To appear, Proc. ACM SIGCOMM '96, Stanford, CA, August 1996.

A Comparison of Mechanisms for Improving TCP Performance over Wireless Links

Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan and Randy H. Katz[1]
{hari,padmanab,ss,randy}@cs.berkeley.edu
Computer Science Division, University of California at Berkeley

Abstract

Reliable transport protocols such as TCP are tuned to perform well in traditional networks where packet losses occur mostly because of congestion. However, networks with wireless and other lossy links also suffer from significant non-congestion-related losses due to reasons such as bit errors and handoffs. TCP responds to all losses by invoking congestion control and avoidance algorithms, resulting in degraded end-to-end performance in wireless and lossy systems. In this paper, we compare several schemes designed to improve the performance of TCP in such networks. These schemes are classified into three broad categories: end-to-end protocols, where the sender is aware of the wireless link; link-layer protocols, that provide local reliability; and split-connection protocols, that break the end-to-end connection into two parts at the base station. We present the results of several experiments performed in both LAN and WAN environments, using throughput and goodput as the metrics for comparison.

Our results show that a reliable link-layer protocol with some knowledge of TCP provides very good performance. Furthermore, it is possible to achieve good performance without splitting the end-to-end connection at the base station. We also demonstrate that selective acknowledgments and explicit loss notifications result in significant performance improvements.

1. Introduction

The increasing popularity of wireless networks indicates that wireless links will play an important role in future internetworks. Reliable transport protocols such as TCP [18, 19] have been tuned for traditional networks comprising wired links and stationary hosts. These protocols assume *congestion* in the network to be the primary cause for packet losses and unusual delays. TCP performs well over such networks by adapting to end-to-end delays and packet losses caused by congestion. The TCP sender uses the cumulative acknowledgments it receives to determine which packets have reached the receiver, and provides reliability by retransmitting lost packets. For this purpose, it maintains a running average of the estimated round-trip delay and the mean linear deviation from it. The sender identifies the loss of a packet either by the arrival of several duplicate cumulative acknowledgments or the absence of an acknowledgment for the packet within a *timeout* interval equal to the sum of the smoothed round-trip delay and four times its mean deviation. TCP reacts to any packet losses by dropping its transmission (congestion) window size before retransmitting packets, initiating congestion control or avoidance mechanisms (e.g., slow start [8]) and backing off its retransmission timer (Karn's Algorithm [11]). These measures result in a reduction in the load on the intermediate links, thereby controlling the congestion in the network.

Unfortunately, when packets are lost in networks for reasons other than congestion, these measures result in an unnecessary reduction in end-to-end throughput and suboptimal performance. Communication over wireless links is often characterized by sporadic high bit-error rates, and intermittent connectivity due to handoffs. TCP performance in such networks suffers from significant throughput degradation and very high interactive delays [5].

Recently, several schemes have been proposed to the alleviate the effects of non-congestion-related losses on TCP performance over networks that have wireless or similar high-loss links [2, 3, 21]. These schemes choose from a variety of mechanisms, such as local retransmissions, split-TCP connections, and forward error correction, to improve end-to-end throughput. However, it is unclear to what extent each of the mechanisms contributes to the improvement in performance. In this paper, we examine and compare the effectiveness of these schemes and their variants, and experimentally analyze the individual mechanisms and the degree of performance improvement due to each.

There are two fundamentally different approaches to improving TCP performance in such lossy systems. The first approach hides any non-congestion-related losses from

---

1. Web page URL http://daedalus.cs.berkeley.edu.
Srinivasan Seshan is now at IBM T.J. Watson Research Center, Hawthorne, NY (srini@watson.ibm.com).
This work was supported by DARPA Contract DAAB07-C-D154.

the TCP sender and therefore requires no changes to existing sender implementations. The intuition behind this approach is that since the problem is local, it should be solved locally, and that the transport layer need not be aware of the characteristics of the individual links. Protocols that adopt this approach attempt to make the lossy link appear as a higher quality link with a reduced effective bandwidth. As a result, most of the losses seen by the TCP sender are caused by congestion. Examples of this approach include wireless links with reliable link layer protocols such as AIRMAIL [1], split connection approaches such as Indirect-TCP [2], and TCP-aware link-layer schemes such as the snoop protocol [3]. The second class of techniques attempts to make the sender aware of the existence of wireless hops and realize that some packet losses are not due to congestion. The sender can then avoid invoking congestion control algorithms when non-congestion-related losses occur — we describe some of these techniques in Section 3. Finally, it is possible for a wireless-aware transport protocol to coexist with link-layer schemes to achieve good performance.

We classify the many schemes into three basic groups, based on their fundamental philosophy: end-to-end proposals, split-connection proposals and link-layer proposals. The end-to-end protocols attempt to make the TCP sender handle losses through the use of two techniques. First, they use some form of selective acknowledgments (SACKs) to allow the sender to recover from multiple packet losses in a window without resorting to a coarse timeout. Second, they attempt to have the sender distinguish between congestion and other forms of losses using an Explicit Loss Notification (ELN) mechanism. At the other end of the solution spectrum, split-connection approaches completely hide the wireless link from the sender by terminating the TCP connection at the base station. Such schemes use a separate reliable connection between the base station and the destination host. The second connection can use techniques such as negative or selective acknowledgments, rather than just regular TCP, to perform well over the wireless link. The third class of protocols, link-layer solutions, lie between the other two classes. These protocols attempt to hide link-related losses from the TCP sender by using local retransmissions and perhaps forward error correction [e.g., 13] over the wireless link. The local retransmissions use techniques that are tuned to the characteristics of the wireless link to provide a significant increase in performance. Since the end-to-end TCP connection passes through the lossy link, the TCP sender may not be fully shielded from wireless losses. This can happen either because of timer interactions between the two layers [5], or more likely because of TCP's duplicate acknowledgments causing sender fast retransmissions even for segments that are locally retransmitted. As a result, some proposals to improve TCP performance use mechanisms based on the knowledge of TCP messaging to shield the TCP sender more effectively and avoid competing and redundant retransmissions [3].

In this paper, we evaluate the performance of several end-to-end, split-connection and link-layer protocols using end-to-end throughput and goodput as performance metrics, in both LAN and WAN configurations. In particular, we seek to answer the following specific questions:

1. What combination of mechanisms results in best performance for each of the protocol classes?
2. How important is it for link-layer schemes to be aware of TCP algorithms to achieve high end-to-end throughput?
3. How useful are selective acknowledgments in dealing with lossy links, especially in the presence of burst losses?
4. Is it important for the end-to-end connection to be split in order to effectively shield the sender from wireless losses and obtain the best performance?

We answer these questions by implementing and testing the various protocols in a wireless testbed consisting of Pentium PC base stations and IBM ThinkPad mobile hosts communicating over a 915 MHz AT&T Wavelan, all running BSD/OS 2.0. For each protocol, we measure the end-to-end throughput, and goodputs for the wired and (one-hop) wireless paths. For any path (or link), goodput is defined as the ratio of the actual transfer size to the total number of bytes transmitted over that path. In general, the wired and wireless goodputs differ because of wireless losses, local retransmissions and congestion losses in the wired network. These metrics allow us to determine the end-to-end performance as well as the transmission efficiency across the network. While we used a wireless hop as the lossy link in our experiments, we believe our results are applicable in a wider context to links where significant losses occur for reasons other than congestion.

We show that a reliable link-layer protocol with some knowledge of TCP results in very good performance. Our experiments indicate that shielding the TCP sender from duplicate acknowledgments caused by wireless losses improves throughput by 10-30%. Furthermore, it is possible to achieve good performance without splitting the end-to-end connection at the base station. We also demonstrate that selective acknowledgments and explicit loss notifications result in significant performance improvements. For instance, the simple ELN scheme we evaluated improved the end-to-end throughput by a factor of more than two compared to TCP Reno, with comparable goodput values.

The rest of this paper is organized as follows. Section 2 briefly describes some proposed solutions to the problem of reliable transport protocols over wireless links. Section 3 describes the implementation details of the different protocols in our wireless testbed, and Section 4 presents the results and analysis of several experiments. We present our conclusions in Section 5, and mention some future work in Section 6.

2. Related Work

In this section, we summarize some protocols that have been proposed to improve the performance of TCP over wireless links. We also briefly describe some proposed methods to add SACKs to TCP.

