GI 2013

Improving TCP Performance Over Wireless Links

Samir Goel and Dheeraj Sanghi
Computer Science & Engineering Department
Indian Institute of Technology
Kanpur - 208 016
dheeraj@iitk.ernet.in

March 31, 1998

Abstract

TCP is the most common transport layer protocol used in Internet today. It has been designed primarily for wired networks. Over the last few years, wireless technology has matured, and several people connect their laptops to the Internet through wireless techniques. The characteristics of wireless links is very different from wired links, particularly in terms of loss behaviour. This results in poor TCP performance over wireless links.

In this paper, we have proposed enhancements to TCP so that it can differentiate between a loss due to congestion in the wired network, and a loss due to noise in the wireless link. With this knowledge, TCP can react differently to the two kinds of losses, leading to improved performance. The modified TCP is compatible with existing implementations. Performance of a TCP connection over wired network is not affected by this change. The overhead of our scheme is also very low.

1 Introduction

Recent years have witnessed rapid progress in the area of mobile computing. This has largely been driven by growing popularity of laptop computers coupled with advances in wireless technology. The current trend strongly suggests that wireless links will be an integral part of future networks.

Wireless links have fundamentally different characteristics than wired links. This is because the environment through which the signal passes introduces noise and echoes. As a result, wireless links are characterized by low bandwidth and error rates that are high, bursty and time-varying. This difference in error characteristics causes a significant degradation in the performance when hosts on wireless LAN make use of reliable transport protocols, such as TCP, that were originally developed and adapted for optimum performance on wired networks.

TCP implementations have evolved considerably over the last decade. Algorithms have been added to adapt to the changing characteristics of wired networks. New congestion control policies have been introduced keeping in view the increased reliability (significantly lower bit error rates) of today's networks. So much so that they assume that packet losses are invariably due to congestion in the network. Consequently they react to packet losses by reducing the sending rate and exponentially backing off the retransmission timer. They have been largely successful in improving the performance of wired networks.

However, the performance of TCP degrades significantly if the loss behaviour of the network is different from the one assumed by TCP. In particular, TCP misinterprets packet loss over a wireless link as a sign of network congestion, causing poor throughput.

The performance degradation of TCP poses an important challenge to researchers. TCP is the most widely used transport protocol on the Internet today. (TCP traffic constitutes 80% of all the wide-area network traffic [4]). It is important that existing applications continue to work in presence of wireless links without severe degradation in their performance.

The problem has been studied by several researchers, and many solutions have been proposed [1, 2, 8, 11]. Nanda *et al.* [8] have suggested to solve the problem by introducing reliability at the link layer for wireless link using finite number of retransmissions. This approach does not completely shield the source transport layer from all losses on the wireless link, resulting in degraded performance. Also, studies [5] have shown that link-layer retransmissions may interfere with TCP's end-to-end retransmissions leading to degraded performance.

Bakre *et al.* [1] and Yavatkar *et al.* [11] advocate splitting end-to-end TCP connection into two separate TCP connections - one over the wired network and the other over the wireless link, with base station serving as the common point for two connections (A base station is a router that interfaces wireless link to the wired network). The advantage of this approach is that it isolates the source transport layer from the erratic behavior of the wireless link. It also allows quick recovery from errors over a wireless link. However, this approach does not preserve the semantics of TCP. Also, every packet incurs the overhead of going through the TCP protocol processing twice at the base station. Another disadvantage is that this approach requires the base station to maintain significant amount of state for every TCP connection passing through it. Lastly, in Bakre's approach [1], since TCP is not tuned for wireless links, TCP sender of the wireless link often times out causing the original sender to stall. This results in performance problems.

Balakrishnan *et al.* [2] have suggested adding a module called snoop agent to the routing code at the base station. This agent monitors packets flowing on a TCP connection and does intelligent processing, utilizing the knowledge of TCP. It maintains a cache of un-acknowledged TCP packets on a per-connection basis and performs local re-transmission when it detects a packet loss. This approach does not completely shield the sender from losses on a wireless link. Also, it assumes that a wireless link is the last hop in the network path. It also requires a base station to maintain significant TCP state information and a cache of unacknowledged packets for every TCP connection passing through it. Another disadvantage of this approach is that the snoop agent performs some additional processing for every TCP data packet or acknowledgment packet that passes through. Since a base station is a common point for many TCP connections, the cumulative effect of this per-packet processing overhead may become significant and may make base station a bottleneck. Lastly, the proposed solution is not symmetric. A base station has different set of policies for connection from fixed host to mobile host than for connection from mobile host to fixed host (assuming one-way connection). This further increases the complexity of a base station.

All proposed solutions are based on the assumption that no changes can be made to existing TCP implementations. It is, of course, impossible to change TCP on all existing hosts, but we note that it is indeed possible that newer hosts run modified TCP implementations. We need to only ensure that the modifications should be backward-compatible. And that the older hosts should not notice a further degradation of performance.

In this paper, we propose to augment TCP to respond to control signals sent from wireless gateway nodes. The control signals are such that they do not cause any sort of confusion in existing TCP implementations. They are unambiguosly discarded by the hosts of older versions that receive them. The simulation results show that the scheme provides substantial performance improvement with low cost overhead.

