Appeared in IEEE/ACM Transactions on Networking, Dec.1997. This is a much-extended and revised version of a paper that appeared at ACM SIGCOMM, 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, Department of EECS, 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 losses due to 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. We classify these schemes into three broad categories: end-to-end protocols, where loss recovery is performed by the sender; 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 that is TCP-aware 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 [24, 26] 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 congestion losses. 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 packet losses by dropping its transmission (congestion) window size before retransmitting packets, initiating congestion control or avoidance mechanisms (e.g., slow start [13]) and backing off its retransmission timer (Karn's Algorithm [16]). 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 hence, sub-optimal 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 [8].

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 [3, 7, 28]. 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 different approaches to improving TCP performance in such lossy systems. The first approach hides any non-congestion-related losses from 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.

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).

Examples of this approach include wireless links with reliable link-layer protocols such as AIRMAIL [1], split connection approaches such as Indirect-TCP [3], and TCP-aware link-layer schemes such as the snoop protocol [7]. 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 standard 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., 18] 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 [10], 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 [7].

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. Examples of such links include high-speed modems and cable modems.

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. Section 5 discusses some miscellaneous issues related to handoffs, ELN implementation and selective acknowledgments. We present our conclusions in Section 6, and mention some future work in Section 7.

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 [15] and TDMA [22] — 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 eventual error recovery to the (reliable) transport layer. Other protocols like the AIRMAIL protocol [1] employ 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 and 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, as described in Section 1. We investigate this in detail in our experiments.

- **Split connection protocols [3, 28]:** Split connection protocols split 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. Over the wireless hop, a specialized protocol tuned to the wireless environment may be used. In [28], the authors propose two protocols — one in which the wireless hop uses TCP, and another in which the wireless hop uses a selective repeat protocol (SRP) on top of UDP. They study the impact of handoffs on performance and conclude that they obtain no significant advantage by using SRP instead of TCP over the wireless connection in their experiments. However, our experiments demonstrate benefits in using a simple selective acknowledgment scheme with TCP over the wireless connection.

  Indirect-TCP [Bakre95] is a split-connection solution that uses standard TCP for its connection over the wireless link. Like other split-connection proposals, it 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 split connections 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 split-connection protocols maintain a significant amount of state at the base station per TCP connection, handoff procedures tend to be complicated and slow. Section 5.1 discusses some issues related to cellular handoffs and TCP performance.

- **The Snoop Protocol [7]:** 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., 11] 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 [14]. 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 relevant proposals are the

Name	Category	Special Mechanisms
E2E	end-to-end	standard TCP-Reno
E2E-NEWRENO	end-to-end	TCP-NewReno
E2E-SMART	end-to-end	SMART-based selective acks
E2E-IETF-SACK	end-to-end	IETF selective acks
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-SMART	link-layer	SMART-based selective acks
LL-SMART-TCP-AWARE	link-layer	SMART and duplicate ack suppression
SPLIT	split-connection	none
SPLIT-SMART	split-connection	SMART-based wireless connection

Table 1. Summary of protocols studied in this paper.

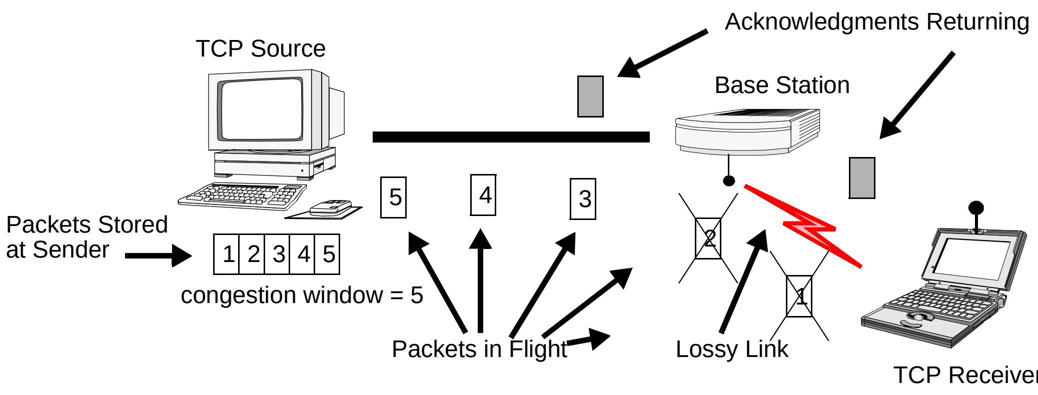


Figure 1. A typical loss situation

recent RFC on TCP SACKs [19] and the SMART scheme [17].

The SACK RFC proposes that each acknowledgment contain information about up to three non-contiguous blocks of data that have been received successfully by the receiver. Each block of data is described by its starting and ending sequence number. Due to the limited number of blocks, it is best to inform the sender about the most recent blocks received. The RFC does not specify the sender behavior, except to require that standard TCP congestion control actions be performed when losses occur.

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 RFC). The sender uses this information to create a bitmask of packets that have been delivered successfully to the receiver. When the sender detects a gap in the bitmask, it immediately assumes that the missing packets have been lost without considering the possibility that they simply may have been reordered. Thus 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 between them. Figure 1 shows a typical loss situation over the wireless 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. As described in the rest of this section, each protocol reacts to these losses in different ways and generates messages that result in loss recovery. Although this figure only shows data packets being lost, our experiments have wireless errors in both directions.

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 [26]. We call this the E2E protocol, and use it as the standard basis for performance comparison.

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. Such partial acknowledgements are indicative of multiple packet losses within the original window of data. Remaining in fast recovery mode enables the connection to recover from losses at the rate of one segment per round trip time, rather than stall until a coarse timeout as TCP-Reno often would [9, 12].

The E2E-SMART and E2E-IETF-SACK protocols add SMART-based and IETF selective acknowledgments respectively 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, 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 used the SMART-based scheme [17] only for the LAN experiments. 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 [17], 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. We used the IETF selective acknowledgement scheme both for the LAN and the WAN experiments. Our implementation is based on the RFC and takes appropriate congestion control actions upon receiving SACK information [4].

The E2E-ELN protocol 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 acknowledgement with the ELN option set (as opposed to the third duplicate acknowledgement in the case of TCP Reno), 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 sufficient knowledge at the receiver about wireless losses to generate ELN information. We describe some possible implementation policies and strategies for the ELN mechanism in Section 5.2.