- **Link-layer protocols**: There have been several proposals for reliable link-layer protocols. The two main classes of techniques employed by these protocols are: error correction (using techniques such as forward error correction (FEC)), and retransmission of lost packets in response to automatic repeat request (ARQ) messages. The link-layer protocols for the digital cellular systems in the U.S. — both CDMA [10] and TDMA [17] — primarily use ARQ techniques. While the TDMA protocol guarantees reliable, in-order delivery of link-layer frames, the CDMA protocol only makes a limited attempt and leaves it to the (reliable) transport layer to recover from errors in the worst case. The AIRMAIL protocol [1] employs a combination of FEC and ARQ techniques for loss recovery.

  The main advantage of employing a link-layer protocol for loss recovery is that it fits naturally into the layered structure of network protocols. The link-layer protocol operates independently of higher-layer protocols (which makes it applicable to a wide range of scenarios), and consequently, does not maintain any per-connection state. The main concern about link-layer protocols is the possibility of adverse effect on certain transport-layer protocols such as TCP. We investigate this in detail in our experiments.

- **Indirect-TCP (I-TCP) protocol [2]:** This was one of the early protocols to use the split-connection approach. It involves splitting each TCP connection between a sender and receiver into two separate connections at the base station — one TCP connection between the sender and the base station, and the other between the base station and the receiver. In our classification of protocols, I-TCP is a split-connection solution that uses regular TCP for its connection over wireless link.

  I-TCP, like other split-connection proposals, attempts to separate loss recovery over the wireless link from that across the wireline network, thereby shielding the original TCP sender from the wireless link. However, as our experiments indicate, the choice of TCP over the wireless link results in several performance problems. Since TCP is not well-tuned for the lossy link, the TCP sender of the wireless connection often times out, causing the original sender to stall. In addition, every packet incurs the overhead of going through TCP protocol processing twice at the base station (as compared to zero times for a non-split-connection approach), although extra copies are avoided by an efficient kernel implementation. Another disadvantage of this approach is that the end-to-end semantics of TCP acknowledgments is violated, since acknowledgments to packets can now reach the source even before the packets actually reach the mobile host. Also, since this protocol maintains a significant amount of state at the base station per TCP connection, handoff procedures tend to be complicated and slow.

- **The Snoop Protocol [3]:** The snoop protocol introduces a module, called the *snoop agent*, at the base station. The agent monitors every packet that passes through the TCP connection in both directions and maintains a cache of TCP segments sent across the link that have not yet been acknowledged by the receiver. A packet loss is detected by the arrival of a small number of duplicate acknowledgments from the receiver or by a local timeout. The snoop agent retransmits the lost packet if it has it cached and suppresses the duplicate acknowledgments. In our classification of the protocols, the snoop protocol is a link-layer protocol that takes advantage of the knowledge of the higher-layer transport protocol (TCP).

  The main advantage of this approach is that it suppresses duplicate acknowledgments for TCP segments lost and retransmitted locally, thereby avoiding unnecessary fast retransmissions and congestion control invocations by the sender. The per-connection state maintained by the snoop agent at the base station is *soft*, and is not essential for correctness. Like other link-layer solutions, the snoop approach could also suffer from not being able to completely shield the sender from wireless losses.

- **Selective Acknowledgments**: Since standard TCP uses a cumulative acknowledgment scheme, it often does not provide the sender with sufficient information to recover quickly from multiple packet losses within a single transmission window. Several studies [e.g., 6] have shown that TCP enhanced with selective acknowledgments performs better than standard TCP in such situations. SACKs were added as an option to TCP by RFC 1072 [9]. However, disagreements over the use of SACKs prevented the specification from being adopted, and the SACK option was removed from later TCP RFCs. Recently, there has been renewed interest in adding SACKs to TCP. Two of the more interesting proposals are the TCP SACKs Internet Draft [14] and the SMART scheme [12].

  The Internet Draft proposes that each acknowledgment contain information about up to three non-contiguous blocks of data that have been received successfully. Each block of data is described by its starting and ending sequence number. Due to the limited number of blocks,

Name	Category	Special Mechanisms
E2E	end-to-end	standard TCP-Reno
E2E-NEWRENO	end-to-end	TCP-NewReno
E2E-SACK	end-to-end	selective acks (SACKs)
E2E-ELN	end-to-end	explicit loss notification (ELN)
E2E-ELN-RXMT	end-to-end	ELN with retransmit on first dupack
LL	link-layer	none
LL-TCP-AWARE	link-layer	duplicate ack suppression
LL-SACK	link-layer	SACKs
LL-OPT	link-layer	SACKs and duplicate ack suppression
SPLIT	split-connection	none
SPLIT-SACK	split-connection	SACK-based wireless connection

Table 1. Summary of Protocols

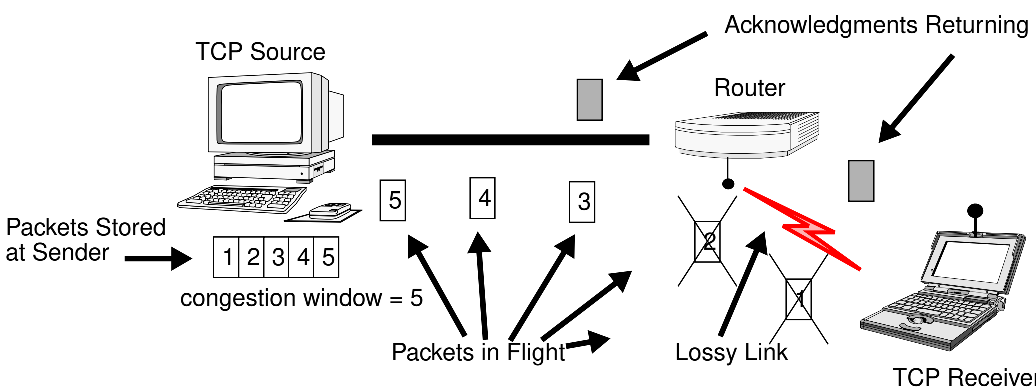


Figure 1. A typical loss situation

it is best to inform the sender about the most recent blocks received.

An alternate proposal, SMART, uses acknowledgments that contain the cumulative acknowledgment and the sequence number of the packet that caused the receiver to generate the acknowledgment (this information is a subset of the three-blocks scheme proposed in the Internet Draft). The sender uses these SACKs to create a bitmask of packets that have been successfully received. This scheme trades off some resilience to reordering and lost acknowledgments in exchange for a reduction in overhead to generate and transmit acknowledgments.

3. Implementation Details

This section describes the protocols we have implemented and evaluated. Table 1 summarizes the key ideas in each scheme and the main differences. Figure 1 shows a typical loss situation over the last link. Here, the TCP sender is in the middle of a transfer across a two-hop network to a mobile host. At the depicted time, the sender's congestion window consists of 5 packets. Of the five packets in the network, the first two packets are lost on the wireless link. For each protocol, we show the messages generated by the receiver and the response from the base station and source nodes in Figures 2 through 9.

3.1 End-To-End Schemes

Although a wide variety of TCP versions are used on the Internet, the current de facto standard for TCP implementations is TCP-Reno [19]. We call this the E2E protocol, and use it as the standard basis for performance comparison (Figure 2).

The E2E-NEWRENO protocol improves the performance of TCP-Reno after multiple packet losses in a window by remaining in fast recovery mode if the first new acknowledgment received after a fast retransmission is "partial", i.e, is less than the value of the last byte transmitted when the fast retransmission was done. This method enables the connection to make progress at the rate of one segment per round trip time, rather than stall until a coarse timeout [6, 7].