The rest of the paper is organized as follows. In Section 2, we propose a simple scheme. We present the simulation results, and point out a situation in which the performance does not improve. In Section 3, we refine and extend this scheme and present results which show improved performance. Section 4 concludes this paper and points out areas of future work.

2 Simple Scheme

In this section, we first propose a relatively simple solution. We conduct simulation experiments to show that it works in most circumstances, but our results also show that it does not improve performance under certain conditions. We propose a refinement to our basic scheme in the next section to take care of this problem.

The basic idea of our solution is that if TCP can distinguish a packet loss due to congestion from that due to noise on the channel, then algorithms can be developed to deal with such losses efficiently. In this paper, we make the assumption that losses in the wired network are due to congestion, while those in the wireless network are related to noise. These are the same assumptions made by researchers working on this problem, and follow directly from the error characteristics of these networks as shown by the studies of losses on these networks [7, 11].

Most other solutions have concentrated on either reducing noise-related losses by having link level retransmissions [3, 8], or isolating TCP on wired network by splitting the connection [1, 11] or having a snoop agent [2]. But in our solution, we would like source TCP entity to differentiate between two kinds of losses, and take appropriate action. This is a more general solution, since it will work even if the wireless link is not the last hop. (This is not to diminish the importance of reducing noise-related losses. TCP retransmissions are expensive, and if link level retransmissions can take care of the problem, they should be used.)

2.1 General Framework

In order that TCP be able to differentiate between two kinds of losses, the network needs to generate some control information and send it to the source TCP entity. A solution will have the following components.

- Timing, frequency and amount of information

This is very crucial. Every control packet is an overhead in the network. Sending too many packets, or large packets, may congest the network, and may do more damage. On the other hand, sending very little information, or sending it infrequently, may not permit TCP to take appropriate action in time.

- Who should generate and send the information

  The node that connects wired network to the wireless link has the most recent informatin, and should be the one responsible for generating and transmitting this information.

- How should this information be sent (protocol)

  There is no real need to define a new protocol for just this purpose. There is an existing protocol for carrying control information in the networks, that is, Internet Control Message Protocol (ICMP) [9]. We propose a new message type within ICMP to carry the required information. This has the additional advantage that nodes which do not understand this message-type will simply ignore the message, and therefore, will remain unaffected by the change. The new message type is called ICMP-DEFER.

- How should TCP handle the information

  This is arguably the most important aspect of the solution, and we will explain our method in subsequent sections.

In addition, we need to make sure that this solution will not have an adverse impact on existing hosts, except that they will continue to perform non-optimally.

2.2 Description of a Simple Scheme

In descrbing our scheme we will assume that the wireless link is the last hop on the connection, and that packets are flowing from wired side to the mobile node. This is only for ease of explanation. And this also happens to be the most common scenario in real life.

Based on discussion above, we need to decide on two things. One, the information, its timing and frequency, and Two, the heuristics for handling this information by TCP.

The policy for generation of control information is governed by the following considerations:

1. An ICMP message should allow TCP to distinguish between a packet loss on a wireless link and a packet loss on the wired network.
2. An ICMP message should help TCP in avoiding conflict between link layer retransmissions and end-to-end retransmissions.
3. The overhead in the network should be minimal.

In our scheme, the base station generates an ICMP-DEFER message when the first attempt at transmitting the packet on the wireless side is unsuccessful. This policy ensures that within one round trip time, TCP will receive either an acknowledgment for the packet or an ICMP message.

This will ensure that end-to-end retransmissions do not start while link-layer retransmissions may be going on. A lack of both will signal a congestion loss. Thus, TCP can distinguish between two kinds of losses. The control information consists of TCP and IP headers (typical ICMP message). This is enough for source TCP to decide which particular packet was lost on wireless link. ICMP-DEFER messages are not retransmitted, thus the overhead on the network is minimal.

TCP performs the following actions on receipt of an ICMP-DEFER message:

- If retransmit timer is set for the segment indicated in the ICMP message, it postpones its expiry by the current estimate of retransmission timeout (RTO) value. This is done to avoid any conflict between the local retransmissions at the base station and end-to-end retransmissions. Since, errors on a wireless link generally occur in burst, there is a high chance that a few subsequent local retransmission attempts at the base station will also fail. Hence, acknowledgment for this segment may get delayed. Therefore, it becomes important to postpone the retransmission timer.

  We assume that one RTO time (minimum value 1 second [10]) is sufficient for the base station to exhaust all local retransmission attempts for a packet. If the wireless link remain in error-state for longer duration, TCP would stop transmission because there would be no acks, and thus send- window would not move. TCP may use end-to-end retransmissions to probe whether the link has returned to good state.

- TCP will store information about the segments for which ICMP-DEFER messages were received. This information will be kept as long as these segments are not acknowledged by the destination.

If a segment needs to be retransmitted, TCP checks if it had received an ICMP-DEFER message for this segment. If it had, it would retransmit the segment but it does not reduce the *cwnd*. Also, when coming out of fast recovery algorithm, it resets *cwnd* back to its value before fast retransmission and fast recovery.