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, Selective Repeat and Forward Error Correction to provide reliability. Our base link-layer algorithm, called LL, 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 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 typical TCP Reno transmitter would time out. LL is equivalent to the snoop agent that does not suppress any duplicate acknowledgments, and does not attempt in-order delivery of packets across the link (unlike protocols proposed in [15], [22]).

While the use of TCP acknowledgments by our LL protocol renders it atypical of traditional ARQ protocols, we believe that it still preserves the key feature of such protocols: the ability to retransmit packets locally, independently of and on a much faster time scale than TCP. Therefore, we expect the qualitative aspects of our results to be applicable to general link-layer protocols.

We also investigated a more sophisticated link-layer protocol (LL-SMART) that uses selective retransmissions to improve performance. The LL-SMART protocol performs this by applying a SMART-based acknowledgment scheme at the link layer. Like the LL protocol, LL-SMART uses TCP acknowledgments instead of generating its own and limits its minimum timeout to 200 ms. LL-SMART is equivalent to the snoop agent performing retransmissions based on selective acknowledgements but not suppressing duplicate acknowledgments at the base station.

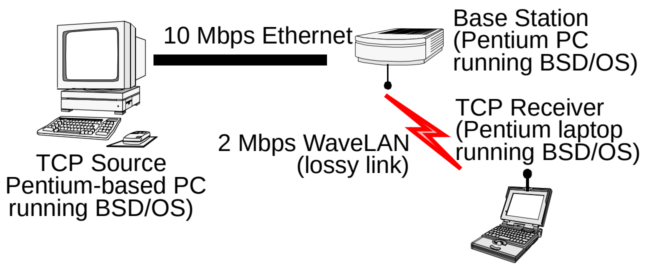

10 Mbps Ethernet
Base Station (Pentium PC running BSD/OS)
TCP Receiver (Pentium laptop running BSD/OS)
2 Mbps WaveLAN (lossy link)
TCP Source Pentium-based PC running BSD/OS)

Figure 2. Experimental topology. There were an additional 16 Internet hops between the source and base station during the WAN experiments.

We added TCP awareness to both the LL and LL-SMART protocols, resulting in the LL-TCP-AWARE and LL-SMART-TCP-AWARE schemes. The LL-TCP-AWARE protocol is identical to the snoop protocol, while the LL-SMART-TCP-AWARE protocol uses SMART-based techniques for further optimization using selective repeat. LL-SMART-TCP-AWARE is the best link-layer protocol in our experiments — 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 uses an intermediate host to divide a TCP connection into two separate TCP connections. The implementation avoids data copying in the intermediate host by passing the pointers to the same buffer between the two TCP connections. A variant of the SPLIT approach we investigated, SPLIT-SMART, uses a SMART-based selective acknowledgment scheme on the wireless connection to perform selective retransmissions. There is little chance of reordering of packets over the wireless connection since the intermediate host is only one hop away from 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 2.1 from BSDI. The machines are interconnected using a 10 Mbps Ethernet and 915 MHz AT&T WaveLANs [27], a shared-medium wireless LAN with a raw signalling bandwidth of 2 Mbps. The network topology for our experiments is shown in Figure 2. 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 typical 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 an exponentially distributed bit-error model. The receiving entity on the lossy link generates an exponential 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. The TCP data packet size in our experiments is 1400 bytes. We first measure and analyze the performance of the various protocols at an average error rate of one every 64 KBytes (this corresponds to a bit-error rate of about 1.9×10^{-6}). Note that since the exponential distribution has a standard deviation equal to its mean, there are several occasions when multiple packets are lost in close succession. We then report the results of some burst error situations, where between two and six packets are dropped in every burst (Section 4.5). Finally, we investigate the performance of many of these protocols across a range of error rates from one every 16 KB to one every 256 KB. The choice of the exponentially 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 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, media access contention, protocol processing delays, etc., which are hard to model realistically in a simulation.

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

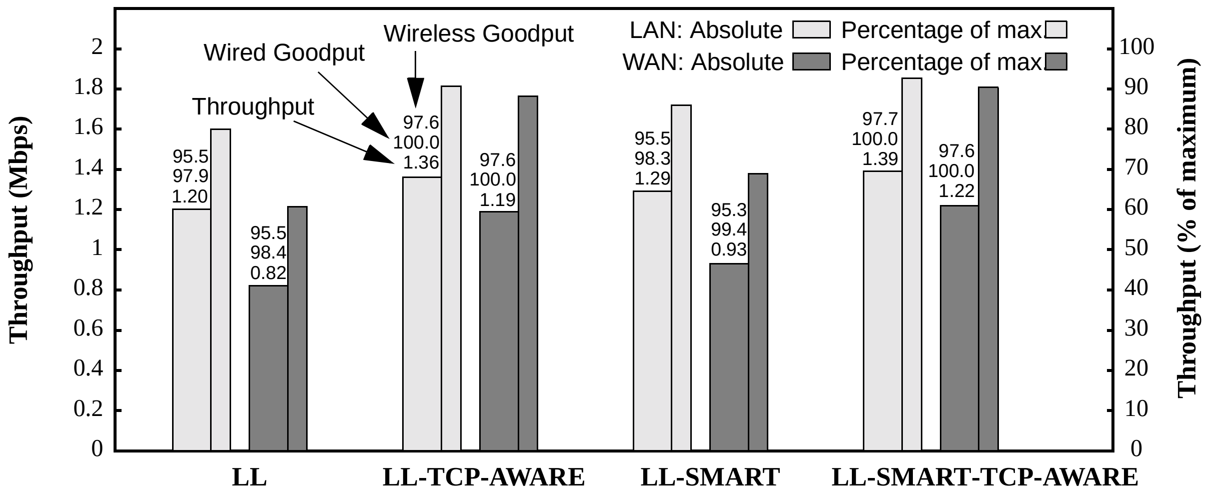


Figure 3. Performance of link-layer protocols: bit-error rate = 1.9x10^{-6} (1 error/65536 bytes), socket buffer size = 32 KB. For each case there are two bars: the thick one corresponds to the scale on the left and denotes the throughput in Mbps; the thin one corresponds to the scale on the right and shows the throughput as a percentage of the maximum, i.e. in the absence of wireless errors (1.5 Mbps in the LAN environment and 1.35 Mbps in the WAN environment).