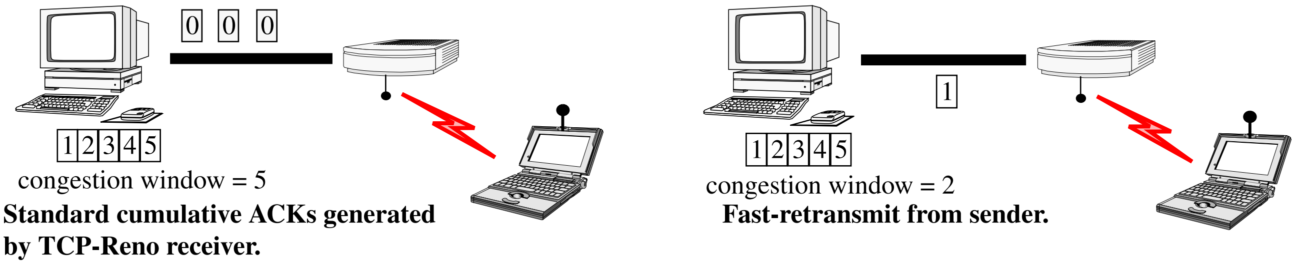


Figure 2. Normal TCP

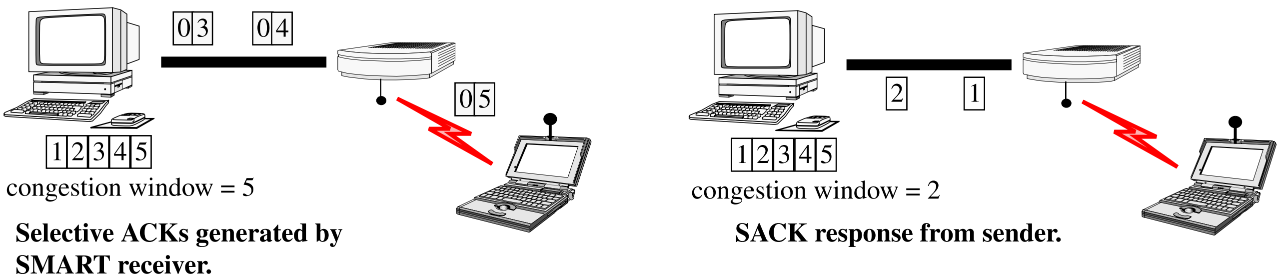


Figure 3. TCP with SACK

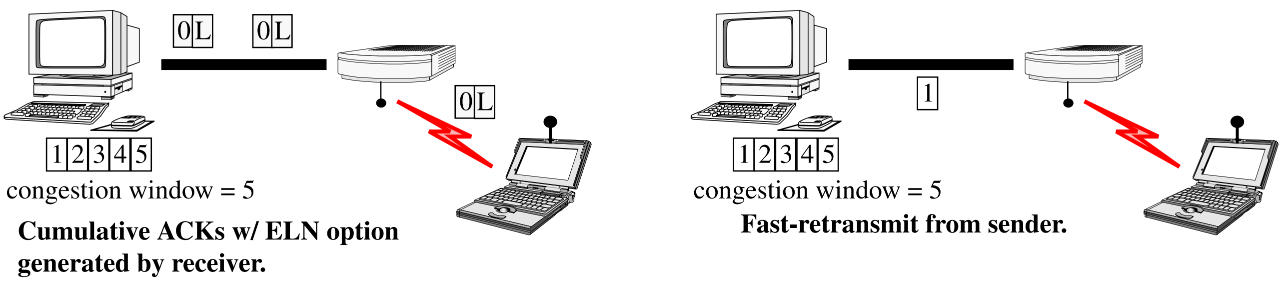


Figure 4. TCP with ELN

The E2E+SACK protocol (Figure 3) adds selective acknowledgments to the standard TCP Reno stack. This allows the sender to handle multiple losses within a window of outstanding data more efficiently. However, the sender still assumes that losses are a result of congestion and invokes congestion control procedures, such as shrinking its congestion window size. This allows us to identify what percentage of the end-to-end performance degradation is associated with standard TCP's handling of error detection and retransmission. We base our selective acknowledgment scheme on the SMART approach [12]. This scheme is well-suited to situations where there is little reordering of packets, which is true for one-hop wireless systems such as ours. Unlike the scheme proposed in [12], we do not use any special techniques to detect the loss of a retransmission. The sender retransmits a packet when it receives a SMART acknowledgment only if the same packet was not retransmitted within the last round-trip time. If no further SMART acknowledgments arrive, the sender falls back to the coarse timeout mechanism to recover from the loss.

The E2E+ELN protocol (Figure 4) adds an Explicit Loss Notification (ELN) option to TCP acknowledgments. When a packet is dropped on the wireless link, future cumulative acknowledgments corresponding to the lost packet are marked to identify that a non-congestion related loss has occurred. Upon receiving this information with duplicate acknowledgments, the sender may perform retransmissions without invoking the associated congestion-control procedures. This option allows us to identify what percentage of the end-to-end performance degradation is associated with TCP's incorrect invocation of congestion control algorithms when it does a fast retransmission of a packet lost on the wireless hop. The E2E+ELN+RXMT protocol is an enhancement of the previous one, where the sender retransmits the packet on receiving the first duplicate ack with the ELN option set, in addition to not shrinking its window size in response to wireless losses.

In practice, it might be difficult to identify which packets are lost due to errors on a lossy link. However, in our experiments we assume perfect knowledge about wireless losses to generate ELN information.

3.2 Link-Layer Schemes

Unlike TCP for the transport layer, there is no de facto standard for link-layer protocols. Existing link-layer protocols choose from techniques such as Stop-and-Wait, Go-Back-N,