2.3 Simulation Experiments

We conducted several simulation experiments to test our solution. We describe our setup, the parameters for the simulation, and present the results below.

2.3.1 Setup

We used a simple network topology for conducting simulation experiments, as shown in Figure 1. The wired link in the topology is representative of the entire network path over the wired network. The propagation delay of this link represents the sum of propagation delay, queuing delay and the processing time of each node in the entire path over the wired network. We assume that the wired link is perfectly reliable. We further assume that the base station does not drop packets because of insufficient buffers.

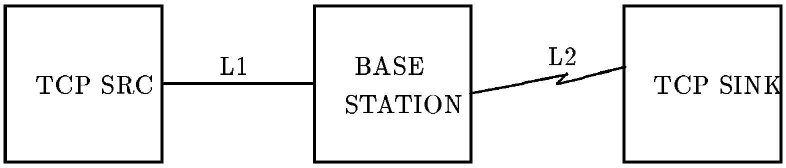

Figure 1: Network configuration used in Simulations

We simulated wireless link by a 2-state Markov model with the stay period in each state (GOOD, BAD) characterized by exponential distribution [3]. The propagation delay of the wireless link is representative of the processing time at the nodes as well.

2.3.2 Parameters

The following is the list of parameters that are of interest in the simulation: Packet size, size of file transfer, and receiver's buffer size (or advertized window size). For each link, we could set bandwidth and propagation delay. For wired link, we could set loss probability. For wireless link, the loss characteristics was described by two states: GOOD and BAD. For each state, we could set the duration of that state and the loss probability in that state. Duration was assumed to be exponentially distributed.

The most important parameter in the simulation was the loss characteristics of the wireless link. Varying the mean value of duration and loss probability for GOOD and BAD states allow us to simulate wireless link of different error characteristics. Other than that, varying the receiver's advertised window size allowed us to control the maximum amount of data that may be in-transit. Varying the propagation delay of the wired link allowed us to simulate various kinds of networks (LAN, WAN, etc.). A WAN can be simulated by keeping the propagation delay value large. A small value simulates a LAN.

Table 1 shows the values of various parameters that we used in the simulation experiments described in this section.

2.3.3 Results

We have conducted several experiments. A detailed discussion of them is available in [6]. In this paper, we present only the most interesting results. The performance criterion we chose to compare our scheme with existing TCP was throughput.

In one set of experiments, we compared the performance of TCP over wired network *No loss case*, normal TCP in presence of a wireless link *Unmodified TCP*, and TCP with our enhancements *Simple Scheme*. The experiment was repeated for different window size to control the amount of data in-transit. The propagation delay is set to 492 milli-seconds to highlight the effects of the clash between link layer retransmissions and end-to-end retransmissions. For more discussion on this, refer to [6].

The simulation results are summarized in the bar graph shown in Figure 2. It shows the time

Parameters	Value
Wired link	
- Bandwidth	8 Mbps
- Propagation delay	492 milli-seconds
- Loss probability	0 (reliable)
Wireless link	
- Bandwidth	1 Mbps
- Propagation delay	1 milli-seconds
- Error characteristics -	
$*$ GOOD state	
- mean value of exponential distribution	5 seconds
- loss probability	0 (reliable)
$*$ BAD state	
- mean value of exponential distribution	0.1 second
- loss probability	1
Packet size	1 KB
Size of file transfer	10 MB
Receiver's advertised window size	16 to 64 KB

Table 1: Parameter values used in simulations of Figure 2

taken for completion of the simulation run for the three schemes for different values of window size. The figure also shows the percentage performance improvement of *Simple Scheme* over *Unmodified TCP*. In the figure, `mean_bad_state` refers to the mean value of duration for BAD state of wireless link.

From the figure, we note that there is degradation in performance of *Unmodified TCP* and *Simple Scheme* over the *No loss case*. Further, we see that the degradation in performance of *Unmodified TCP* increases with the increase in window size. This is because it treats losses on the wireless link as signs of congestion and reduces its *cwnd* and *ssthresh*. The effect of this reduction becomes more prominent at larger values of window size. The performance of *Simple Scheme* does not suffer much degradation. As the figure shows, the percentage improvement of *Simple Scheme* over *Unmodified TCP* reaches 24.39% for window size of 64KB.

2.4 Problem

In *Simple Scheme*, TCP reacts to a packet loss on a wireless link by retransmitting the packet while maintaining the same sending rate (i.e., it does not reduce its *cwnd* and *ssthresh*). This results in substantial performance improvement. But, it also introduces a problem. For every packet lost on the wireless link (that is, all local retransmissions failed), TCP now recovers it by suffering either a fast retransmit or a retransmission timeout. Since, fast retransmit and fast recovery algorithms were designed to recover from single packet loss, in case when multiple packets in a transmission window are lost on the wireless link, TCP recovers them by suffering retransmission timeouts as many times as the number of losses. This is in contrast to unmodified TCP which recovers from multiple consecutive packet losses in a window by suffering at most one retransmission timeout.

As an illustration of the problem, consider the graph shown in Figure 3. The graph shows the reaction of TCP, based on *Simple Scheme*, to multiple consecutive packet losses in a window. For this example, the receiver's advertised window size was kept at 4 KB and propagation delay of wired link was kept at 170 milli-seconds (the remaining parameters' values being the same as mentioned in Table 1).

