

Congestion or Corruption? A Strategy for Efficient Wireless TCP Sessions

Jorge A. Cobb

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188

Prathima Agrawal

AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974

Abstract

We present a new acknowledgment strategy to improve the performance of TCP sessions that originate or terminate in noisy wireless networks for mobile computers. This acknowledgment strategy allows the TCP source to distinguish between losses due to congestion and losses due to corruption. With this distinction, the source can reduce its sending rate when congestion occurs, and quickly retransmit when corruption occurs. Without this distinction, TCP throughput is shown to suffer significantly over a path with a large bandwidth-delay product. The strategy is also appropriate for dealing with losses due to hand-offs of a mobile computer from one wireless cell to another.

1. Introduction

Recent advances in hardware and communication technology have made possible the birth of mobile computing. Advances in radio and infra-red technology, in combination with powerful portable computers, will provide users with access to a network at all times. This continuous connectivity will allow users to be quickly notified of changing events, and provide them with the necessary resources to respond to these events, even while in transit. In the near future, it is expected that millions of users will carry a portable computer capable of accessing the worldwide Internet [BIV92]. The possibility of mobile computing and its expected popularity has generated a great deal of research, due to its ramifications in networking and distributed data management [IB94] [FZ94] [PB93].

The expected demand for mobile computing has already prompted research on updating the Internet's network layer protocol (IP) [Pos81] to incorporate mobility. These proposals range from supporting mobility at the campus level [IDM91] [CPR92], to allowing computers to move to any network in the Internet [CEG94] [IP94] [Per93] [TKT91]. The goal of these schemes is to change the addressing and

routing structure of IP so that mobile computers can effortlessly join the Internet at any remote location. With such schemes, users can make full use of their Internet applications while away from their offices.

Although much research exists on the effects of mobility on the network layer, little work exists on its effects on transport protocols, such as TCP [Com91]. Due to the arrival of new user-friendly applications for information distribution over the Internet, TCP traffic has increased significantly, and it is reasonable to assume that most mobile computer traffic will be TCP sessions. Thus, we investigate the effects of wireless computing on TCP performance.

Wireless media have different properties from those upon which the design of TCP is based. In particular, TCP assumes that network links rarely corrupt messages [Jac88], because the loss of a message is taken as a signal for congestion in the network. Thus, upon a message loss, TCP reduces its sending rate to alleviate congestion. This approach is effective if corruption losses are rare. However, wireless media have higher bit error rates [GP95], causing significant corruption losses. This will cause TCP to needlessly reduce its sending rate, yielding poor performance.

In this paper, we present an acknowledgment scheme that allows the transport protocol to distinguish between congestion and corruption losses. This is accomplished by adding a small amount of functionality to the routers of the wireless networks, while leaving all other routers unchanged. We show how to incorporate this scheme into the TCP protocol, and demonstrate its effectiveness through simulation.

The paper is organized as follows. The assumptions on the internetwork structure are presented in Section 2. In Section 3, we overview the congestion control and avoidance strategy of TCP and its vulnerability to corruption losses. The new acknowledgment strategy is presented in Section 4, along with how to incorporate it into the existing TCP protocol. The simulation model and results are presented in Section 5. A summary and concluding remarks are given in Section 6.

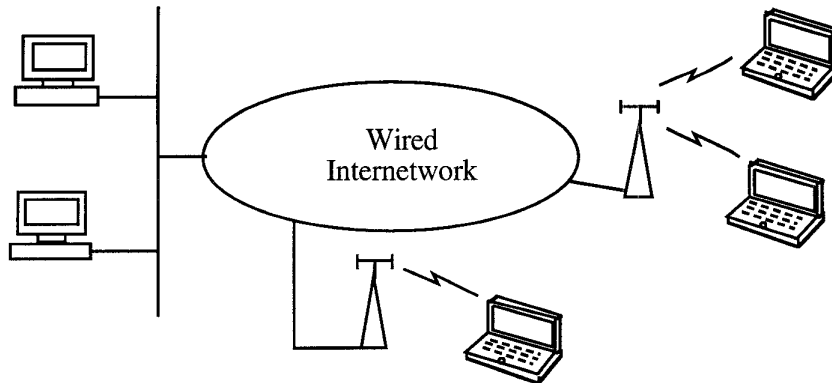


Figure 1: Internetwork model

2. Network Model

The internetwork model we assume consists of multiple networks joined together by computers attached to more than one network, also known as routers. To deliver a message from a computer in one network to a computer in another network, the message is forwarded from one intermediate router to another until the final network is reached.