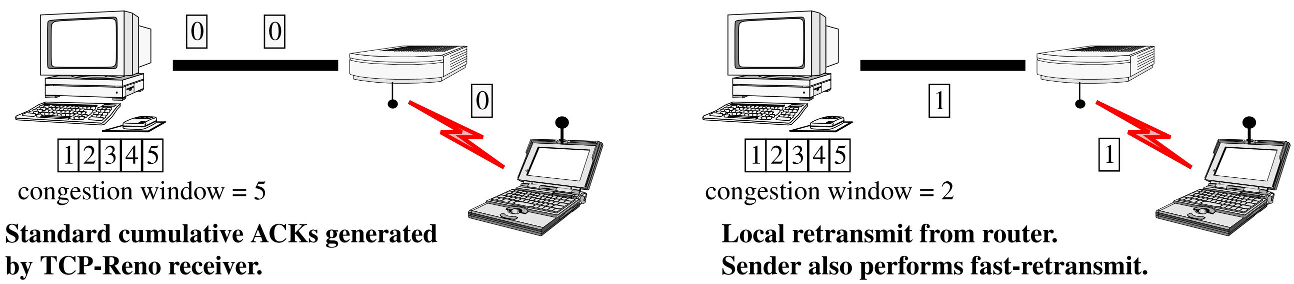


Figure 5. Link-Layer

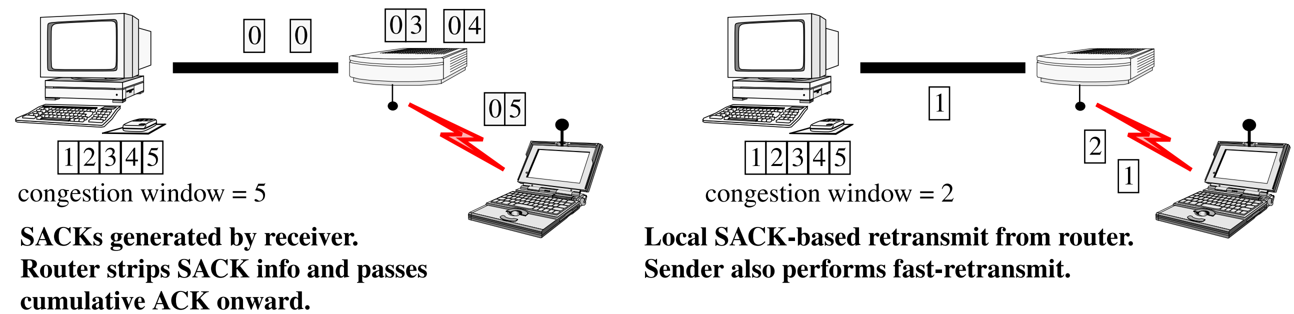


Figure 6. Link-Layer with SACK

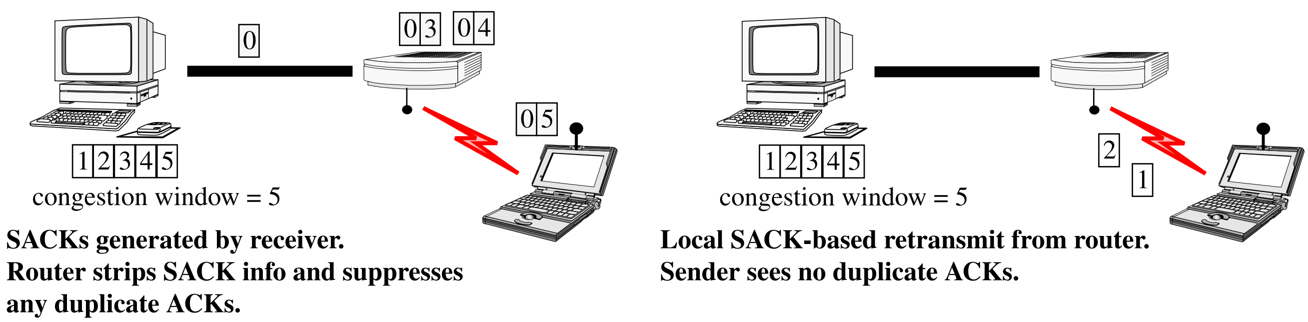


Figure 7. Link-Layer with SACK and TCP awareness

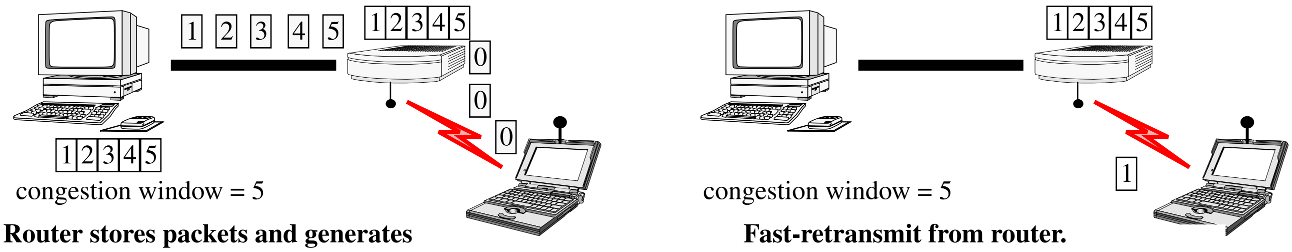
Selective Repeat and Forward Error Correction to provide reliability. Our base link-layer algorithm, called LL (Figure 5), uses cumulative acknowledgments to determine lost packets that are retransmitted locally from the base station to the mobile host. To minimize overhead, our implementation of LL leverages off existing TCP acknowledgments instead of generating its own. Timeout-based retransmissions are done by maintaining a smoothed round-trip time estimate, with a minimum timeout granularity of 200 ms to limit the overhead of processing timer events. This still allows the LL scheme to retransmit packets several times before a TCP-Reno transmitter would time out. LL is equivalent to the snoop agent that does not suppress any duplicate acknowledgments, so it does not attempt in-order delivery of packets across the link (unlike protocols proposed in [10], [17]).

We also investigate a more sophisticated link-layer protocol (LL-SACK) that uses selective retransmission to improve performance. The LL-SACK protocol (Figure 6) performs this by applying a SMART-based acknowledgment scheme to the link layer. Like the LL protocol, LL-SACK uses TCP acknowledgments instead of generating its own and limits its minimum timeout to 200 ms. LL-SACK is equivalent to the snoop agent performing retransmissions based on SACKs and not suppressing duplicate acknowledgments from the TCP source.

We added TCP awareness to both the LL and LL-SACK protocols, resulting in the LL-TCP-AWARE and LL-OPT schemes. The LL-TCP-AWARE protocol is identical to the snoop protocol, while the LL-OPT protocol (Figure 7) uses SMART-based techniques for further optimization using selective repeat. LL-OPT is the optimal link-layer protocol in our experiments in that it performs local retransmissions based on selective acknowledgments and shields the sender from duplicate acknowledgments caused by wireless losses.

3.3 Split-Connection Schemes

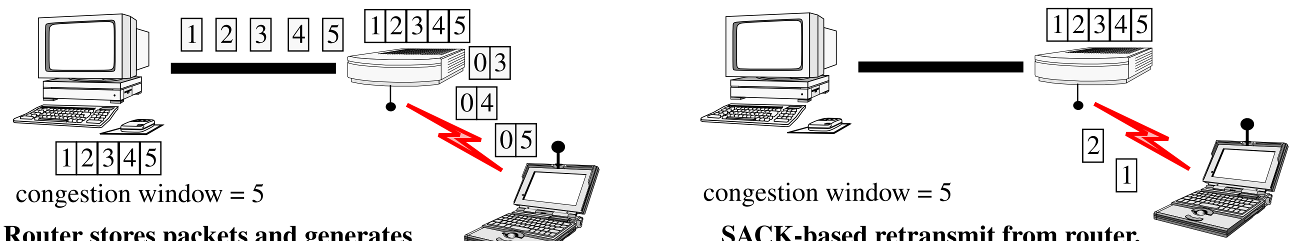
Like I-TCP, our SPLIT scheme (Figure 8) uses an intermediate host to divide a TCP connection into two separate TCP connections. The implementation avoids data copying in the



Router stores packets and generates cumulative ACKs.
Receiver generates cumulative ACKs also.

Fast-retransmit from router.
Sender frees packets from TCP stack.

Figure 8. Split-Connection



Router stores packets and generates cumulative ACKs.
Receiver generates SACKs.

SACK-based retransmit from router.
Sender frees packets from TCP stack.

Figure 9. Split-Connection with SACKs

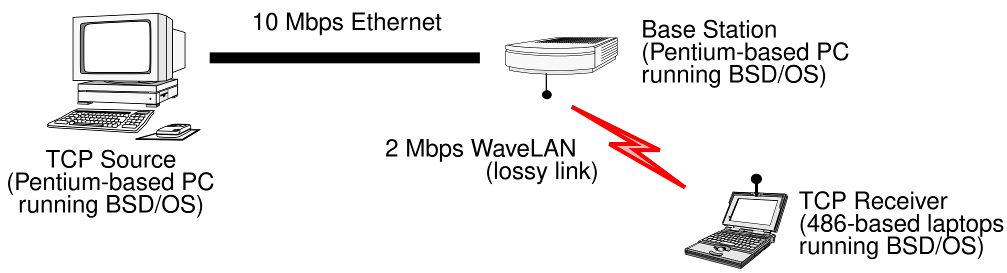


Figure 10. Experimental topology. There were an additional 16 Internet hops between the source and base station during the wide area experiments.

intermediate host by passing the pointers to the same buffer between the two TCP connections. The variations on the SPLIT approach investigate the use of more sophisticated protocols for the connection over the lossy link. The SPLIT-SACK protocol (Figure 9) uses a selective acknowledgment scheme on the wireless connection to perform selective retransmissions. As before, the selective acknowledgments are based on the SMART scheme. There is little chance of reordering of packets over the wireless connection since the intermediate host is close to the final destination.

4. Experimental Results

In this section, we describe the experiments we performed and the results we obtained, including detailed explanations for observed performance. We start by describing the experimental testbed and methodology. We then describe the performance of the various link-layer, end-to-end and split-connection schemes.

4.1 Experimental Methodology

We performed several experiments to determine the performance and efficiency of each of the protocols. The protocols were implemented as a set of modifications to the BSD/OS TCP/IP (Reno) network stack. To ensure a fair basis for comparison, none of the protocols implementations introduce any additional data copying at intermediate points from sender to receiver.

Our experimental testbed consists of IBM ThinkPad laptops and Pentium-based personal computers running BSD/OS 2.0 from BSDI. The machines are interconnected using a 10 Mbps Ethernet and 915 MHz AT&T WaveLANs [20], a shared-medium wireless LAN with a raw signalling bandwidth of 2 Mbps. The network topology for our experiments is shown in Figure 10. The peak throughput for TCP bulk transfers is 1.5 Mbps in the local area testbed and 1.35 Mbps in the wide area testbed in the absence of congestion or wireless losses. These testbed topologies represent typi-