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. We chose this rather long transfer size in order to limit the impact of transient behavior at the start of a TCP connection. During each run, 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 [20], 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. In spite of local retransmissions, TCP performance could be poor for two reasons: (i) competing retransmissions caused by an incompatible setting of timers at the two layers, and (ii) unnecessary invocations of the TCP fast retransmission mechanism due to out-of-order delivery of data. In [10], 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 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. This 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 (Figure 3 and Table 2). For a packet size of 1400 bytes, a bit error rate of 1.9x10^{-6} (1/65536 bytes) translates to a packet error rate of about 2.2 to 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

2. WAN experiments across the US were performed between 10 pm and 4 am, PST and we verified that no congestion losses occurred in the runs reported.

	LL	LL-TCP-AWARE	LL-SMART	LL-SMART-TCP-AWARE
LAN (8 KB)	1.20 (95.6%,97.9%)	1.29 (97.6%,100%)	1.29 (96.1%,98.9%)	1.37 (97.6%,100%)
LAN (32 KB)	1.20 (95.5%,97.9%)	1.36 (97.6%,100%)	1.29 (95.5%,98.3%)	1.39 (97.7%,100%)
WAN (32 KB)	0.82 (95.5%,98.4%)	1.19 (97.6%,100%)	0.93 (95.3%,99.4%)	1.22 (97.6%,100%)

Table 2. This table summarizes the results for the link-layer schemes for an average error rate of one every 65536 bytes of data. Each entry is of the form: throughput (wireless goodput, wired goodput). Throughput is measured in Mbps. Goodput is expressed as a percentage.

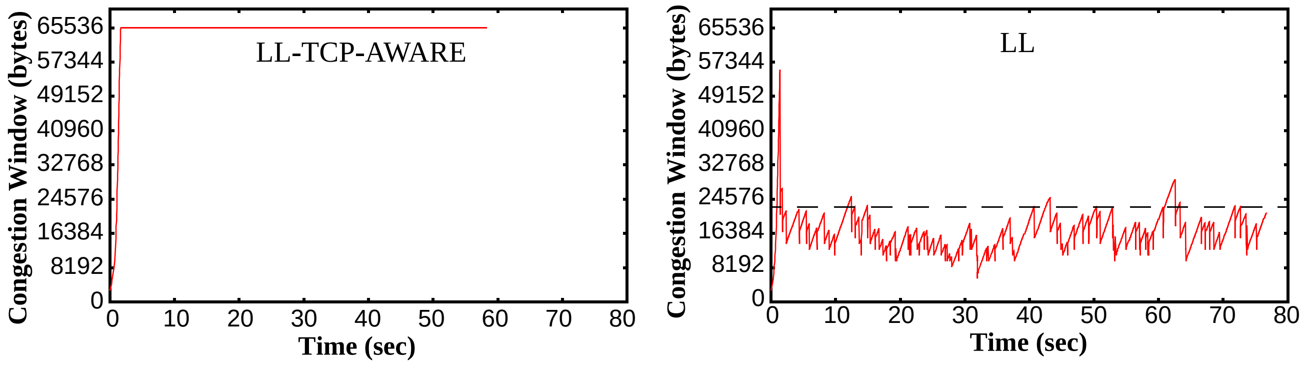


Figure 4. Congestion window size for link-layer protocols in wide area tests. The horizontal dashed line in the LL graph shows the 23000 byte WAN bandwidth-delay product.

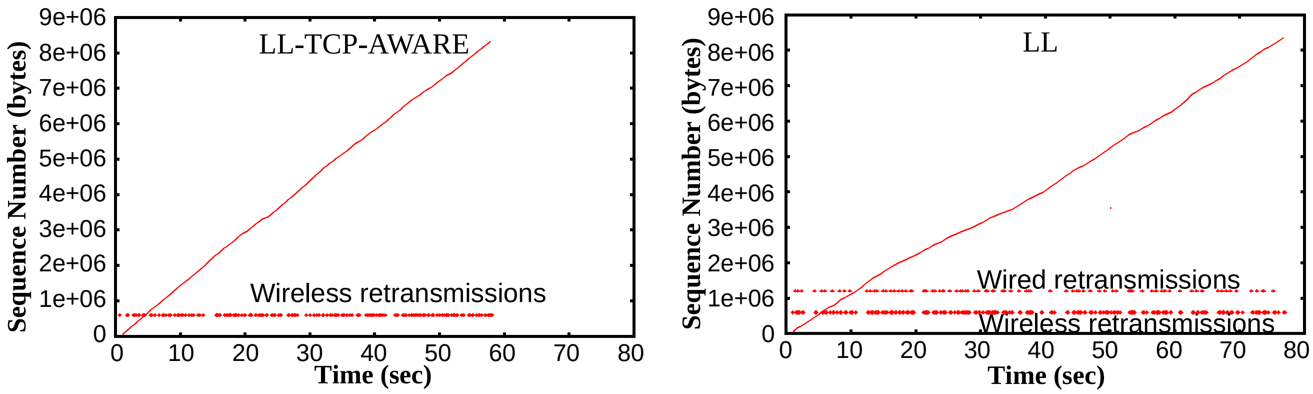


Figure 5. Packet sequence traces for LL-TCP-AWARE and LL. No coarse timeouts occur in either case. For LL-TCP-AWARE, the horizontal row of dots shows the times of wireless link retransmissions. For LL, the top row shows sender fast retransmission times and the bottom row shows both local wireless and sender retransmissions.

reduced goodput; about 90% of the lost packets are retransmitted by both the source 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 4, 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 bounded by the receiver's socket buffer size. In the congestion window graphs for each protocol, the receiver socket buffer is 32KB.