As shown in Figure 1, the internetwork contains a group of wired networks, such as Ethernet and Token Ring, that form the core of the internetwork. These networks are assumed to be relatively corruption free, causing very few messages to be corrupted. Thus, it is reasonable to assume that if a message is lost in these networks it is due to buffer overflows, i.e., congestion. The less reliable wireless networks are assumed to be at the edges of the internetwork, and they forward messages to and from mobile computers. Wireless networks are not allowed to serve as intermediate hops in the routing path between two networks. Any pair of computers in the internetwork can establish a TCP session between them, regardless of whether they are mobile (wireless) or stationary (wired). Computers involved in a TCP session are also known as hosts.

The geographical region covered by a single wireless network is called a cell. Multiple mobile hosts may be operating on a single cell by sharing the medium with multiplexing techniques such as CSMA. Mobile hosts are free to move from one cell to another. After a mobile host moves to a different cell, it must notify its correspondent host of its new location, and inform the router of the previous cell to forward messages addressed to it to the new cell. This notification process is called a hand-off. During a hand-off, the communication between the mobile and its correspondent host may be temporarily interrupted if the coverage of the old cell and the new cell do not overlap sufficiently. We assume that this rarely occurs, and concern ourselves mainly

with losses due to corruption. We address the issue of losses due to hand-offs in Section 6.

We assume that routers provide only datagram delivery, i.e., they make an effort to deliver a message, but do not guarantee its delivery. In particular, routers do not ensure that a message is received by the next computer in the path to the destination. Routers forward each message to the next computer in the path, but this computer may drop the message due to corruption or buffer overflow.

Routers attached to a wireless network are aware that their network is prone to corruption and is located at the edge of the internetwork. Routers in the wired network are not aware of mobile hosts, and must not be affected by any changes to support wireless communication. We assume that enhancing the functionality of wireless routers to a small degree is allowed. However, requiring wireless routers to reliably deliver messages is disallowed, due to the overhead of maintaining timers and performing retransmissions.

3. TCP Congestion Control

In this section, we give an overview of the flow control and congestion control mechanisms of TCP [Jac88], and explain its sensitivity to corruption losses. In the next section, we present our acknowledgment scheme as an approach to remedy this problem.

TCP is a reliable transport protocol for the delivery of a sequence of messages over an unreliable datagram network. It uses a sliding window form of flow control. That is, the source assigns a sequence number to each sent message, and expects an acknowledgment from the destination for each of these messages. The source ensures that no more than W messages are outstanding without being acknowledged, where W is known as the window size. The destination uses cumulative acknowledgments, i.e., an acknowledgment for message i implicitly acknowledges all messages from 0 up to i . Thus, if i is the last acknowledged

message, the source may send messages $i + 1$ through $i + W$.

Receiving acknowledgments is used as a self-clocking mechanism for the source. Assume the destination returns an acknowledgment for every message received. When the source receives an acknowledgment, it indicates that a message has been removed from the network, implying that one more message can be sent without causing congestion.

Changing the window size is used as a form of rate control. Since the maximum number of outstanding messages is W , the maximum possible sending rate is W/D , where D is the round-trip propagation delay to the destination. If W/D is smaller than the available network bandwidth, the source is not maximizing its throughput, and if it is larger, it will cause queuing at the intermediate routers, leading to congestion.