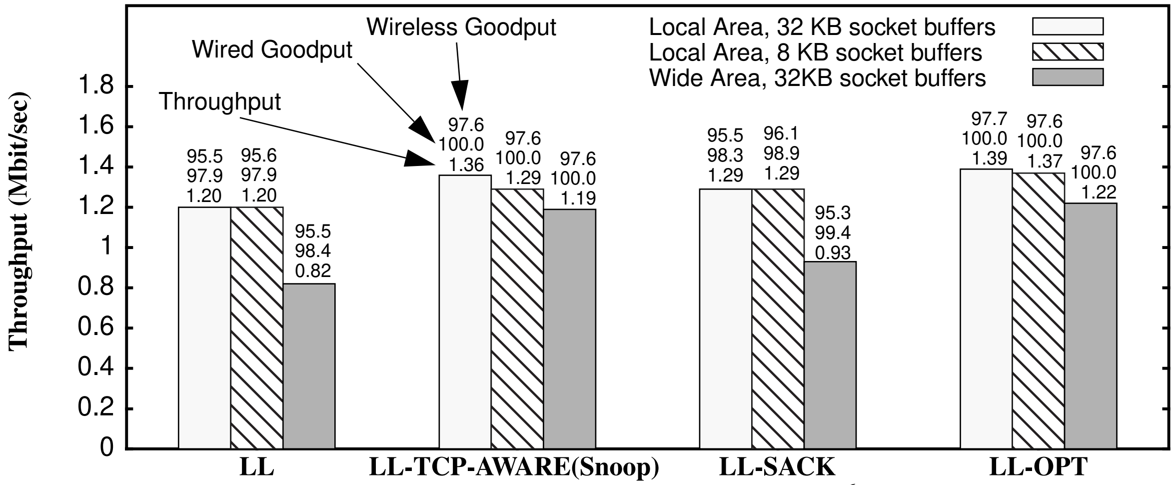


Figure 11. Performance of link-layer protocols: bit-error rate = 1.9x10^{-6} (1 error/65536 bytes).

cal scenarios of wireless links and mobile hosts, such as cellular wireless networks. In addition, our experiments focus on data transfer to the mobile host, which is the common case for mobile applications (e.g., Web accesses).

In order to measure the performance of the protocols under controlled conditions, we generate errors on the lossy link using a Poisson-distributed bit-error model. The receiving entity on the lossy link generates a Poisson distribution for each bit-error rate and changes the TCP checksum of the packet if the error generator determines that the packet should be dropped. Losses are generated in both directions of the wireless channel, so TCP acknowledgments are dropped too, albeit at a lower per-packet rate. For most of the experiments, the TCP data packet size is 1400 bytes and the average error rate is one every 64 KBytes (this corresponds to a bit-error rate of about 1.9x10^{-6}). Note that since a Poisson distribution has a standard deviation equal to its mean, there are several occasions when multiple packets are lost in each window. We also report the results of some burst error situations (Section 4.5), to compare the performance of the different mechanisms in response to burst losses. The choice of the Poisson-distributed error model is motivated by our desire to understand the precise dynamics of each protocol in response to a wireless loss, and is not an attempt to empirically model a wireless channel. While the actual performance numbers will be a strong function of the exact error model, the relative performance is dependent on how the protocol behaves after one or more losses in a single TCP window. Thus, we expect our overall conclusions to be applicable under other patterns of wireless loss as well. Finally, we believe that though wireless errors are generated artificially in our experiments, the use of a real testbed is still valuable in that it introduces realistic effects such as wireless bandwidth limitation, protocol processing delays, and so on.

In our experiments, we attempt to ensure that losses are only due to wireless errors (and not congestion). This allows us to focus on the effectiveness of the mechanisms in handling such losses. The WAN experiments are performed across 16 Internet hops with minimal congestion[2] in order to study the impact of large delay-bandwidth products.

Each run in the experiment consists of an 8 MByte transfer from the source to receiver across the wired net and the WaveLAN link. During each run (repeated multiple times for consistency), we measure the throughput at the receiver in Mbps, and the wired and wireless goodputs as percentages. In addition, all packet transmissions on the Ethernet and WaveLan are recorded for analysis using tcpdump [15], and the sender's TCP code instrumented to record events such as coarse timeouts, retransmission times, duplicate acknowledgment arrivals, congestion window size changes, etc. The rest of this section presents and discusses the results of these experiments.

4.2 Link-Layer Protocols

Traditional link-layer protocols operate independently of the higher-layer protocol, and consequently, do not necessarily shield the sender from the lossy link. This could adversely impact TCP performance for two reasons: (i) competing retransmissions caused by an incompatible setting of timers at the two layers, and (ii) the effect of the link layer protocol on the TCP fast retransmission mechanism. In [5], the effects of the first situation are simulated and analyzed for a TCP-like transport protocol (that closely tracks the round-trip time to set its retransmission timeout) and a reliable link layer protocol. The conclusion was that unless the packet loss rate is high (more than about 10%), competing retransmissions by the link and transport layers often lead to significant performance degradation. However, this is not the dominating effect when link layer schemes, such

2. WAN experiments were performed between 10 pm and 4 am, PST.

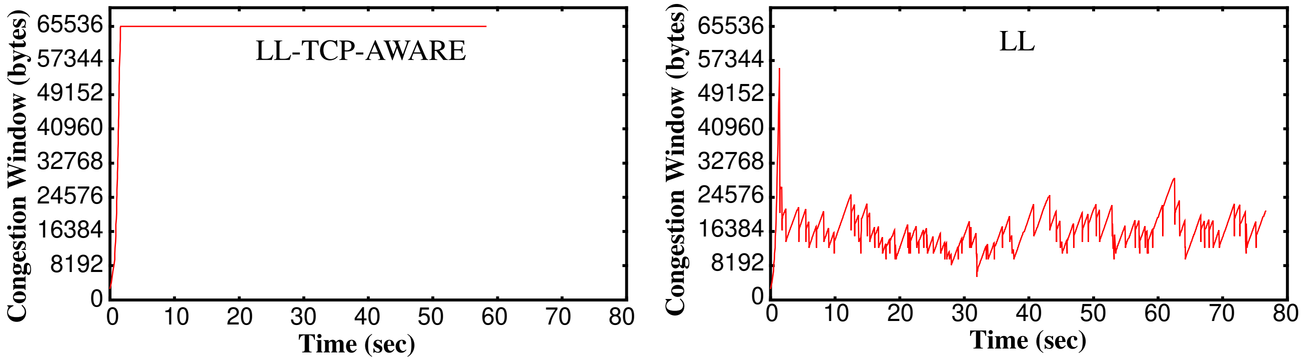


Figure 12. Congestion window size for link-layer protocols in wide area tests.

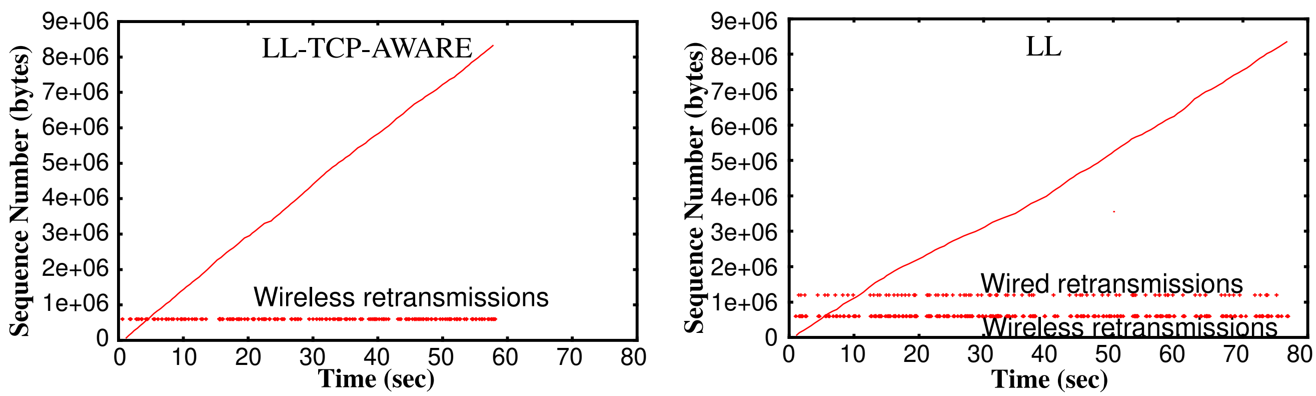


Figure 13. Packet sequence traces for LL-TCP-AWARE and LL. No coarse timeouts occur in either case. For LL-TCP-AWARE, the horizontal row of dots shows the times of wireless link retransmissions. For LL, the top row shows sender fast retransmission times and the bottom row shows both local wireless and sender retransmissions.

as LL, are used with TCP Reno and its variants. These TCP implementations have coarse retransmission timeout granularities that are typically multiples of 500 ms, while link-layer protocols typically have much finer timeout granularities. The real problem is that when packets are lost, link-layer protocols that do not attempt in-order delivery across the link (e.g., LL) cause packets to reach the TCP receiver out-of-order. This leads to the generation of duplicate acknowledgments by the TCP receiver, which causes the sender to invoke fast retransmission and recovery, and can potentially cause degraded throughput and goodput, especially when the delay-bandwidth product is large.