In the wide area, the bandwidth-delay product is about 23000 bytes (1.35 Mbps * 135 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 connection's 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

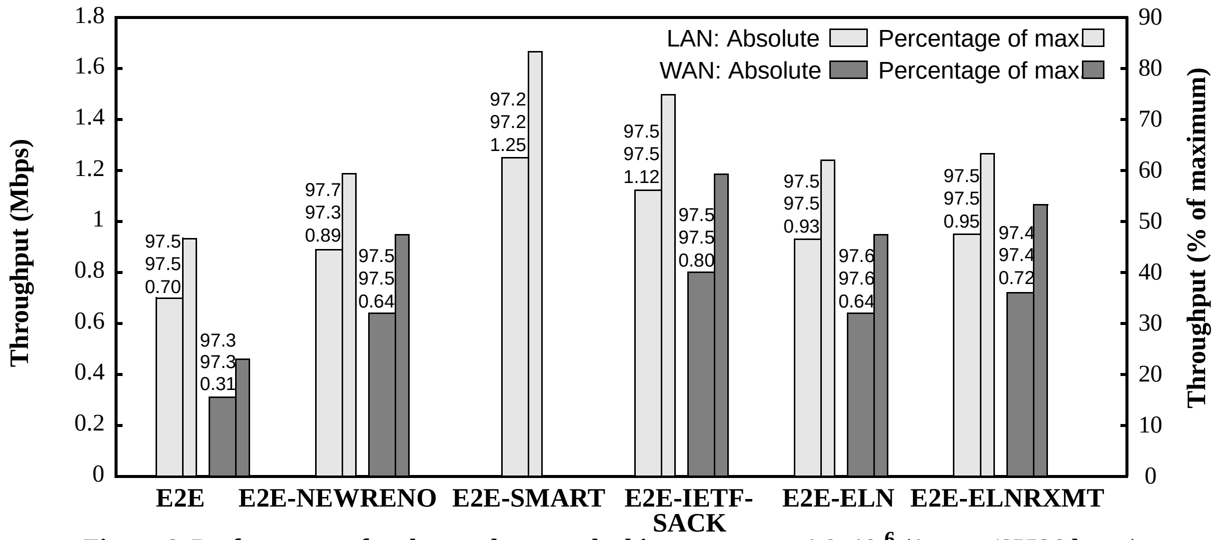


Figure 6. Performance of end-to-end protocols: bit error rate = 1.9x10^{-6} (1 error/65536 bytes).

	E2E	E2E-NEWRENO	E2E-SMART	E2E-IETF-SACK	E2E-ELN	E2E-ELN-RXMT
LAN (8 KB)	0.55 (97.0,96.0)	0.66 (97.3,97.3)	1.12 (97.6,97.6)	0.68 (97.3,97.3)	0.69 (97.3,97.2)	0.86 (97.4,97.3)
LAN (32 KB)	0.70 (97.5,97.5)	0.89 (97.7,97.3)	1.25 (97.2,97.2)	1.12 (97.5,97.5)	0.93 (97.5,97.5)	0.95 (97.5,97.5)
WAN (32 KB)	0.31 (97.3,97.3)	0.64 (97.5,97.5)	N.A.	0.80 (97.5,97.5)	0.64 (97.6,97.6)	0.72 (97.4,97.4)

Table 3. This table summarizes the results for the end-to-end schemes for an average error rate of one every 65536 bytes of data. The numbers in the cells follow the same convention as in Table 2.

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. Figure 5 shows the sequence traces of TCP transfers for LL-TCP-AWARE and LL.

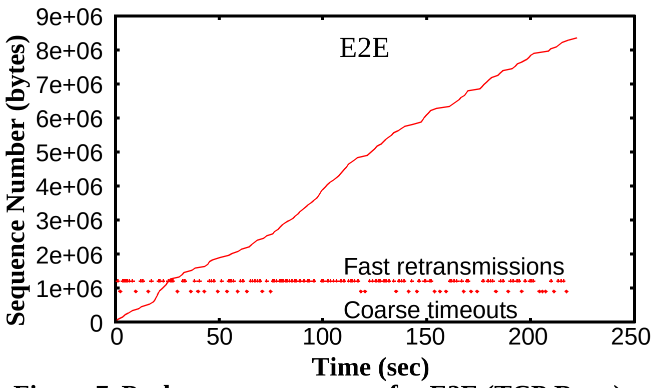
In summary, our results indicate that a simple link-layer retransmission scheme does not entirely avoid the adverse effects of TCP fast retransmissions and the consequent performance degradation. An enhanced link-layer scheme that uses knowledge of TCP semantics to prevent duplicate acknowledgments caused by wireless losses from reaching the sender and locally retransmits packets achieves significantly better performance.

4.3 End-To-End Protocols

The performance of the various end-to-end protocols is summarized in Figure 6 and Table 3. The performance of TCP Reno, the baseline E2E protocol, highlights the problems with TCP over lossy links. At a 2.3% packet loss rate (as explained in Section 4.2), the E2E protocol achieves a throughput of less than 50% of the maximum (i.e., throughput in the absence of wireless losses) in the local-area and less than 25% of the maximum in the wide-area experiments. However, all the end-to-end protocols achieve goodputs close to the optimal value of 97.7%. The primary reason for the low throughput is the large number of timeouts that occur during the transfer (Figure 7). The resulting average window size during the transfer is small, preventing the “data pipe” from being kept full and reducing the effectiveness of the fast retransmission mechanism (Figure 8).

The modified end-to-end protocols improve throughput by retransmitting packets known to have been lost on the wireless 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, E2E-SMART and E2E-IETF-SACK protocols each use new TCP options and more sophisticated acknowledgment processing techniques to improve the speed and accuracy of identifying and retransmitting lost packets, as well as by recovering from multiple losses in a single transmission window without timing out. The remainder of this section discusses the benefits of three techniques — partial acknowledgments, explicit loss notifications, and selective acknowledgments.

Partial acknowledgments: 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 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



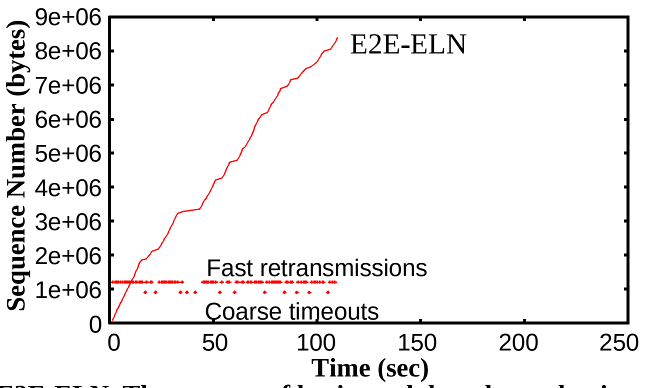
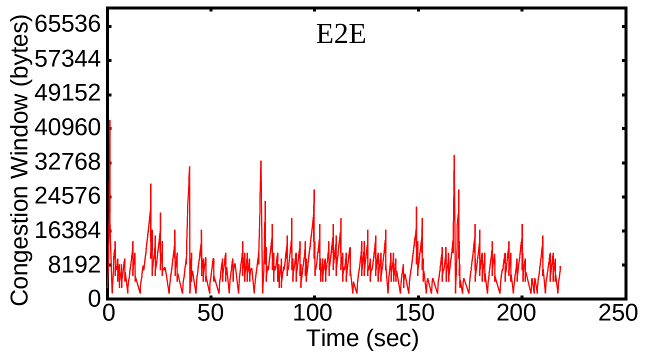


Figure 7. Packet sequence traces for E2E (TCP Reno) and E2E-ELN. The top row of horizontal dots shows the times when fast retransmissions occur; the bottom row shows the coarse timeouts.