At the start of a TCP session, the appropriate sending rate is unknown, so an upper bound on the window size is chosen arbitrarily. The window size is set to 1, and it grows exponentially with each round-trip time until the limit is reached. This exponential growth is accomplished by increasing the size of the window by one with each acknowledgment. This effectively duplicates the window with each round-trip time. This is known as the slow-start phase, and the upper bound on the window size is called the slow-start threshold¹. Once the window reaches this threshold, it begins a congestion avoidance phase, in which the window size is increased slowly, adding one every round-trip time. That is, if the window size is W , then, after receiving W acknowledgments, the window size is increased by one.

An estimate is maintained of the round-trip time of messages, and a single timer is kept with the sending time of the last message plus its expected round-trip time. If a timeout occurs, it is assumed that the window is so large that it has caused congestion in the network, leading to message loss. Thus, the window is reduced as follows. The slow-start threshold is set to half the current window size, and the window is set to one. The window is increased using the slow-start scheme until it reaches the slow-start threshold. Then, the congestion avoidance phase is resumed.

If consecutive timeouts occur, the timeout value is increased exponentially with each timeout. Also, the measured round-trip time of a retransmitted message is not used for the overall estimate of the round-trip time, since it is unknown if the acknowledgment is for the first or second copy of the message sent [KP91].

If a message is lost, the source will receive multiple acknowledgments for the sequence number prior to the lost message. This is because the destination returns an acknowledgment for every message received, and since the ac-

knowledgments are cumulative, it continues to send the same acknowledgment until the lost message is received. Thus, when several duplicate acknowledgments are received, it is assumed that a loss occurred, so the current window is retransmitted. This is known as a fast-retransmit. In an earlier version of TCP, TCP Tahoe, the same window adjustments that are done for a timeout are also performed for a fast-retransmit. In this paper, we use a newer version, TCP Reno, in which the window size is reduced by a half, but no slow-start is performed. Instead, a count is kept of the duplicate acknowledgments received. This gives the source an indication of how many messages have been removed from the network, and thus also indicates how many messages remain in the network. When the latter reaches the new window size, the source begins to retransmit the lost messages.

Notice that if a message is lost due to corruption, the source will either timeout or fast-retransmit, decreasing the window by half. Since the source's maximum rate is limited by the window size, a corruption loss will in effect reduce the sending rate by half. If corruption losses occur often, the window size will never reach its optimum size, leading to significant throughput losses. This is particularly true when the bandwidth-delay product of the path is large, which requires a large window for optimum throughput.

4. Last Hop Acknowledgment

One way to eliminate corruption losses is for the mobile host to establish a connection with its adjacent router in the wireless network. The router could then ensure that all messages it sends to the mobile are eventually received. However, this requires the router to maintain a connection table and the overhead associated with it, such as keep-alive messages, maintaining timers, and retransmitting unacknowledged messages. Furthermore, when the mobile moves to a new cell, there is the additional overhead of dissolving the connection with the old router and establishing a new one with the new router. Finally, the wireless cell may have more than one router, and messages for the same TCP session may arrive over different routers², requiring the mobile to establish a connection with each of these routers. Instead, we would like a scheme which increases the throughput of TCP over wireless networks without forcing a significant overhead on the wireless routers.

Consider the case when a stationary host is the source and a mobile host is the destination. To determine that a message traversed the wired network and arrived to the router of the wireless cell, we suggest that this router return

¹ Slow-start is a misnomer, due to quick exponential growth of the window size.

² Some routing algorithms, for load-balancing purposes, distribute the messages equally over multiple equi-distant paths to the destination.

a last hop acknowledgment (henceforth, lhack) to the source for every message it receives. The lhack indicates to the source that, if it does not receive an acknowledgment from the destination (henceforth, dack) for this message, the message must have been lost due to corruption. This is because the message arrived to the final hop of the path to the destination, so its loss occurred in the wireless network. Thus, no congestion occurred, and the source should not reduce its window size. However, if neither a dack nor lhack is received for a message, then the message was probably lost due to congestion at an intermediate node. Hence, the source must reduce its window size in this case.

Notice that dacks may also get corrupted by the wireless network. Since the router does not perform retransmissions, dacks are not guaranteed to reach the source. However, since dacks are cumulative, the loss of a single or even a few consecutive dacks will affect the source only slightly, since the next dack received will allow the window to slide and new data messages to be sent.