Our results substantiate this claim, as can be seen by comparing the LL and LL-TCP-AWARE results. For a packet size of 1400 bytes and a bit error rate of 1.9×10^{-6} (1/65536 bytes), the packet error rate is about 2.3%. Therefore, an optimal link-layer protocol that recovers from errors locally and does not compete with TCP retransmissions should have a wireless goodput of 97.7% and a wired goodput of 100% in the absence of congestion. In the LAN experiments, the throughput difference between LL and LL-TCP-AWARE is about 10%. However, the LL wireless goodput is only 95.5%, significantly less than LL-TCP-AWARE's wireless goodput of 97.6%, which is close to the maximum achievable goodput. When a loss occurs, the LL protocol performs a local retransmission relatively quickly. However, enough packets are typically in transit to create more than 3 duplicate acknowledgments. These duplicates eventually propagate to the sender and trigger a fast retransmission and the associated congestion control mechanisms. These fast-retransmissions result in reduced goodput; about 90% of the lost packets are retransmitted by both the source (due to fast retransmissions) and the base station.

The effects of this interaction are much more pronounced in the wide area experiments — the throughput difference is about 30% in this case. The cause for the more pronounced deterioration in performance is the higher bandwidth-delay product of the wide-area connection. The LL scheme causes the sender to invoke congestion control procedures often due to duplicate acknowledgments and causes the average window size of the transmitter to be lower than for LL-TCP-AWARE. This is shown in Figure 12, which compares the congestion window size of LL and LL-TCP-AWARE as a function of time. Note that the number of outstanding data bytes in the network is the minimum of the congestion window and the receiver advertised window. This is upper bounded by the receiver's socket buffer size. In the congestion window graphs for each protocol, the receiver socket buffer is 32KB.

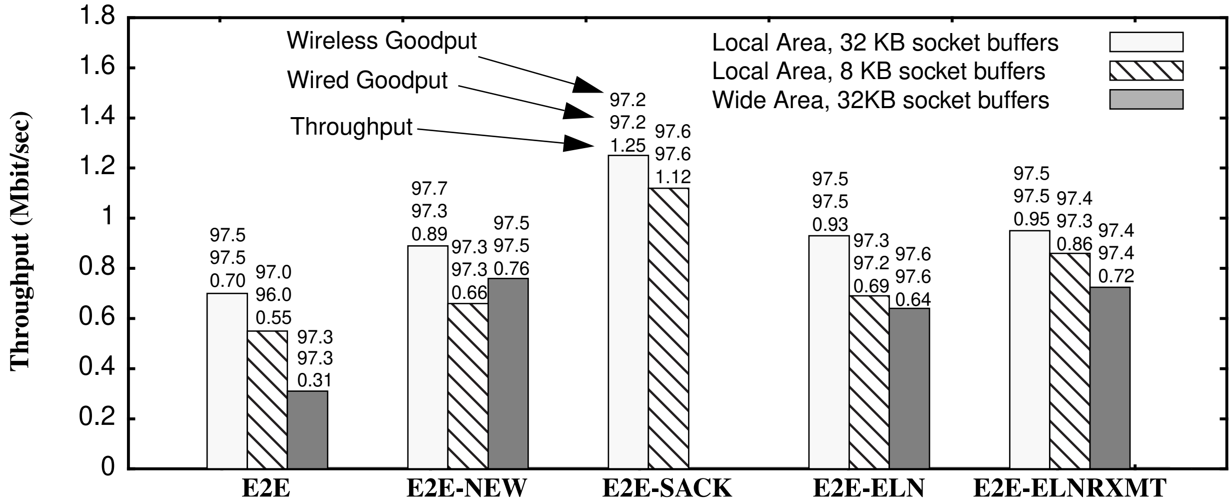


Figure 14. Performance of end-to-end protocols: bit error rate = 1.9×10^{-6} (1 error/65536 bytes).

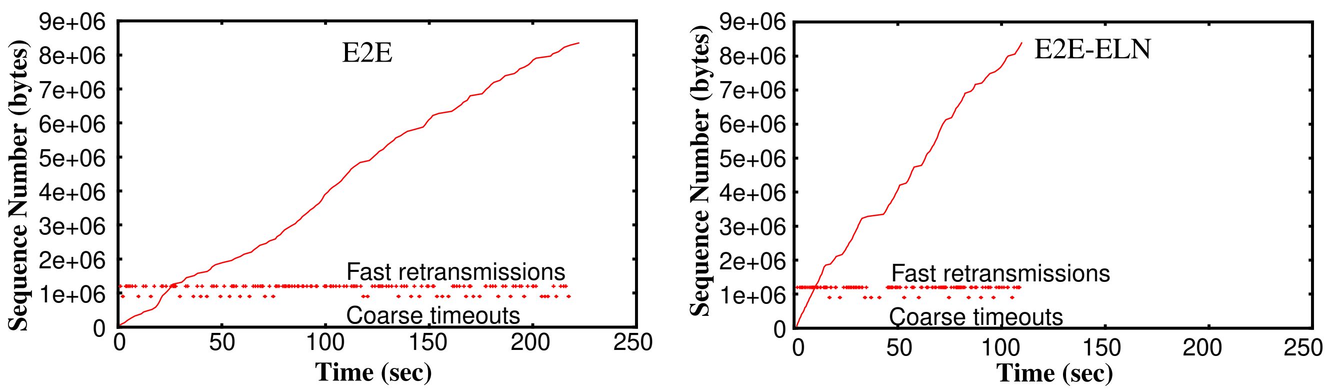


Figure 15. Packet sequence traces for E2E (TCP Reno) and E2E-ELN. The top row of horizontal dots shows the times when fast retransmissions occur; the bottom row shows the coarse timeouts.

In the wide area, the bandwidth-delay product is about 17000 bytes (1.35 Mbps * 100 ms), and the congestion window drops below this value several times during each TCP transfer. On the other hand, the LAN experiments do not suffer from such a large throughput degradation because LL's lower congestion-window size is usually still larger than the delay-bandwidth product of about 1900 bytes (1.5 Mbps * 10 ms). Therefore, the LL scheme can maintain a nearly full "data pipe" between the sender and receiver in the local connection but not in the wide area one. The 10% LAN degradation is almost entirely due to the excessive retransmissions over the wireless link and to the smaller average congestion window size compared to LL-TCP-AWARE. Another important point to note is that LL successfully prevents coarse timeouts from happening at the source, as shown in Figure 13 (there are no points corresponding to coarse timeouts in the figure)

In summary, our results indicate that a simple link-layer retransmission scheme could adversely impact TCP performance. An enhanced link-layer scheme, that uses knowledge of TCP semantics to prevent duplicate acknowledgments caused by wireless losses from reaching the sender, achieves significantly better performance.

4.3 End-To-End Protocols

The performance of the various end-to-end protocols is summarized in Figure 14. The performance of unmodified TCP Reno, the baseline E2E protocol, highlights the problems with TCP over lossy links. With a 2.3% packet loss rate (as explained in Section 4.2), the E2E protocol utilizes less than 50% of the available bandwidth in the local area and less than 25% of the available bandwidth in the wide area experiments. However, all the end-to-end protocols achieve goodputs close to the optimal value of 97.7%. The primary cause for the low bandwidth is the large number of timeout-based retransmissions that occur during the transfer (Figure 15), and the small average window size during the transfer that prevents the "data pipe" from being kept full and reduces the effectiveness of the fast retransmission mechanism (Figure 16).