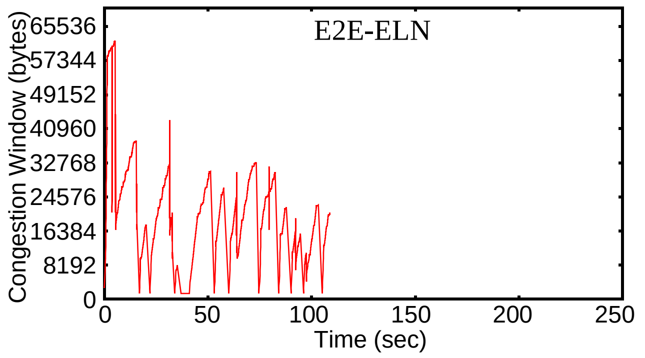


Figure 8. Congestion window size as a function of time for E2E (TCP Reno) and E2E-ELN. This figure clearly shows the utility of ELN in preventing rapid fluctuations, thereby maintaining a larger average congestion window size.

the relative performance. This is because in situations that E2E suffers a coarse timeout for a loss, the probability that E2E-NEWRENO does not, increases with the number of outstanding packets in the network.

Explicit Loss Notification: One way of eliminating the 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 7) and rapid window size fluctuations (Figure 8). The E2E-ELN-RXMT 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 (Table 3). This is because E2E-ELN-RXMT does not wait for three duplicate acknowledgments before retransmitting a packet, if it has ELN information for it. The maximum socket buffer size of 8 KB limits the number of unacknowledged packets to a small number at any point in time, which reduces the probability of three duplicate acknowledgments arriving after a loss and triggering a fast retransmission.

Despite explicit awareness of wireless losses, timeouts sometimes occur in the ELN-based protocols. This is a result of our implementation of the ELN protocol, which does not convey information about multiple wireless-related losses to the sender. Since it is coupled with only cumulative acknowledgments, the sender is unaware of the occurrence of multiple wireless-related losses in a window; we plan to couple SACKs and ELN together in future work. Section 5.2 discusses some possible implementation strategies and policies for ELN.

Selective acknowledgments: We experimented with two different SACK schemes. In the LAN case, we used a simple SACK scheme based on a subset of the SMART proposal. 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-SMART-TCP-AWARE, obtained a throughput of 1.39 Mbps).

In the WAN case, we based our SACK implementation [4] on RFC 2018. For the exponentially-distributed loss pattern

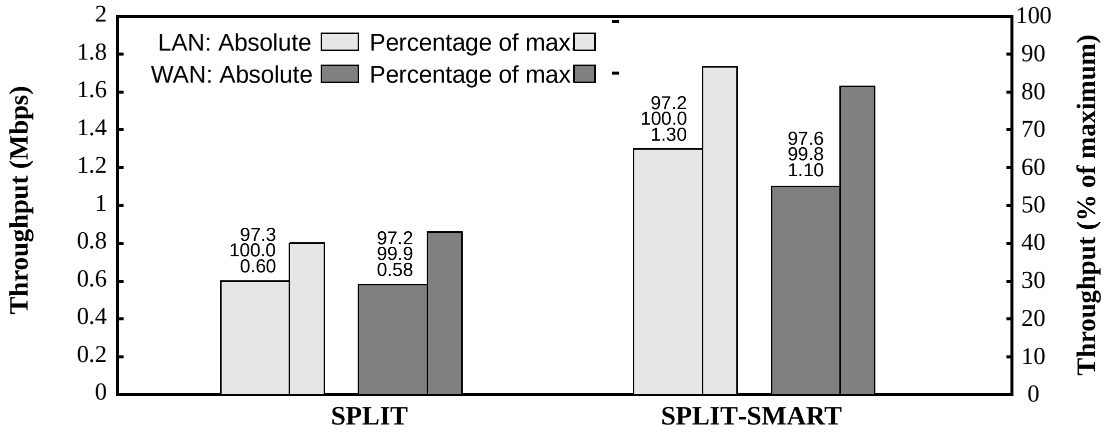


Figure 9. Performance of split-connection protocols: bit error rate = 1.9x10^{-6} (1 error/65536 bytes).

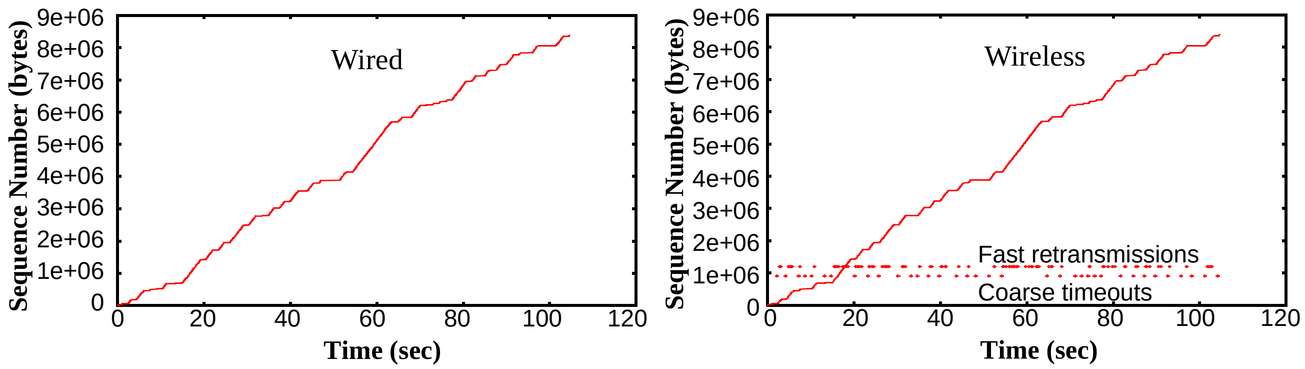


Figure 10. Packet sequence trace for the wired and wireless parts of the SPLIT protocol. The wireless part has two rows of horizontal dots: the top one shows the times of fast retransmissions and the bottom one the times of the time-out-based ones.

we used, the throughput was about 0.8 Mbps, significantly higher than the 0.31 Mbps throughput of TCP Reno. However, this is still about 35% worse than LL-OPT. Even though SACKs allow the sender to often recover from multiple losses without timing out, the sender's congestion window decreases every time there is a packet dropped on the wireless link, causing it to remain small.