Next, consider a mobile host sending data to a stationary host. A data message sent by the mobile can be corrupted by the wireless network. Thus, the router should send a first-hop acknowledgment (henceforth, fhack) to the mobile. If the mobile does not receive a fhack for the message, it is retransmitted. Thus, eventually, the data message is received by the router and sent through the internetwork.

The final case is a mobile to mobile connection. In this case, the source receives both an fhack and lhack for each data message. Note that it is possible that a data message is corrupted at the destination network, and the lhack for this message is corrupted at the source network. Since an fhack is received, but not an lhack, the source incorrectly assumes that congestion occurred and decreases its window size. We argue, however, that this is unlikely. If we assume that the probability of a loss at the source and destination networks are independent, the probability that both a message and its lhack are lost is small, most likely smaller than the probability of a corruption in a wired network. Thus, throughput should not be affected significantly. Also, the two mobiles may choose a stationary computer, such as the desk workstation of one of the users, as an intermediate system to which both mobiles establish a connection. This reduces the problem to the two cases above.

Sending an lhack for each data message has the side effect of doubling the amount of traffic from the destination to the source, since two types of acknowledgments are sent per message. However, acknowledgments are small (40 bytes) relative to data messages (500 bytes), so this increase is modest. Furthermore, it can be reduced, for example, by having the destination return an acknowledgment for every three data messages. Since dacks are cumulative, this has little, if any, effect on the throughput, and the acknowledgment traffic overhead is reduced to one third.

We next describe how to incorporate our acknowledgment scheme into TCP's congestion control. If no lhacks are received, either because the destination is stationary or the messages were lost due to congestion, the source should behave as in the standard TCP protocol. If lhacks are received, the source should maintain its current throughput when retransmitting messages lost due to corruption.

The source maintains a bit array indicating, for each sent message, if an lhack has been received. Upon a timeout, the source checks if an lhack was received for the timed-out message. If it was not, congestion is assumed, so the standard TCP steps are taken. Otherwise, the sending rate, i.e., the window size, should not be reduced. However, since TCP only maintains a single timer, then after a timeout, the network no longer has any messages from this source. Thus, the self-clocking property of sending one message for each received acknowledgment is lost, so a slow-start must be performed and the window set to 1. However, the slow-start threshold is set to the window size prior to the timeout, allowing the window to grow exponentially to its size prior to the timeout.

A similar decision is made for the case of a fast-retransmit. If an lhack was not received for the message, congestion is assumed, and the standard TCP steps are taken. Otherwise, the window size remains unchanged, because the loss is by corruption, and the sending rate need not decrease. Also, a similar technique to that of TCP reno is used to extend the window, except that the number of lhacks received is used, rather than the number of duplicate dacks.

5. Simulation Results

We next simulate TCP reno with and without our acknowledgment scheme, using the REAL simulator from [Kes94]. The scenario is shown in Figure 2, and consists of: a TCP source, TS, its destination, TD, four routers, R1 through R4, and cross traffic sources, C1 through C7. Cross traffic sources have a poisson distribution.

All links have a bandwidth of 100 Kbytes/sec. Links between routers have a propagation delay of 30 msec., and links between a router and a neighboring source or destination have a propagation delay of 5 msec. This yields a round-trip time of 200 msec., which is adequate if the destination is in another continent or if a satellite link is being used. Furthermore, the complex coding and decoding in wireless media adds about 100 msec. to the round-trip processing pipeline of messages [GP95]. Thus, an overall delay of 200 msec. is reasonable.