As indicated in the figure, two consecutive packets are lost on the wireless link. The two "$*$" marks on the graph indicate the time at which ICMP-DEFER messages were received by TCP, corresponding to these packets. The remaining two packets successfully reached the destination.

At 98003 milli-seconds, TCP suffers a retransmission timeout and retransmits the first lost packet. Since, it had received an ICMP-DEFER message for this packet, it simply resends the packet without reducing its *cwnd* and *ssthresh*. TCP again times out at 99354 milli-seconds and retransmits the second packet lost on the wireless link. Since, it had received an ICMP-DEFER message for this packet as well, it refrains from reducing its *cwnd* and *ssthresh*. In this case, TCP recovered from two packet losses by suffering two retransmit timeouts. It took 3054 milli-seconds to recover from these two losses.

The graph shown in Figure 4, illustrates the reaction of unmodified TCP under similar situation. Note that in this case, TCP recovered from two packet losses by suffering a single retransmit timeout. It took 1754 milli-seconds to recover from the two packet losses. This is almost half of the time required to recover in *Simple Scheme*.

In Simple Scheme, this large delay in recovering from multiple packet losses, detracts the advantage gained by not reducing the *cwnd* and *ssthresh*. Note that this situation is not specific to a particular value of window size. We observed similar pattern for larger window also, though the frequency of such patterns depends on window size [6].

The performance improvement gained by *Simple Scheme* then depends on how many times such multiple packet losses occur in a transfer. Simulation with window size of 4 KB showed that the use of ICMP messages resulted in degraded performance! While *Unmodified TCP* took 929824 milli-seconds to complete the transfer, *Simple Scheme* took 945579 milli-seconds to complete the transfer. A closer look reveals that, during the course of transfer, *Unmodified TCP* suffered 74 packet losses on the wireless link. It recovered them with 26 fast retransmits and 19 retransmit timeouts. On the other hand, *Simple Scheme* suffered 67 packet losses on the wireless link during the transfer. It recovered them with 23 Fast Retransmits and a staggering 42 retransmit timeouts.

3 Refined Scheme

In this section, we discuss the problem with Simple Scheme, and propose modifications for better performance.

3.1 Problem with Simple Scheme

The main problem with *Simple scheme* is that the loss detection latency is very high. ICMP-DEFER message only indicates that the segment may get lost due to noise, but source has no way to find out when exactly the segment was discarded by the base station, or more importantly, whether multiple segments were discarded by the base station. Without this information, TCP depends on retransmit timer for loss detection, but that detects only one-segment loss. For multiple losses, the timer has to be set that many times.

3.2 Modifications to Simple Scheme

To solve the above-mentioned problem, we decided that the base station should also inform TCP about losses, and not just the fact that there is noise in the link. The base station generates an ICMP error message when all the local retransmission attempts at the base station have been exhausted. We refer to this type of ICMP message as ICMP-RETRANSMIT message. Note that we still need ICMP-DEFER messages to avoid conflict between end-to-end and link level retransmissions.

The TCP response to two types of ICMP messages is summarized below.

1. On receiving an ICMP-DEFER message, TCP postpones the retransmission timer if the lost segment is same as the one for which timer is active. But TCP does not store information about DEFER messages.

2. When TCP receives an ICMP-RETRANSMIT message, it retransmits the segment indicated. It also stores the information that the ICMP-RETRANSMIT message was received for this segment.

3. An ICMP-RETRANSMIT message indicates that one segment was lost from the chain of segments sent by the source TCP. When the destination receives subsequent packets, it generates duplicate acknowledgments. When the source TCP receives the first of such duplicate acknowledgments, it switches to fast recovery algorithm. It discards the information that it received an ICMP-RETRANSMIT message for this segment. When it finally receives a new acknowledgement, it comes out of the fast recovery algorithm and resets *cwnd* to the value prior to its entering the fast recovery phase.

4. When source TCP suffers a retransmit timeout, it follows normal TCP algorithms.

5. When source TCP receives duplicate acknowledgments for a segment for which it has not received ICMP-RETRANSMIT message earlier, it follows normal TCP algorithms.

The modified reaction to ICMP-DEFER message enhances the scheme to work on networks with a wireless link anywhere in the network path (and not necessarily the last hop) [6].

3.3 Simulation Results

The figures at the end of the paper summarize the simulation results for different values of propagation delay of the wired link and for different error characteristics of the wireless link (the remaining parameters' value are same as mentioned in Table 1).

Each of the bar charts in Figures 5–11, show the time taken for file transfer for various window sizes. The four schemes compared are: *No Loss Case*, *Unmodified TCP*, *Simple Scheme*, and (*Refined Scheme*).

Results shown in Figure 5 were generated using the parameters' value given in Table 1. The figure also shows the percentage improvement in performance of *Simple Scheme* and *Refined Scheme* over *Unmodified TCP*.

We note that the *Refined Scheme* performs better than *Simple Scheme* for all values of window size. As the figure shows, the percentage improvement of *Refined Scheme* over *Unmodified TCP* reaches 28.96%.