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 link-layer schemes in the LAN experiments, and about 35% worse in the WAN experiments. These results suggest that an end-to-end protocol that has both ELN and SACKs will result in good performance, and is an area of current work.

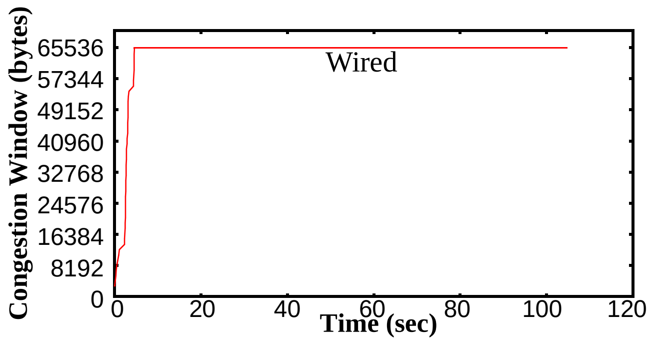
4.4 Split-Connection Protocols

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.

	SPLIT	SPLIT-SMART
LAN (8 KB)	0.54 (97.4%,100%)	1.30 (97.6%,100%)
LAN (32 KB)	0.60 (97.3%,100%)	1.30 (97.2%,100%)
WAN (32 KB)	0.58 (97.2%,100%)	1.10 (97.6%,100%)

Table 4. Summary of results for the split-connection schemes at an average error rate of 1 every 64 KB.

Figure 9 and Table 4 show 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 TCP Reno (labeled SPLIT) and when it uses the SMART-based selective acknowledgment scheme described earlier (labeled SPLIT-SMART). 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 6). The reason for this is apparent from Figures 10 and 11, 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



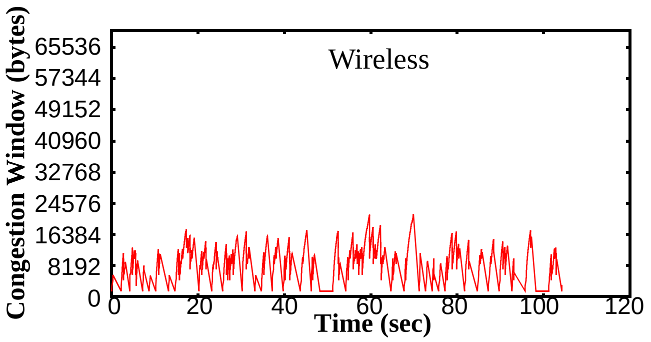


Figure 11. Congestion window sizes as a function of time for the wired and wireless parts of the split TCP connection. The wired sender never sees any losses and maintains a 64 KB congestion window. However, the wireless TCP connection's congestion window fluctuates rapidly.

space at the base station (64 KB in our experiments) is bounded[3]. In the WAN case, the throughput of the SPLIT approach is about 0.58 Mbps which is better than the 0.31 Mbps that the E2E approach achieves (Figure 6), but not as good as 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, the throughput for the SPLIT-SMART 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 timeout (as described in Section 3.1). This explains the difference in throughput between the SPLIT-SMART scheme and the LL-SMART-TCP-AWARE scheme (Figure 3).

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

3. A larger buffer at the base station will not necessarily improve performance for two reasons: (1) we measure performance in terms of receiver throughput, which is limited by the small congestion window size of the wireless connection, and (2) a long enough transfer will still fill up the buffer.

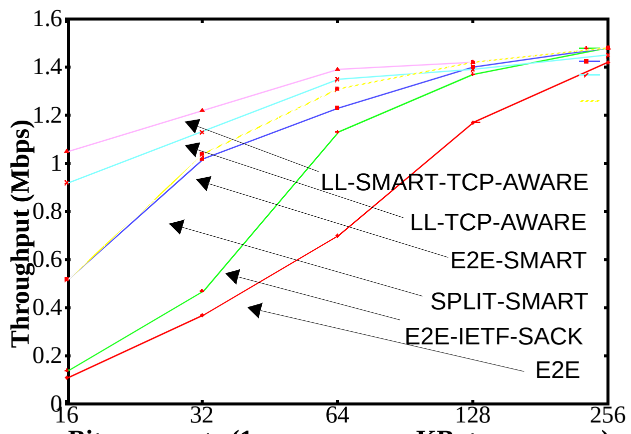
Burst Length	LL-TCP-AWARE (Mbps)	LL-SMART-TCP-AWARE (Mbps)
2	1.25	1.28
4	1.02	1.20
6	0.84	1.10

Table 5. Throughputs of LL-TCP-AWARE and LL-SMART-TCP-AWARE at different burst lengths. This illustrates the benefits of SACKs, even for a high-performance, TCP-aware link protocol.

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-SMART-TCP-AWARE (Snoop with SMART-based selective acknowledgments). 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-SMART-TCP-AWARE 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 5 shows the performance of the two protocols for bursts of lengths 2, 4, and 6 packets. These errors are generated at an average rate of one every 64 KBytes of data, and 2, 4, or 6 packets are destroyed in each case. Selective acknowledgments improve the performance of LL-SMART-TCP-AWARE over LL-TCP-AWARE by up to 30% in the presence of burst errors. While this is a fairly



Bit-error rate (1 error every *x* KBytes, average)

Figure 12. Performance of six protocols (LAN case) across a range of bit-error rates, ranging from 1 error every 16 KB to 1 every 256 KB shown on a log-scale.

simplistic burst-error model, it does illustrate the problems caused by the loss of multiple packets in succession. We are in the process of experimenting with a *temporal* burst-loss model based on average lengths of fades and other causes of wireless losses. The parameters of this model are derived from a trace-based modeling and characterization of the WaveLAN network [23].

4.6 Performance at Different Error Rates

In this section, we present the results of several experiments performed across a range of bit-error rates, for some of the protocols described earlier — E2E (the baseline case), LL-TCP-AWARE, LL-SMART-TCP-AWARE, E2E-SMART, E2E-IETF-SACK, and SPLIT-SMART. We chose the best performing protocols from each category, as well as some other protocols (e.g., E2E-IETF-SACK) to illustrate some interesting effects.

