit notification.

A.2 Long Delay Noisy Links

Channel errors are shown to have more severe effect on TCP performance over long delay links [13]–[15]. It is due to that slow growing of window size after the congestion window is halved in reaction to a segment loss.

In this section, the performance improvement achieved using our method over long delay noisy links is examined. As shown in Fig. 2 (b), our method still performs better than TCP SACK by 77% for mean bad state period equal to 1 msec, 48% for 5 msec, and 44% for 10 msec. Compare with the previous section, as expected both methods drop dramatically in throughput over long delay noisy links by about 80%. However, loss in improvement of our method over TCP SACK is light, by less than an average of 10%.

B. TCP Connections over Wireline Links

In this section, performance of our method is examined on a connection that consists of wireline links only. The topology in the simulations is depicted in Fig. 1 (b). It consists of a server, a router, and a receiver. The link '1' between the server and the router and the link '2' between the router and the receiver are both wireline of 10 Mbps. The delay of link 1 is fixed at 10 msec; while the delay of

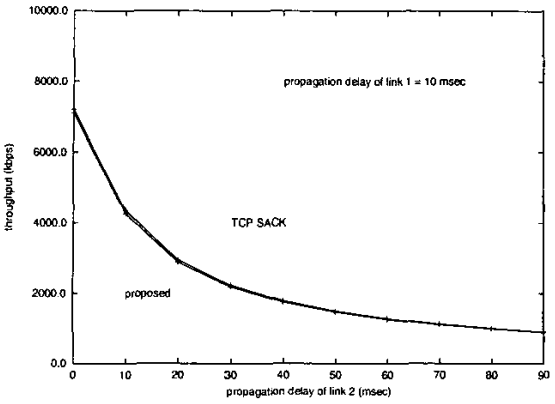


Fig. 3. Comparison of throughput performance on a wired connection.

link 2 is varied from 0 to 90 msec to reflect various RTT values of a TCP connection.

As shown in Fig. 3, performance of the proposed method is almost equivalent to that of TCP SACK on a wired connection, with a very slight average degradation of only 2.2%. In the simulations, performance loss of the proposed method is mainly due to the additional time taken for TCP to recover a possible extra drop (of a tagged segment) as discussed in Sec. III.B.

V. Conclusions

In this paper, we have proposed a method to improve the performance of TCP SACK over wireless links. The proposed method is made possible by the use of tagged segments accompanied by a new dropping policy. In the proposed method, a sender is able to discern congestion from corruption in large likelihood. In contrast to most other existing approaches employing explicit notification, our method is attractive in that it requires no modification at the receiver. However, a support is required at the routers to generate congestion notification signals by dropping tagged segments. Through simulations we have compared the performance of our proposed method with TCP SACK over both low and long delay noisy links. The results show that our method significantly improves TCP SACK over a wide range of channel conditions in both environments.

References

[1] G. Huston, "TCP in a Wireless Word," *IEEE Internet Computing*, pp. 82–84, March/April 2001.

[2] E Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin, "AIRMAIL: A Link-layer Protocol for Wireless Networks," *ACM/Baltzer Wireless Networks*, vol. 1, pp. 47–60, Feb. 1995.

[3] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks," *Proc. 1st ACM Conf. on Mobile Computing and Networking*, pp. 2–11, Nov. 1995.

[4] C. Partridge, and T. J. Shepard, "TCP/IP Performance over Satellite Links," *IEEE Network*, pp. 44–49, Sep./Oct. 1997.

[5] N. K. G. Samaraweera, "Non-congestion Packet Loss Detection for TCP Error Recovery Using Wireless Links," *IEE Proc.-Commun.*, vol. 146, no. 4, pp. 222–230, Aug. 1999.

[6] C. Parsa and J. J. Garcia-Luna-Aceves, "Differentiating Congestion vs. Random Loss: A Method for Improving TCP Performance over Wireless Links," *IEEE WCNC 2000*, vol. 1, pp. 90–93, 2000.

[7] R. Ramani and A. Karandikar, "Explicit Congestion Notification (ECN) in TCP over Wireless Network," *IEEE International Conference on Personal Wireless Communications 2000*, pp. 495–499, 2000.

[8] W. Ding and A. Jamalipour, "A New Explicit Loss Notification with Acknowledgement for Wireless TCP," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2001*, vol. 1, pp. 65–69, Sept. 2001.

[9] J. A. Cobb and P. Agrawal, "Congestion or Corruption? A Strategy for Efficient Wireless TCP Sessions," *IEEE Symposium on Computers and Communications*, pp. 262–268, 1995.

[10] F. Peng, S. Cheng, and J. Ma, "An Effective Way to Improve TCP Performance in Wireless/Mobile Networks," *IEEE/AFCEA EUROCOMM 2000*, pp. 250–255, 2000.

[11] R. K. Balan, B. P. Lee, K. R. R. Kumar, L. Jacob, W. K. G. Seah, and A. L. Ananda, "TCP HACK: TCP Header Checksum Option to Improve Performance over Lossy Links," *IEEE Infocom 2001*, pp. 309–318, 2001.

[12] S. Floyd, "A Report on Recent Developments in TCP Congestion Control," *IEEE Communications Magazine*, pp. 84–90, April 2001.

[13] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Trans. on Networking*, vol. 9, No. 3, pp. 307–321, June 2001.

[14] I. Minei and R. Cohen, "High-Speed Internet Access Through Unidirectional Geostationary Satellite Channels," *IEEE J. Select. Areas Commun.*, vol. 17, no. 2, pp. 345–359, Feb. 1999.

[15] C. P. Charalambos, V. S. Frost, and J. B. Evans, "Performance of TCP Extensions on Noisy High BDP Networks," *IEEE Communications Letters*, vol. 3, no. 10, pp. 294–296, 1999.

[16] V. N. Padmanabhan and R. H. Katz, "TCP Fast Start: A Technique for Speeding up Web Transfers," *Proc. IEEE Globecom Internet Mini-Conference*, Nov. 1998.

[17] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," *Proceedings of ACM Mobicom 2001*, pp. 287–297, July, 2001.

[18] M. Zorzi, A. Chockalingam, and R. R. Rao, "Throughput Analysis of TCP on Channels with Memory," *IEEE J. Select. Areas Commun.*, vol. 18, No. 7, pp. 1289–1300, July 2000.

[19] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. on Networking*, vol. 7, no. 4, pp. 458–472, August, 1999.

[20] K. Ramakrishnan and S. Floyd, "RFC 2481: A Proposal to Add Explicit Congestion Notification (ECN) to IP," Jun. 1999.

[21] M. Allman, "A Web Server's View of the Transport Layer," *ACM Comput. Commun. Rev.*, vol. 30, no. 5, pp. 10–20, Oct. 2000.

[22] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "RFC 2018: TCP Selective Acknowledgement Options," Oct. 1996.

[23] G. Bianchi, A. Capone, and C. Petrioli, "Throughput Analysis of End-to-End Measurement-Based Admission Control in IP," *Proc. IEEE Infocom*, vol. 3, pp. 1461–1470, March 2000.

[24] R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy, B. Vandalore, and X. Cai, "Selective Acknowledgements and UBR+ Drop Policies to Improve TCP/UBR Performance over Terrestrial and Satellite Networks," *ATM Forum/97-0423*, April 1997.

[25] V. Paxson, "End-to-end Internet Packet Dynamics," Proceedings of SIGCOMM'97, pp. 139–152, 1997.

[26] S. McCanne and S. Floyd, "Ns (network simulator)," [online]. Available WWW: http://www-nrg.ee.lbl.gov/ns.