The length of data and acknowledgment messages are 500 bytes and 40 bytes, respectively. The bottleneck is link (R2, R3), where cross traffic consumes half the bandwidth. Thus, the maximum possible throughput for TS is 100 data messages per sec., and its optimum window size is 20 data

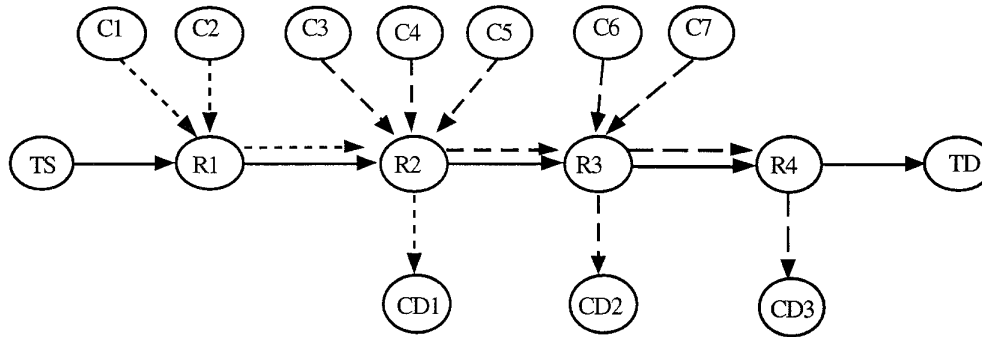


Figure 2: Simulation scenario

messages. The initial slow-start threshold is set to 20. Each router can buffer 30 data messages per link.

Links (TS, R1) and (R4, TD) corrupt messages according to some bit error probability (bep). For a wired link, the bep is zero. For a wireless link, the bep is varied from $3 \cdot 10^{-5}$ (one in every 75 data messages) down to $1 \cdot 10^{-6}$ (one in every 250 data messages).

Figure 3 shows the throughput of TCP both with and without the use of lhacks for the case of a wired source and a wireless destination. Because of cross traffic, the maximum throughput is limited to 100 msgs/sec. If lhacks are provided, the throughput reaches 96% of this maximum. Without lhacks, the throughput is reduced significantly. Even with a relatively mild loss probability of 1/150 messages, the throughput is less than 60% of the maximum.

The reason behind the loss of throughput is shown in Figure 4, where the source's window size is shown for the specific case of a loss probability of 1/150 messages. As the figure shows, the window is decreased often, once every time a message is corrupted. Thus, the window size never reaches its optimum value of 20, and thus the throughput is significantly less than optimal.

For the case of lhacks, the window first begins an exponential increase until it reaches the slow-start threshold of 20. Afterwards, it begins its linear growth in the congestion avoidance phase. To prevent a loss in throughput, whenever a fast retransmit due to corruption occurs, the window is not decreased. However, since the window continues to grow linearly, it eventually reaches a value large enough to overflow the buffer of the bottleneck router. This is a well-known problem of TCP [Kes91], but solving it is not the topic of this paper. This causes a fast retransmit due to congestion, and the window is reduced by half. Since the router can buffer an entire bandwidth-delay product of the source (20 messages), the window grows to twice its optimum. Thus, it does not drop below the optimum when halved and a high throughput is maintained.

The case of a mobile source sending to a stationary destination showed similar results and are not presented

here. For the case in which both the source and the destination are wireless, we did not encounter in our simulations a case when both a message and its lhack were lost due to corruption, so the results are similar to those above and are not repeated either. Again, even if such double loss occurred, it should not affect the overall throughput, since it should be a rare occurrence.

Since a source can distinguish between congestion and corruption losses, it must be able to handle a congestion period, i.e., a decrease in available bandwidth, in the same way as the standard TCP reno, even in the presence of corruption losses. We show this next. A new cross traffic source is added to the bottleneck with a rate of 20,000 bytes/sec. The source is initially inactive, it becomes active at time 10, and becomes inactive again at time 25. We compare the case where both the source and destination are stationary (no lhacks) against the case of a wireless destination with a corruption rate of 1/125 messages with lhacks.

Figure 5 shows the window size of the two cases. The window size increases until it causes congestion at around time 7, and is reduced by a half. It then resumes its linear increase, but congestion occurs prematurely at time 10 because the new cross traffic source becomes active. Since the bandwidth decreases, the maximum window size also decreases to about 20. When the new source becomes inactive, normal operation is resumed and the window can increase once more to its maximum of about 40. Notice that the behavior of the source in both the stationary case and the wireless case is similar, even though the wireless case suffers in addition from corruption losses. Hence, the ability to distinguish between corruption and congestion losses allows the source to correctly determine when to reduce its window size to achieve maximum throughput.