The modified end-to-end protocols improve throughput by retransmitting packets known to have been lost on the wire-

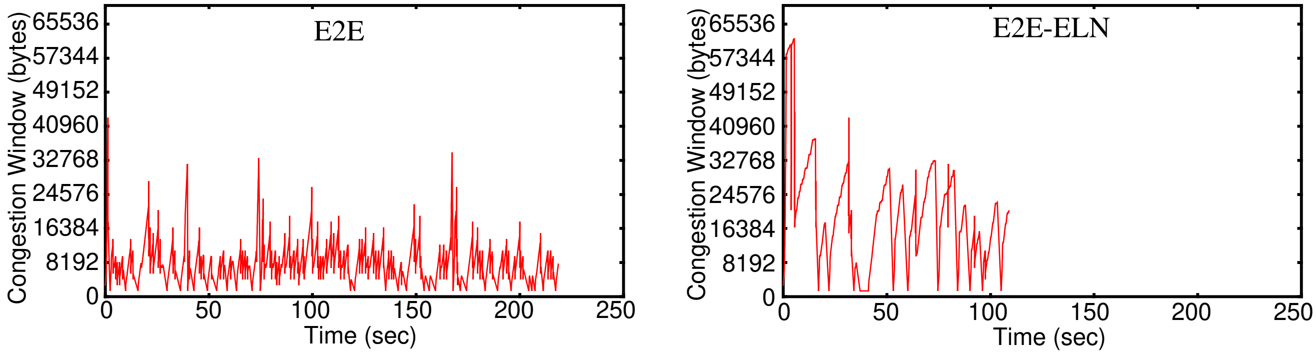


Figure 16. Congestion window size as a function of time for E2E (TCP Reno) and E2E-ELN. This figure clearly shows the utility of ELN preventing rapid fluctuations in the congestion window.

less hop earlier than they would have been by the baseline E2E protocol, and by reducing the fluctuations in window size. The E2E-NEWRENO, E2E-ELN and E2E-SACK each use new TCP options and more sophisticated acknowledgment processing techniques to improve the speed and accuracy of identifying and retransmitting lost packets. E2E-NEWRENO, which uses partial acknowledgment information to recover from multiple losses in a window at the rate of one packet per round-trip time, performs between 10 and 25% better than E2E over a LAN and about 2.3 times better than E2E in the WAN experiments. The performance improvement is a function of the socket buffer size; the larger the buffer size, the better the relative performance. This is because the probability that E2E will suffer a coarse timeout for a loss, but E2E-NEWRENO will not, increases with the number of outstanding packets in the network.

One way of eliminating long delays caused by coarse timeouts is to maintain as large a window size as possible. E2E-NEWRENO remains in fast recovery if the new acknowledgment is only partial, but reduces the window size to half its original value upon the arrival of the first new acknowledgment. The E2E-ELN and E2E-ELN-RXMT protocols use ELN information (Section 3.1) to prevent the sender from reducing the size of the congestion window in response to a wireless loss. Both these schemes perform better than E2E-NEWRENO, and over two times better than E2E. This is a result of the sender's explicit awareness of the wireless link which reduces the number of coarse timeouts (Figure 15), and rapid window size fluctuations (Figure 16). The E2E-ELN-RXMIT protocol performs only slightly better than E2E-ELN when the socket buffer size is 32 KB. This is because there is usually enough data in the pipe to trigger a fast retransmission for E2E-ELN. The performance benefits of E2E-ELN-RXMT are more pronounced when the socket buffer size is smaller, as the numbers for the 8 KB socket buffer size indicate.

Finally, we also experimented with a simple SACK scheme based on a subset of the SMART proposal in the local area. This protocol was the best of the end-to-end protocols in this situation, achieving a throughput of 1.25 Mbps (in contrast, the best local scheme, LL-OPT, obtained a throughput of 1.40 Mbps). Our current implementation of the SACK option based on SMART is not particularly well-suited for the wide area; we are currently in the process of tuning the implementation for this environment. We are also experimenting with the SACK option as defined in the recent IETF Draft over such networks.

In summary, E2E-NEWRENO is better than E2E, especially for large socket buffer sizes. Adding ELN to TCP improves throughput significantly by successfully preventing unnecessary fluctuations in the transmission window. Finally, SACKs provide significant improvement over TCP Reno, but perform about 10-15% worse than the best local schemes in the LAN tests.

4.4 Split-Connection Protocols

The main advantage of the split-connection approaches is that they isolate the TCP source from wireless losses. The TCP sender of the second, wireless connection performs all the retransmissions in response to wireless losses.

Figure 17 shows the throughput and goodput for the split connection approach in the LAN and WAN environments. We report the results for two cases: when the wireless connection uses regular TCP Reno (labeled SPLIT) and when it uses the SMART-based selective acknowledgment scheme described earlier (labeled SPLIT-SACK). We see that the throughput achieved by the SPLIT approach (0.6 Mbps) is quite low, about the same as that for end-to-end TCP Reno (labeled E2E in Figure 14). The reason for this is apparent from Figures 18 and 21, which show the progress of the data transfer and the size of the congestion window for the wired and wireless connections. We see that the wired connection neither has any retransmissions nor any timeouts, resulting in a wired goodput of 100%. However, it (eventually) stalls whenever the sender of the wireless connection experiences a timeout, since the amount of buffer space at the base station (64 KB in our experiments) is bounded. In the WAN case, the throughput of the SPLIT approach is about 0.58 Mbps which is significantly better than the 0.31 Mbps that the E2E approach achieves (Figure 14), but not as good as

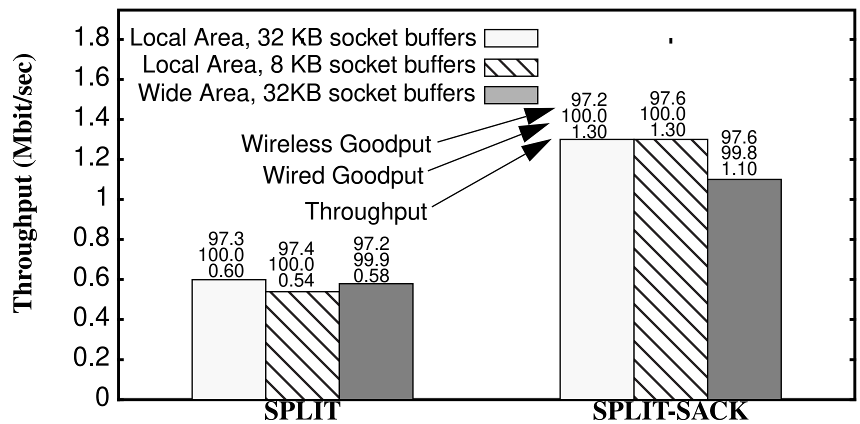


Figure 17. Performance of split-connection protocols: bit error rate = 1.9×10^{-6} (1 error/65536 bytes).

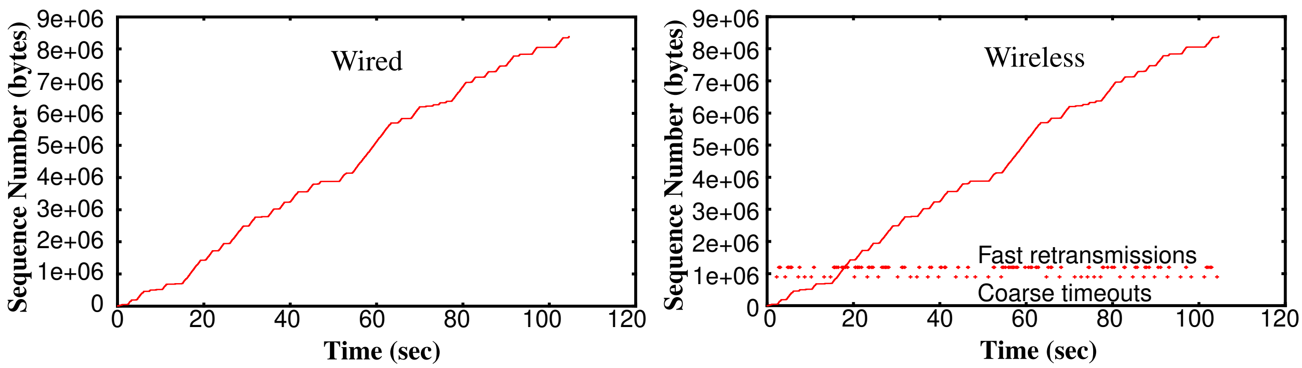


Figure 18. Packet sequence trace for the wired and wireless parts of the SPLIT protocol. The wireless part has two rows of horizontal dots: the top one shows the times of fast retransmissions and the bottom one the times of the time-out-based ones.