Figure 12 shows the performance of these protocols for an 8 MByte end-to-end transfer in a LAN environment, across exponentially distributed error rates ranging from 1 error every 16 KB to 1 error every 256 KB, in increasing powers of two. We find that the overall qualitative results and conclusions are similar to those presented earlier for the 64 KB error rate. At low error rates (128 KB and 256 KB points in the graph), all the protocols shown perform almost equally well in improving TCP performance. At the 16 KB error rate, the performance of the TCP-aware link-layer schemes is about 1.75-2 times better than E2E-SMART and about 9 times better than TCP Reno.

Another interesting point to note is the relative performance of E2E-IETF-SACK and E2E-SMART, especially at the high error rates. The congestion window does not grow larger than a few packets in the steady state at these error rates where there are multiple losses in many windows. E2E-IETF-SACK does not retransmit any packet using SACK information unless it receives three duplicate acknowledgments (to overcome potential reordering of packets in the network), which implies that no fast retransmissions are triggered if the number of packets in the window is less than four or five[4]. The sender's congestion window is often smaller than this, resulting in timeouts and degraded performance. In contrast, our implementation of E2E-SMART assumes no reordering of packets (which is justified in the LAN case) and retransmits the lost packet when the first duplicate acknowledgment with loss information arrives. This reduces the number of timeouts and results in better end-to-end performance. In Section 5.3, we outline a scheme in which the IETF protocol can be modified to work well even when the sender's congestion window is not large enough to provide enough duplicate acknowledgments.

5. Discussion

In this section, we present a discussion of some miscellaneous issues. We discuss the effects of handoff on TCP performance, some implementation strategies and policies for the ELN mechanism introduced in Section 3.1, and some issues related to SMART-based and IETF selective acknowledgment schemes.

5.1 Wireless Handoffs

Wireless networks are usually organized in a cellular topology where each cell includes a base station that acts as a router between the wireless subnet and a wireline backbone. Mobile hosts typically communicate via the base station in the cell they are currently located in. Examples of networks organized in this fashion include cellular telephone networks and wireless local-area networks.

As a mobile host moves, it may get out of the range of its current base station but still be within the range of other neighboring base stations. To maintain the mobile host's connectivity, a *handoff* procedure is invoked to re-route traffic to and from the mobile host via the new base station. However, depending on the details of the handoff algorithms, this procedure could lead to packet losses and reordering, which in turn could cause significant deterioration in the performance of ongoing TCP transfers [8].

Several proposals have been made for achieving fast handoffs. Two examples include multicast-based handoffs [25] and hierarchical handoffs [9]. In both these schemes, handoffs are made fast by restricting updates to the immediate vicinity of the mobile host. As a result the handoff latency in a WaveLAN-based wireless local-area network is of the order of 10-30 ms.

4. This depends on whether delayed acknowledgments are used.

A small amount of buffering and retransmission from base stations prevents packet loss during the short handoff period. In [9], the buffering happens at the mobile host's old base station, which forwards packets to the new base station at the time of handoff. In [25], one or more base stations in the vicinity join a multicast group corresponding to the mobile host and receive all packets destined to it, in anticipation of a handoff. When the handoff happens, the new base station is readily able to forward the buffered and the newly arriving packets without introducing any reordering, thereby preventing unnecessary invocations of TCP fast retransmissions. Experimental results reported in [25] indicate that such fast handoffs have a minimal adverse effect on TCP performance, even when the handoff frequency is as high as once per second.

In contrast to the above schemes that operate at the network layer, handoffs in a split-connection context, such as in I-TCP [3], involve the transfer of transport-layer state from the old base station to the new one. This results in significantly higher latency; for example, [2] reports I-TCP handoff latencies of the order of hundreds of milliseconds in a WaveLAN-based network.

5.2 Implementation Strategies for ELN

Section 3.1 described the ELN mechanism by which the transport protocol can be made aware of losses unrelated to network congestion and react appropriately to such losses. In this section, we outline possible implementation strategies and policies for this mechanism.

A simple strategy for implementing ELN would be to do so at the receiver, as we did for the results presented in this paper. In this method, the corruption of a packet at the link-layer, indicated by a CRC error, is passed up to the transport layer, which sends an ELN message with the duplicate acknowledgments for the lost packet. In practice, it may be hard to determine the connection that a corrupted packet belongs to, since the header could itself be corrupted: this can be handled by protecting the TCP/IP header using an FEC scheme. However, there are circumstances in which entire packets, including link-level headers, are dropped over a wireless link. In such circumstances, the base station generates ELN messages to the sender (in-band, as part of the acknowledgment stream) when it observes duplicate TCP acknowledgments arriving from the mobile host.

We expect Explicit Loss Notifications to be useful in the context of multi-hop wireless networks, and are exploring this in on-going work. Such networks (e.g., Metricom's Ricochet network [21]) typically use packet radio units to route packets to and from a wired infrastructure. Here, in order to implement ELN, periodic messages are exchanged between adjacent packet radio units about queue lengths and this information is used as a heuristic to distinguish between congestion and packet corruption, especially when entire packets (including headers) are corrupted or dropped over a wireless link. This, coupled with a simple link-level scheme to convey NACK information about missing packets, is sufficient to generate ELN messages to the source.

5.3 Selective Acknowledgment Issues

Our experience with the IETF SACK scheme highlights some weaknesses with it both when sender window sizes are small. This situation can be improved by enhancing the sender's loss recovery algorithm as follows. In general, the arrival of one duplicate acknowledgment at the receiver indicates that one segment has successfully reached the receiver. Rather than wait for three duplicate acknowledgments and perform a fast retransmission, the sender now transmits a *new* segment from beyond the "right edge" of the current window upon the arrival of the first and second duplicate acks. This probes the network for sustained congestion and generates duplicate acknowledgments. Note that we have not violated standard congestion control procedures by doing this: we only send out a segment when one has left the data pipe, following the principle of conservation of packets [13]. This enhancement can coexist with SACKs to further avoid timeouts, since the arrival of an acknowledgment with a SACK block indicating the reception of the newly transmitted segment is a strong indicator that the original segment was lost, independent of whether three duplicate acknowledgments arrive or not. Thus, this mechanism will improve performance when the sender's window is small and losses occur, and is further explored and described in [6].

6. 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 with selective acknowledgements performs the best.

2. The split-connection approach, with standard 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. In the WAN experiments, the SACK scheme based on the IETF Draft resulted in significantly improving end-to-end performance, although its performance was not as good as in the best link schemes. 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 extensive 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.