6. Summary and Concluding Remarks

Recent advances in hardware and communication technology have made possible the birth of mobile computing. However, the wireless media used in mobile computing

have different properties from those upon which the design of TCP was based. For example, wireless media are known to have high bit error rates [GP95], leading to significant corruption losses. TCP assumes that the network links are relatively error free, since the loss of a message is treated as a signal for congestion in the network. Thus, TCP reduces its window size, and hence also its sending rate, upon a message loss in order to alleviate network congestion. This approach is effective when losses due to corruption are rare, but not effective for the new and unreliable wireless media. If losses due to corruption are significant, the window will never reach its optimum value, and throughput will suffer.

In this paper, we presented an acknowledgment scheme that allows the transport protocol to distinguish between congestion and corruption losses. The router at the wireless network returns a last-hop acknowledgment (lhack) to the source for each message received. The source uses these lhacks to determine if the loss of a packet was due to congestion in the wired network or corruption in the wireless network, and thus it adjusts the window size accordingly. This scheme adds only a small amount of functionality to the routers at the wireless networks, while leaving all other routers unchanged. We showed how to incorporate this scheme into the TCP protocol, and demonstrate its effectiveness through simulation.

It is possible that mobility causes message loss due to hand-offs from one wireless cell to another. In this case, the source should not reduce its sending rate either, since the cause of the loss is not congestion. Notice that our acknowledgment scheme is also useful for this case. If the destination changes cells and messages are lost during hand-off, their lhacks will still arrive to the source, indicating that the window need not be decreased. However, if the source is mobile and changes location, then its is unknown if the losses are due to congestion or hand-off, so a slow-start must be performed to the current window size.

In [MGS94], a modification to TCP is suggested in which a mobile destination notifies its source that it has changed cells. If the source receives this notification and detects a loss, then it will not reduce its window size unless several timeouts occur, since it is assumed that the loss occurred due to the hand-off. This scheme suffers from the disadvantage that if a congestion loss occurs while the destination moves, the source will again send at the same rate, causing more congestion and loss for other connections sharing the same path. Furthermore, the scheme is not able to cope with corruption losses.

Finally, our acknowledgment strategy need not be limited to the TCP protocol. Many transport protocols use the loss of a message as a signal of congestion, and thus lower their throughput upon a message loss. Any protocol that behaves in this manner will increase its throughput in a wireless session if it incorporates our strategy.

References

- [BIV92] Badrinath B., Imielinski T., Virmani A., "Locating Strategies for Personal Communication Networks", *IEEE GLOBECOM 92 Workshop on Networking of Personal Communications Applications*, Dec.1992.
- [CEG94] Cobb J., Edmondson-Yurkanan C., Gouda M., "Universal Mobile Addressing", *Joint International Conference on Information Sciences*, 1994.
- [Com91] Comer D., *Internetworking with TCP/IP*, Volume I, Second Edition, Prentice-Hall Inc., 1991.
- [CPR92] Cohen D., Postel J, Rom R., "IP Addressing and Routing in a Local Wireless Network", *IEEE INFOCOM Conference*, 1992.
- [FZ94] Forman G., Zahorjan J., "The Challenges of Mobile Computing", *IEEE Computer*, April, 1994.
- [GP95] Gouda M. G., Paul S., "A Wireless Link Protocol: Design by Refinement", submitted for publication.
- [IB94] Imielinski T., Badrinath B., "Mobile Wireless Computing: Challenges in Data Management", *Communications of the ACM*, 1994.
- [IDM91] Ionnidis J., Duchamp D., Maguire G., "IP-based Protocols for Mobile Internetworking", *Proc. of the ACM SIGCOMM Conference*, 1991.
- [IP94] IETF/Mobile-IP Working Group, IP Mobility Support Internet Draft - work in progress, July 1994.
- [Jac88] Jacobson V., "Congestion Avoidance and Control", *ACM SIGCOMM Conference*, 1988.
- [Kes91] Keshav S., "Flow Control in High-Speed Networks with Long Delays", *USENIX Conference*, 1991.
- [Kes94] Keshav S., "The REAL Simulation Package", available from keshav@research.att.com.
- [KP91] Karn P., Partridge C., "Improving Round-Trip Time Estimates in Reliable Transport Protocols", *ACM Trans. on Computer Systems*, 9, 4, Nov. 1991.
- [MGS94] Manzoni P., Ghosal D., Serazzi G., "A Simulation Study of the Impact of Mobility on TCP/IP", *IEEE Int'l. Conf. on Network Protocols*, 1994.
- [PB93] Pitoura E., Bhargava B., "Dealing with Mobility: Issues and Research Challenges", Department of Computer Sciences Technical Report CSD-TR-93-070, Purdue University, November 1993.
- [Per93] Perkins C., "Providing Continuous Network Access to Mobile Hosts Using TCP/IP", *Computer Networks and ISDN Systems*, Vol. 26, pp. 357-369.
- [Pos81] Postel J., Internet Protocol, RFC-791, Sept. 1981.
- [TKT91] Teraoka F., Yokote Y., Tokoro M., "A Network Architecture Providing Host Migration Transparency", *ACM SIGCOMM Conference*, 1991.