Table 2 gives the range of parameters' values used in simulations. Figures 6 to 11 show the results. In these figures, we show the *percentage degradation* in performance of each scheme with respect to the *No Loss Case*. We also mention the percentage improvement of the *Refined Scheme* over the *Unmodified TCP*, in parentheses in the bar chart. We note that the performance of *Refined Scheme* is better than that of *Simple Scheme* and *Unmodified TCP*, for all values of window size.

Parameters	Value
Propagation delay of wired link	50 to 100 milli-seconds
Error characteristics of wireless link	
- GOOD state	
- mean value of exponential distribution	5 seconds
- loss probability	0
- BAD state	
- mean value of exponential distribution	0.1 to 0.5 second
- loss probability	1

Table 2: Table showing parameter values used in the simulation (figures 6 to 11)

Note that when `mean_bad_state` value is 0.1 second, the improvement of *Refined Scheme* over *Unmodified TCP* is not siginifacnt. This is because as the duration of BAD state reduces, the number of losses reduce. The performance in this case is anyway close to that of *No Loss Case*.

The performance of *Simple Scheme* is, many a times, worse than *Unmodified TCP*. This happens especially at higher values of `mean_bad_state`. This is because when the duration of BAD state increases, the possibility of multiple losses increases. As we have discussed earlier, *Simple Scheme* is unable to recover from multiple losses in an efficient fashion.

3.4 Advantages of Refined Scheme

Below, we summarize the advantages of our scheme.

- Modified TCP performs significantly better than existing TCP over network paths involving wireless links. At the same time, there is no difference in performance or overheads when used in wired networks. (There won't be any ICMP messages, and none of the modifications would come into play.)
- Overhead of the scheme is very low. During simulation runs involving a 10MB file transfer (10,000 packets transfer), the overhead in terms of number of packets peaked at 5%. Mostly, it remained less than 1%. Overhead in terms of bandwidth used was significantly lower, since ICMP packets are very small.
- ICMP messages are silently discarded by hosts which do not implement these enhancements.
- Our scheme preserves the semantics of TCP, unlike the 'Split connection' approach.
- In our scheme, generation of ICMP messages is triggered by the link layer for the wireless link. So, in the case when a mobile host is acting as the source of packets, ICMP messages will be triggered by the link layer of the mobile host. This policy makes our scheme symmetric. There is no special case when mobile host acts as the source.
- The scheme does not make any assumption about the location of the wireless link in the network path.

4 Conclusions

In this paper, we have proposed an ICMP based scheme to improve the performance of TCP over network paths involving wireless links. In our scheme, we have proposed enhancements to TCP to respond to certain control information from the network. This enables TCP to differentiate between losses on a wireless link and those on the wired network, resulting in significant improvement in performance. The control information is generated by the base station, and is carried to the source using new message-types in ICMP. There are two types of messages, ICMP-DEFER, when the base station transmits the packet for the first time and fails, and ICMP-RETRANSMIT, when the base station discards the packet after several link layer retransmissions.

The simulation results show that the proposed scheme achieves upto about 29% improvement in performance (measured in terms of the time take for completion of simulated data transfer) over unmodified TCP. The results also show that the scheme is significantly more robust in face of burst errors on the wireless link and its performance gain improves with the increase in frequency of burst errors on the wireless link.

The scheme is compatible with existing TCP implementations, and there is no difference in performance or overheads when used in wired networks. ICMP messages are silently discarded by hosts which do not implement these enhancements.

Overhead of the scheme is very low. During simulation runs involving a 10MB file transfer (10,000 packets transfer), the overhead in terms of number of packets peaked at 5%. Mostly, it remained less than 1%. Overhead in terms of bandwidth used was significantly lower, since ICMP packets are very small.

In our scheme, generation of ICMP messages is triggered by the link layer for the wireless link. So, in the case when a mobile host is acting as the source of packets, ICMP messages will be triggered by the link layer of the mobile host. This policy makes our scheme symmetric. There is no special case when mobile host acts as the source.

The scheme does not make any assumption about the location of the wireless link in the network path.

References

[1] A. Bakre and B.R. Badrinath. I-TCP : Indirect TCP for mobile hosts. In *Proc. 15th International Conference on Distributed Computing Systems (ICDCS)*, May 1995.

[2] H. Balakrishnan, S. Seshan, E. Amir, and R.H. Katz. Improving TCP/IP performance over wireless networks. In *Proc. 1st ACM International Conference on Mobile Computing and Networking (Mobicom)*, November 1995.

[3] P. Bhagwat, P. Bhattacharya, A. Krishna, and S.K. Tripathi. Enhancing throughput over wireless LANs using channel state dependent packet scheduling. In *Proc. IEEE INFOCOM'96, San Francisco, CA*, pages 1133–1140, March 1996.

[4] R. Caceres, P.B. Danzig, S. Jamin, and D.J. Mitzel. Characteristics of wide-area TCP/IP conversations. In *Proc. ACM SIGCOMM'91, Zurich, Switzerland*, pages 101–112, September 1991.