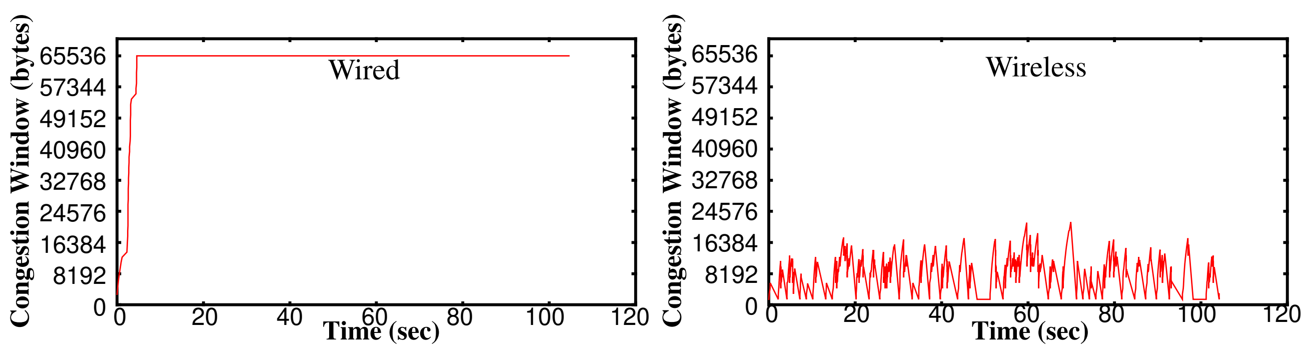


Figure 19. Congestion window sizes as a function of time for the wired and wireless parts of the split TCP connection. The wired sender never sees any losses and maintains a 64 KB congestion window. However, the wireless TCP connection's congestion window fluctuates rapidly.

several other protocols described earlier. The large congestion window size of the wired sender in SPLIT enables a higher bandwidth utilization over the wired network, compared to an end-to-end TCP connection where the congestion window size fluctuates rapidly.

As expected, throughput for the SPLIT-SACK scheme is much higher. It is about 1.3 Mbps in the LAN case and about 1.1 Mbps in the WAN case. The SMART-based selective acknowledgment scheme operating over the wireless link performs very well, especially since no reordering of packets occurs over this hop. However, there are a few times when both the original transmission and the first retransmission of a packet get lost, which sometimes results in a coarse timeouts (as described in Section 3.1). This explains the difference in throughput between the SPLIT-SACK scheme and the LL-OPT scheme (Figure 11).

In summary, while the split-connection approach results in good throughput if the wireless connection uses some special mechanisms, the performance does not exceed that of a well-tuned, TCP-aware link-layer protocol (LL-OPT). Moreover, the link-layer protocol preserves the end-to-end semantics of TCP acknowledgments, unlike the split-connection approach. This demonstrates that the end-to-end connection need not be split at the base station in order to achieve good performance.

4.5 Reaction to Burst Errors

In this section, we report the results of some experiments that illustrate the benefit of selective acknowledgments in handling burst losses. We consider two of the best performing local protocols: LL-TCP-AWARE (Snoop) and LL-OPT (Snoop with SACKs). LL-TCP-AWARE recovers from a single loss by retransmitting the lost packet when two duplicate acknowledgments arrive for it. It also keeps track of the number of expected duplicate acknowledgments and the next expected new acknowledgment after this local retransmission. If this loss is part of a burst, the first new acknowledgment to arrive after the duplicates will be less than the next expected new one; this causes an immediate retransmission of the lost segment. This is similar to the mechanism used by E2E-NEWRENO (Section 3.1). LL-OPT uses the additional useful information provided by the SMART scheme — the sequence number of the segment that caused the duplicate acknowledgment — to accurately determine losses and recover from them.

Table 2 shows the performance of the two protocols for bursts of lengths 2, 4, and 6 packets. These errors are generated once every 64 KBytes of data, and 2, 4, or 6 packets are destroyed in each case. SACKs improve the performance of LL-OPT over LL-TCP-AWARE by up to 30% in the presence of burst errors.

Burst Length	LL-TCP-AWARE (Mbps)	LL-OPT (Mbps)
2	1.25	1.28
4	1.02	1.20
6	0.84	1.10

Table 2. Throughputs of LL-TCP-AWARE and LL-OPT at different burst lengths. This illustrates the benefits of SACKs, even for a high-performance, TCP-aware link protocol.

5. Conclusions

In this paper, we have presented a comparative analysis of several techniques to improve the end-to-end performance of TCP over lossy, wireless hops. We categorize these techniques as end-to-end, link-layer or split-connection based. We use the end-to-end throughput, and the wired and wireless goodputs as metrics for comparison.

Our results lead to the following conclusions:

1. A reliable link-layer protocol that uses knowledge of TCP (LL-TCP-AWARE) to shield the sender from duplicate acknowledgments arising from wireless losses gives a 10-30% higher throughput than one (LL) that operates independently of TCP and does not attempt in-order delivery of packets. Also, the former avoids redundant retransmissions by both the sender and the base station, resulting in a higher goodput. Of the schemes we investigated, the TCP-aware link-layer protocol performs the best.

2. The split-connection approach, with regular TCP used for the wireless hop, shields the sender from wireless losses. However, the sender often stalls due to timeouts on the wireless connection, resulting in poor end-to-end throughput. Using a SMART-based selective acknowledgment mechanism for the wireless hop yields good throughput. However, the throughput is still slightly less than that for a well-tuned link-layer scheme that does not split the connection. This demonstrates that splitting the end-to-end connection is not a requirement for good performance.

3. The SMART-based selective acknowledgment scheme we used is quite effective in dealing with a high packet loss rate when employed over the wireless hop or by a sender in a LAN environment. We are in the process of tuning this scheme for use in a WAN environment. From our results we conclude that selective acknowledgment schemes are very useful in the presence of lossy links, especially when losses occur in bursts.

4. End-to-end schemes, while not as effective as local techniques in handling wireless losses, are promising since significant performance gains can be achieved without any support from intermediate nodes in the network. The explicit loss notification scheme we evaluated resulted in a throughput improvement of more than a factor of two over TCP-Reno, with comparable goodput values.

6. Future Work

Our experiments with simple, SMART-based selective acknowledgments demonstrate the significant benefits of such schemes. We have also implemented the SACK scheme proposed in the Internet Draft in the BSD TCP stack and have configured the sender to react to both SACKs and ELN. We are in the process of evaluating the protocol in the presence of congestion as well as wireless losses.

We are investigating the impact of large variations in connection round-trip times on performance. Such variation is common in networks like the Metricom Ricochet wireless network [16], especially in the presence of bidirectional traffic. We are also investigating the performance of more sophisticated link-layer protocols that attempt in-order delivery of packets to a limited extent.

7. Acknowledgments

We are grateful to Steven McCanne and the anonymous SIGCOMM reviewers for several comments and suggestions that helped improve the quality of this paper.

8. References

[1] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin. AIRMAIL: A Link-Layer Protocol for Wireless Networks. *ACM Wireless Networks*, 1:47–60, February 1995.

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

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

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

[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] K. Fall and S. Floyd. Comparisons of Tahoe, Reno, and Sack TCP. ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z, December 1995.

[7] J. C. Hoe. Start-up Dynamics of TCP's Congestion Control and Avoidance Schemes. Master's thesis, Massachusetts Institute of Technology, 1995.

[8] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM 88*, August 1988.

[9] V. Jacobson and R. T. Braden. *TCP Extensions for Long Delay Paths*. RFC, Oct 1988. RFC 1072.

[10] P. Karn. The Qualcomm CDMA Digital Cellular System. In *Proc. 1993 USENIX Symp. on Mobile and Location-Independent Computing*, pages 35–40, August 1993.

[11] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. *ACM Transactions on Computer Systems*, 9(4):364–373, November 1991.

[12] S. Keshav and S. Morgan. Smart retransmission: Performance with Overload and Random Losses. http://www.cs.att.com/netlib/att/cs/home/keshav/papers/ smart.ps.Z, 1996. Preprint.

[13] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Inc., 1983.

[14] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgments Options. Internet draft, Draft-ietf-tcplw-sack-00.txt, January 1996. Expires July 1996.

[15] S. McCanne and V. Jacobson. The BSD Packet Filter: A New Architecture for User-Level Packet Capture. In *Proc. Winter '93 USENIX Conference*, San Diego, CA, January 1993.

[16] Metricom, Inc. http://www.metricom.com, 1996.

[17] 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.

[18] J. B. Postel. *Transmission Control Protocol*. RFC, Information Sciences Institute, Marina del Rey, CA, September 1981. RFC 793.

[19] W. R. Stevens. *TCP/IP Illustrated, Volume 1*. Addison-Wesley, Reading, MA, Nov 1994.

[20] *WaveLAN: PC/AT Card Installation and Operation*, 1994.

[21] 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.