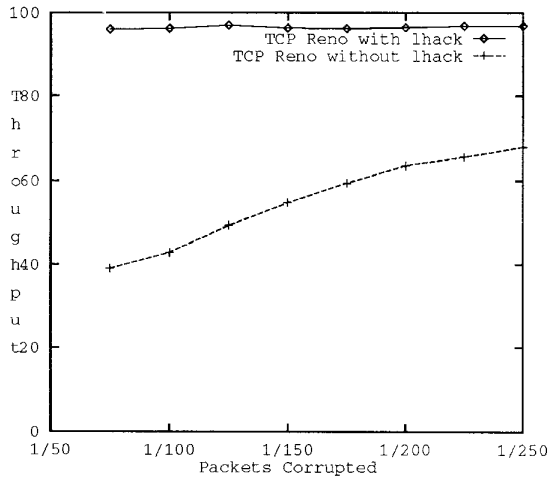


Figure 3: Throughput vs. corruption rate.

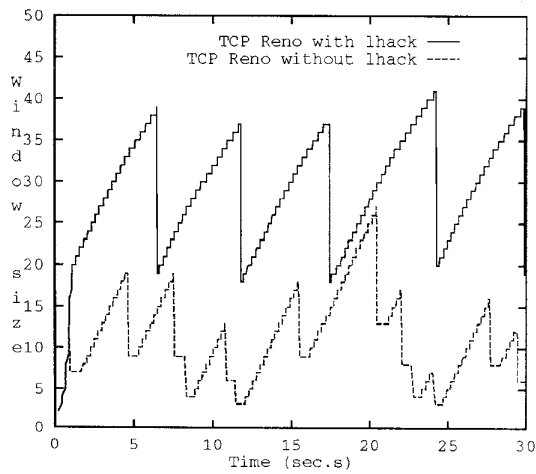


Figure 4: Window size vs. time, corruption rate 1/125.

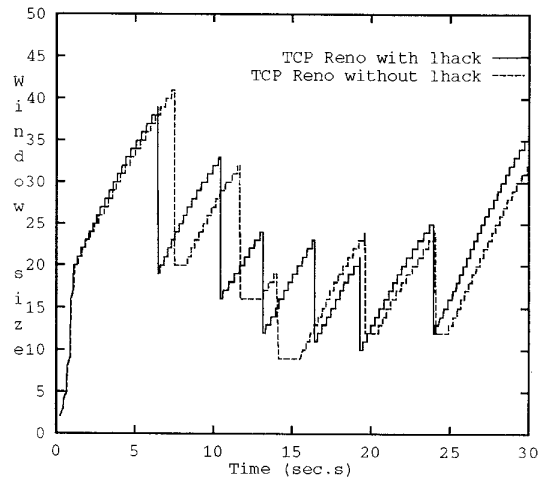


Figure 5: Window size during decrease in bandwidth.