[5] A. DeSimone, M.C. Chuah, and O.C. Yue. Throughput performance of transport layer protocols over wireless LANs. In *Proc. Globecom'93*, December 1993.

[6] S. Goel. Improving the performance of TCP over wireless links. Master's thesis, Indian Institute of Technology, Kanpur, May 1997.

[7] V. Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM'88, Stanford, CA*, pages 314–329, August 1988.

[8] S. Nanda, R. Ejzak, and B.T. Doshi. A retransmission scheme for circuit mode data on wireless links. *IEEE Journal on Selected Areas in Communications*, 12(8), October 1994.

[9] J. B. Postel. Internet Control Message Protocol. RFC: 792, Network Information Center, ISI, University of Southern California, Marina del Rey, CA, September 1981.

[10] G. R. Wright and W. R. Stevens. *TCP/IP Illustrated, Volume 2 : The Implementation.* Addison-Wesley, Reading, MA, 1996.

[11] R. Yavatkar and N. Bhagwat. Improving end-to-end performance of TCP over mobile internetworks. *In Mobile 94 Workshop on Mobile Computing Systems and Applications*, December 1994.

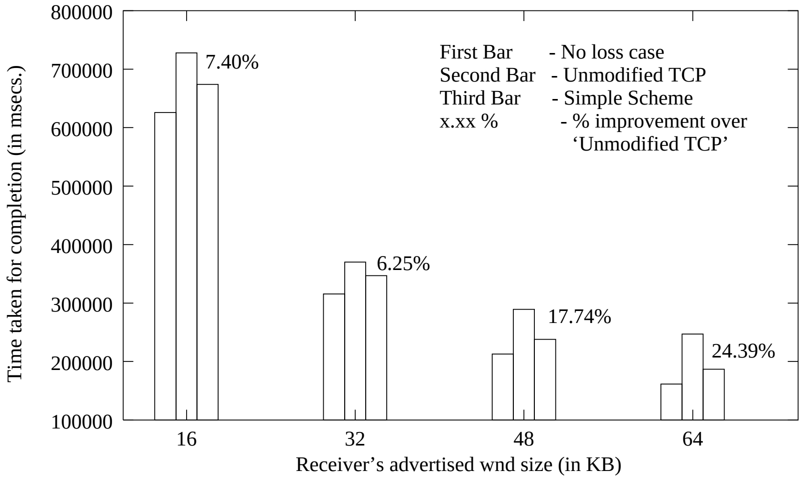

Figure 2: Propagation delay of wired link: 492 ms and `mean_bad_state`: 0.1 sec.