7. Future Work

Our experiments with various SACK and ELN mechanisms demonstrate the significant benefits of such schemes, as described in Section 5. We are in the process of evaluating protocol enhancements based on these ideas in the presence of both network congestion and wireless losses in different network topologies, especially in networks with multiple wireless hops. In addition, we are evaluating the performance of several of the protocols described in this paper under other patterns of loss derived from traces in [23].

We are investigating the impact of large variations in connection round-trip times and the impact of bandwidth and latency asymmetry on transport performance [5]. Large round-trip variations are common in networks like the Metricom Ricochet wireless network [21], especially in the presence of bidirectional traffic. Bandwidth asymmetry is prevalent in many cable and satellite networks with low-bandwidth return channels.

8. Acknowledgments

We are grateful to Steven McCanne and the anonymous reviewers for ACM SIGCOMM ’96 and IEEE/ACM Transactions on Networking for several comments and suggestions that helped improve the quality of this paper. We thank Sally Floyd and Vern Paxson for useful discussions on SACKs and related topics.

This work was supported by DARPA contract DAAB07-95-C-D154, by the State of California under the MICRO program, and by the Hughes Aircraft Corporation, Metricom, Fuji Xerox, Daimler-Benz, Hybrid Networks, and IBM. Hari is partially supported by a research grant from the Okawa Foundation.

9. 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 ACM/Baltzer Wireless Networks Journal*, 1:47–60, February 1995.

[2] A. Bakre and B. R. Badrinath. Handoff and System Support for Indirect TCP/IP. In *Proc. Second Usenix Symp. on Mobile and Location-Independent Computing*, April 1995.

[3] 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.

[4] H. Balakrishnan. An Implementation of TCP Selective Acknowledgments. ftp://daedalus.cs.berkeley.edu/pub/tcpsack/, 1996.

[5] H. Balakrishnan, V. N. Padmanabhan, and R.H. Katz. The Effects of Asymmetry on TCP Performance. In *Proc. ACM MOBICOM ’97*, September 1997.

[6] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm, and R.H. Katz. TCP Behavior of a Busy Web Server: Analysis and Improvements. Technical Report UCB/CSD-97-966, University of California at Berkeley, 1997.

[7] 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.

[8] 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.

[9] R. Caceres and V. N. Padmanabhan. Fast and Scalable Handoffs in Wireless Internetworks. In *Proc. 1st ACM Conf. on Mobile Computing and Networking*, November 1996.

[10] A. DeSimone, M. C. Chuah, and O. C. Yue. Throughput Performance of Transport-Layer Protocols over Wireless LANs. In *Proc. Globecom ’93*, December 1993.

[11] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno, and Sack TCP. *Computer Communications Review*, 1996.

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

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

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

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

[16] 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.

[17] S. Keshav and S. Morgan. SMART Retransmission: Performance with Overload and Random Losses. In *Proc. Infocom '97*, 1997.

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

[19] Mathis, M. and Mahdavi, J. and Floyd, S. and Romanow, A. *TCP Selective Acknowledgment Options*, 1996. RFC-2018.

[20] 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.

[21] Metricom, Inc. http://www.metricom.com, 1997.

[22] 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.

[23] G. T. Nguyen, R. H. Katz, B. D. Noble, and M. Satyanarayanan. A Trace-based Approach for Modeling Wireless Channel Behavior. In *Proc. Winter Simulation Conference*, Dec 1996.

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

[25] S. Seshan, H. Balakrishnan, and R. H. Katz. Handoffs in Cellular Wireless Networks: The Daedalus Implementation and Experience. *Kluwer Journal on Wireless Personal Communications*, January 1997.

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

[27] AT&T WaveLAN: PC/AT Card Installation and Operation. AT&T manual, 1994.

[28] 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.

Hari Balakrishnan (S '95 / ACM S '95) is a Ph.D. candidate in Computer Science at the University of California at Berkeley. His research interests are in the areas of computer networks, wireless and mobile computing, and distributed computing and communication systems. His current research is in the area of reliable data transport over heterogeneous networking technologies.

Hari received a B. Tech. degree in Computer Science and Engineering from the Indian Institute of Technology, Madras, in 1993 and an M.S. degree in Computer Science from Berkeley in 1995. He received best student paper awards at the Winter Usenix '95 and at the ACM Mobicom '95 conferences, and is the recipient of a research grant from the Okawa Foundation. On the WWW, his URL is http://www.cs.berkeley.edu/~hari and his e-mail address is hari@cs.berkeley.edu.

Venkata N. Padmanabhan (IEEE S '94 / ACM S '94) is a Ph.D. candidate in Computer Science at the University of California at Berkeley. He received his B.Tech. degree from the Indian Institute of Technology, Delhi in 1993 and his M.S. degree from the University of California at Berkeley in 1995, both in Computer Science.

Venkat has done research in the areas of Computer Networking, Mobile Computing and Operating Systems. The focus of his current work is network support for efficient Web access, and data transport over asymmetric networks. He received the best student paper award at the Usenix '95 conference. He may be reached via e-mail at padmanab@cs.berkeley.edu and on the Web at http://www.cs.berkeley.edu/~padmanab.

Srinivasan Seshan (ACM M '92) received a B.S. in Electrical Engineering, an M.S., and a Ph.D. in Computer Science from the University of California at Berkeley in 1990, 1993 and 1995 respectively. Since 1995, he has been a research staff member at the IBM T.J. Watson Research Center. His research interests include computer networks, mobile computing and distributed computing. His e-mail address is srini@watson.ibm.com and his WWW home page is at http://www.research.ibm.com/people/s/srini.

Randy H. Katz (F '96, ACM F '96) is a professor of computer science at the University of California at Berkeley, and is a principal investigator in the Bay Area Research Wireless Access Network (BARWAN) project. He has taught at Berkeley since 1983, with the exception of 1993 and 1994 when he was a program manager and deputy director of the Computing Systems Technology Office at the Defense Department's Advanced Research Projects Agency. He has written over 130 technical publications on CAD, database management, multiprocessor architectures, high performance storage systems, video server architectures, and computer networks.

Dr. Katz received a B.S. degree at Cornell University, and an M.S. and a Ph.D. at the University of California at Berkeley, all in computer science. His e-mail address is randy@cs.berkeley.edu and his WWW home page is http://www.cs.berkeley.edu/~randy.