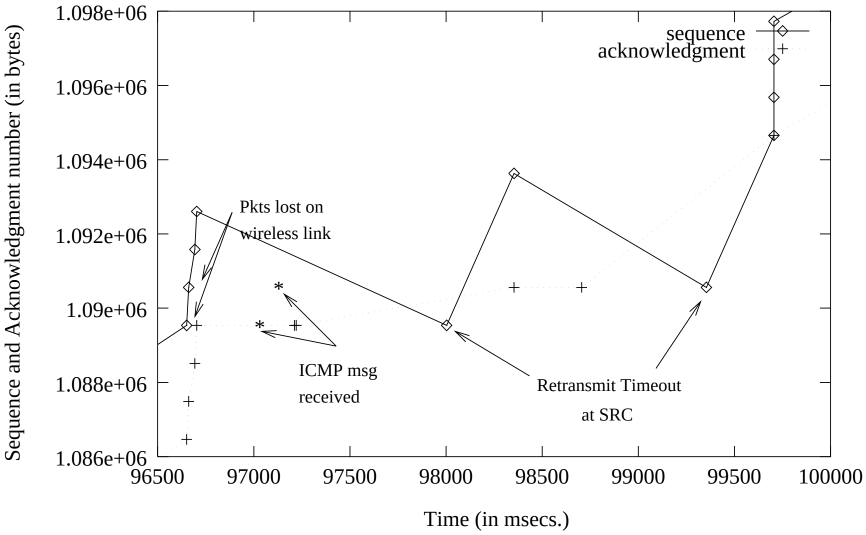

Figure 3: Graph illustrating problem with Simple Scheme

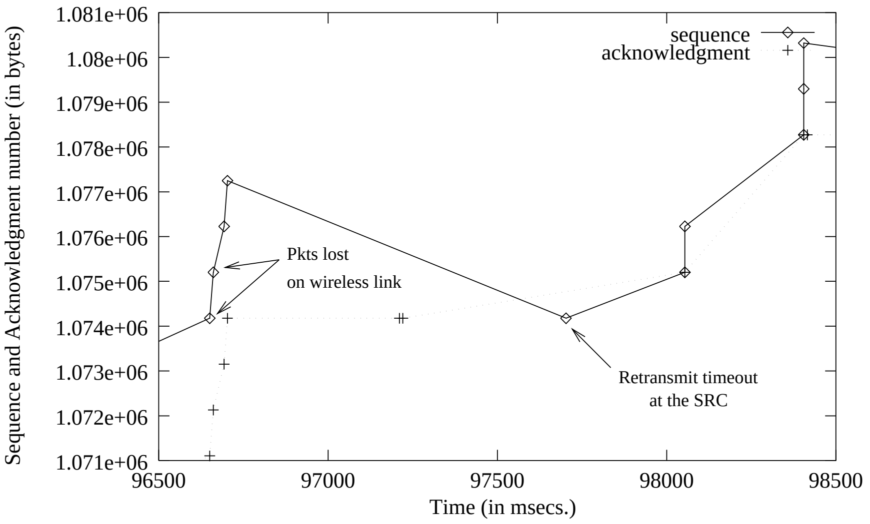

Figure 4: Response of (unmodified) TCP to multiple packet losses

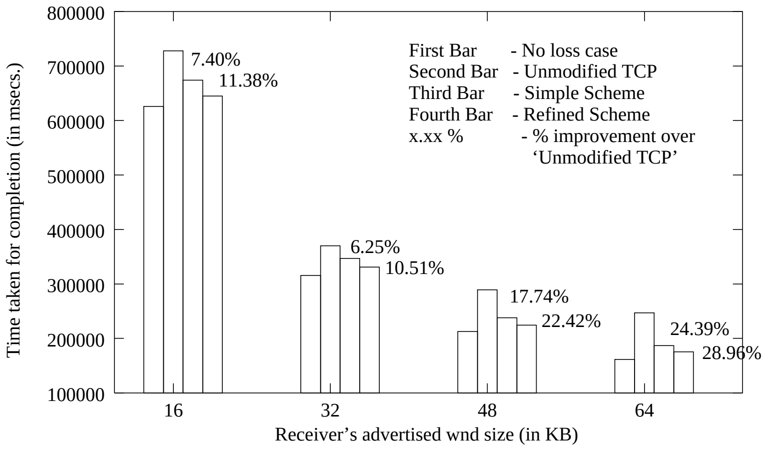

Figure 5: Propagation delay of wired link: 492 ms and `mean_bad_state`: 0.1 sec.

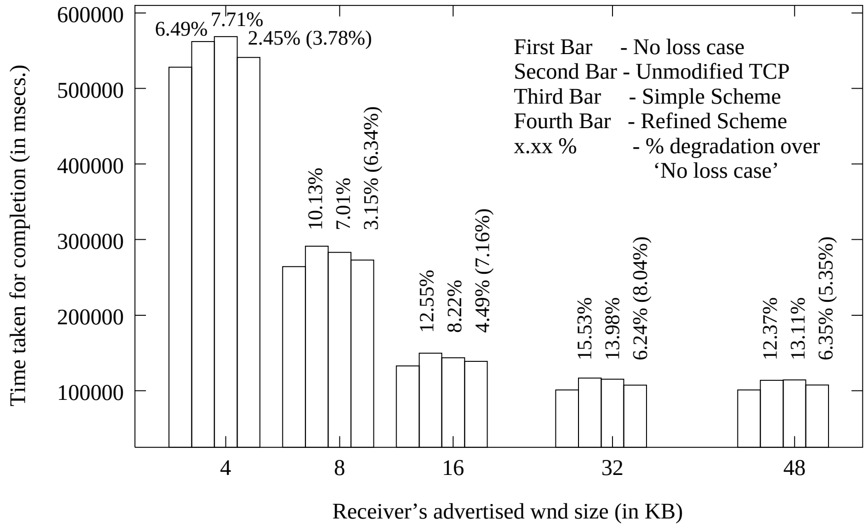

Figure 6: Propagation delay of wired link: 100 ms and `mean_bad_state`: 0.1 sec.

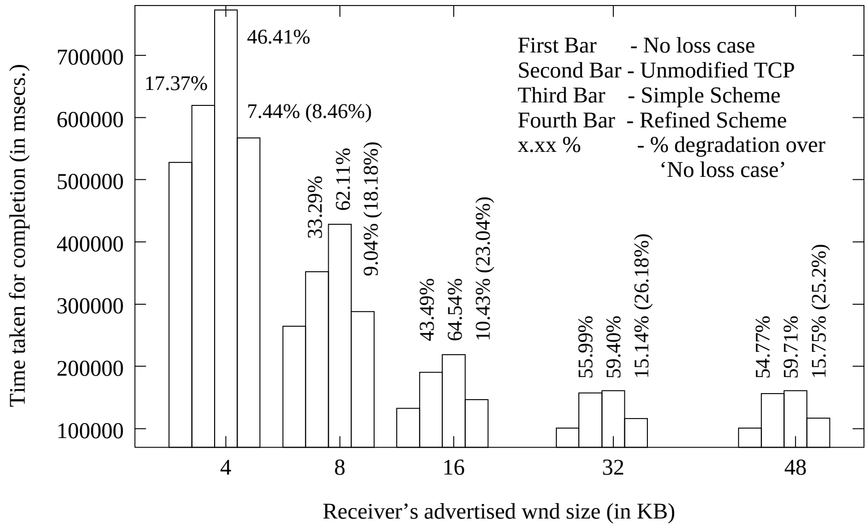

Figure 7: Propagation delay of wired link: 100 ms and `mean_bad_state`: 0.3 sec.

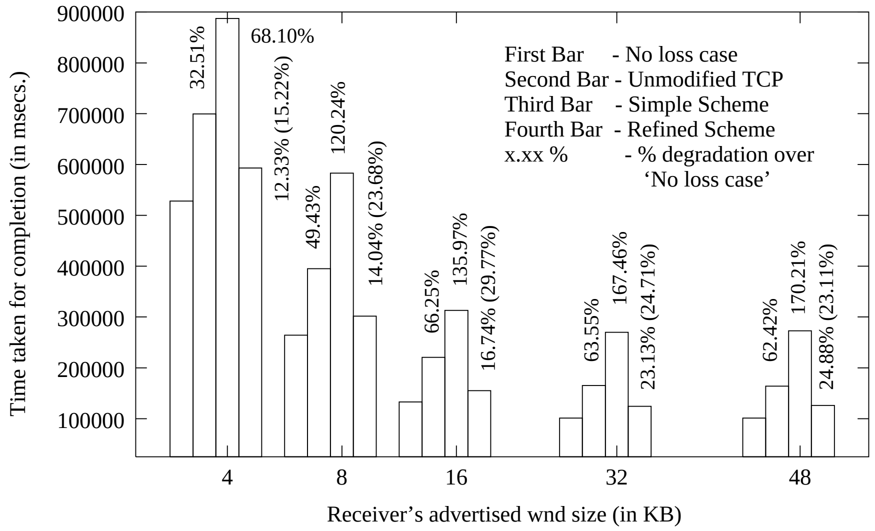

Figure 8: Propagation delay of wired link: 100 ms and `mean_bad_state`: 0.5 sec.

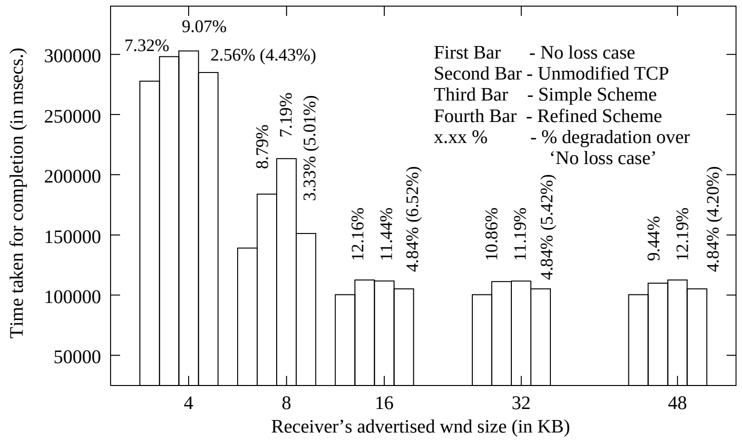

Figure 9: Propagation delay of wired link: 50 ms and `mean_bad_state`: 0.1 sec.

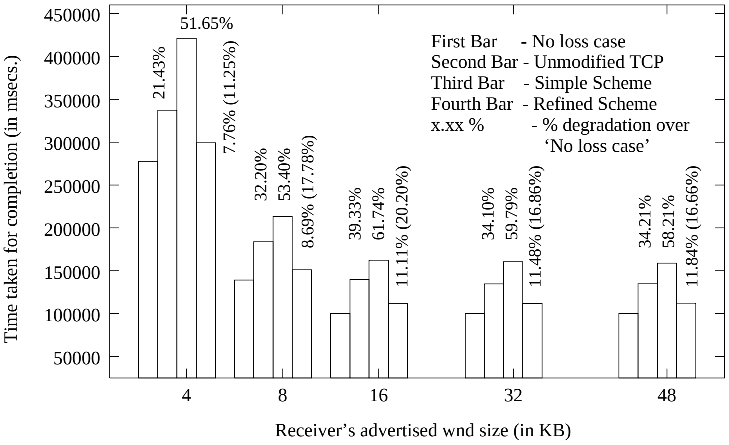

Figure 10: Propagation delay of wired link: 50 ms and `mean_bad_state`: 0.3 sec.

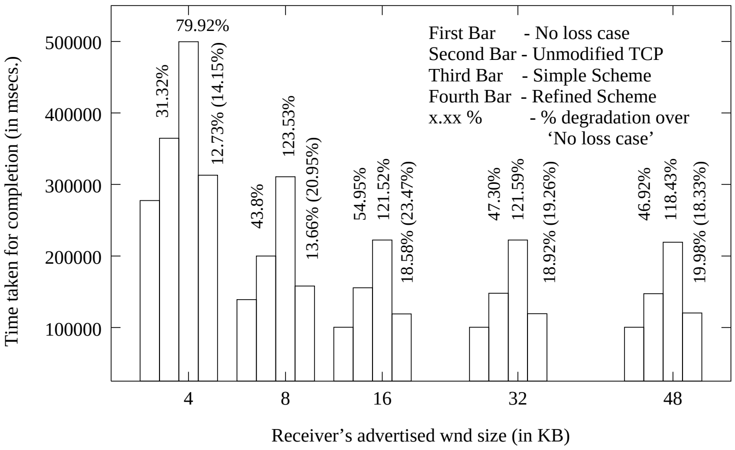

Figure 11: Propagation delay of wired link: 50 ms and `mean_bad_state`: 0.5 sec.