TCP Performance over Wireless Links

EE359 – Wireless Communications
Autumn 2001
Stanford University

December 12, 2001

Amol Shah

Table of Contents:

Introduction

Wireless networks play a very important role in communications today. People want information on demand – any place, any time. Wireless networks help solve this problem. Reliable transport protocols however, such as TCP that have been traditionally used for wired networks do not perform as well on wireless networks. This is because TCP assumes that packet loss and unusual delays are mainly caused by congestion. TCP is thus tuned to adapt to such congestion losses by slowing down the amount of data it transmits. It drops its transmission window size and backs off its retransmission timer, thus reducing the load and congestion on the network. In wireless networks however, packet loss is often caused due to other factors besides congestion. Wireless channels often suffer from high bit error rates (BER) and intermittent connectivity due to handoffs. Moreover, the BER may vary continuously during a session. TCP unfortunately assumes that these losses are due to congestion and invokes its congestion control measures. This results in an unnecessary reduction in end-to-end throughput and thus sub-optimal performance.

As a result, several schemes have been proposed to deal with these issues and to alleviate the effects of non-congestion related losses on TCP performance. These schemes usually follow one of two approaches [1]. The first approach is to hide any non-congestion related losses from the TCP sender. This requires no changes to existing TCP implementations. The reasoning behind this approach is that since the problem is local, it should be handled locally, and the transport layer should not have to be aware of the characteristics of the individual links. These protocols attempt to make the lossy link appear as a higher quality link with a reduced effective bandwidth. The other approach is 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 thus avoid invoking congestion control algorithms when non-congestion related losses occur. The rest of this paper takes a look at a number of different schemes that have been proposed to deal with these problems and their performance in a wireless environment.

Background

The various schemes proposed to improve the performance of TCP on wireless links can be broadly categorized into one of three groups: end-to-end protocols, split-connection protocols and link-layer protocols. The key concepts in each of the three categories are as follows:

End-to-end protocols

These protocols retain a single TCP connection from sender to receiver and attempt to make TCP aware of wireless losses so that it can deal with them. They handle losses through the use of some form of selective acknowledgements (SACKs). This allows the sender to recover from losses within a window without needing to timeout. They also attempt to have the sender distinguish between congestion and other forms of losses using an Explicit Loss Notification (ELN) mechanism.

Split connection protocols

Unlike the end-to-end protocols, these protocols completely hide the wireless link from the sender by terminating the TCP connection at the base station. A separate reliable connection is then used between the base station and the destination. This allows the second connection to use different techniques such as selective acknowledgements and so on. The advantage of this is that the TCP implementation in the sender does not need to be modified to deal with the enhanced functionality required for the wireless hop.

Link-layer protocols

These protocols lie between the end-to-end protocols and the split-connection protocols. They attempt to hide link-related losses from the TCP sender by using local retransmissions at the link layer. The local retransmissions use techniques that are tuned to the characteristics of the wireless links. However, since the end-to-end TCP connection passes through the lossy link, the TCP sender may not be fully shielded from wireless losses.

Protocol implementations

This section now takes a look at specific protocols that have been implemented in each of the three areas discussed above.

End-to-end schemes

End-to-end protocols attempt to make TCP senders be aware of non-congestion related losses through the use of an explicit loss notification mechanism. The TCP congestion control mechanisms are not invoked when the packet loss is due to a non-congestion related cause. The following are different schemes that have been proposed to deal with these issues [2].

Caceres and Iftode [4] proposed a scheme that uses fast retransmission in the event of packet loss. This focuses on improving throughput and reducing interactive delay to an acceptable level by forcing the TCP layer to retransmit packets as soon as handoff completes, without waiting for a TCP retransmission timeout. The advantages of this are that it only requires some modification to end systems and does not rely on special support from the underlying networks. It requires however that the transport layer protocol be able to differentiate between channel-related and congestion related loss. Moreover, it only considers handoff related losses, but does not take into account losses due to the wireless channel itself.

Space Communication Protocol Standard – Transport Protocol (SCPS-TP) is another proposed protocol that copes with different sources of packet loss in addition to handoff related losses – network congestion, corruption, and link outage [5]. It implements an appropriate response to each of the losses. The main drawback though is that it would require modifying existing TCP implementations on the Internet, which may not always be feasible.

Selective acknowledgements are yet another way of dealing with the wireless loss issue, and are defined as an option to TCP in RFC 2108 [6]. Since standard TCP uses a cumulative acknowledgement scheme, it does not provide the sender with sufficient information to recover quickly when multiple packets are lost within a single transmission window. With the SACK scheme, each acknowledgement contains information on up to three noncontiguous blocks of data that have been received successfully by the receiver. This allows TCP senders to retransmit only the missing data segments. Similar to the earlier two proposals, the main drawback of SACK is that it requires modifying the acknowledgement procedure at the sender and the receiver.

Thus, end-to-end schemes involve modifying the TCP implementation in the sender to make it “wireless” aware. This may not always be feasible and is a potential problem for widespread implementation of end-to-end schemes.

Link layer schemes

Error correction techniques such as forward error correction and retransmission of lost packets in response to automatic repeat request (ARQ) messages are the two main techniques employed by link-layer protocols. The following are some link-layer schemes that have been proposed to deal with the TCP wireless problem.

Asymmetric Reliable Mobile Access In Link-Layer (AIRMAIL) is a link layer scheme proposed by [7] to improve reliability. It involves placing a large number of intelligent functions in the base station and making the timers always at the base station. The mobile host combines several acknowledgments into a single acknowledgement to conserve power, and the base station sends periodic status messages to the mobile host. It uses forward error correction with automatic repeat request (ARQ) for error recovery and detection over unreliable links.

Another scheme called the snoop protocol introduces agents called snoop agents at the base station [8]. The snoop agent monitors every packet that passes through the TCP segments sent from the source and that the receiver has not yet acknowledged. Packet loss is detected by the arrival of duplicate acknowledgements from the receiver or by a local timeout. It also suppresses duplicate acknowledgements from the TCP sender, thus avoiding unnecessary fast retransmission and congestion control procedures. The main disadvantage of the snoop protocol is that it does not consider packet loss and delay due to handoff, and the interference of data link with transport layer retransmission is still present [2].

Thus, the advantage of link-layer schemes is that they fit naturally into the layered structure of the network protocol stack. They do not require maintaining per-connection state, and can improve the reliability of transmission independent of the higher-layer protocols. However, since TCP has an end-to-end reliable transmission scheme, the interaction between the link-layer transmission and end-to-end transmission can become complicated and can still pose a potential problem leading to degradation in the overall TCP throughput [2].

Split connection schemes

As mentioned earlier, split-connection protocols hide the unreliability of the wireless link from the sender by terminating the TCP connection at the base station and using a separate protocol from the base station to the mobile host. This has the advantage that the TCP implementation in the sender does not have to be modified.

Indirect-TCP (I-TCP) is one of the split connection schemes that have been proposed. The advantages of I-TCP are that it can isolate host mobility and wireless-related problems through the use of a Mobility Support Router (MSR) as an intermediary that provides backward compatibility with fixed network protocols [2]. The drawbacks of I-TCP however include the fact that end-to-end semantics of TCP acknowledgements are violated. Since the connection is split into two separate TCP connections, an acknowledgement may reach the sender before the data packet is actually received by the receiver. Also, the handoff latency is usually long as each I-TCP connection's states must be transferred from one base station to another.

An alternative proposal called M-TCP also involves separating the wireless network from the fixed high speed Internet. The TCP connection between the Internet and wireless network is through a supervisor host (SH). The fixed host to SH part of the connection uses regular TCP, while M-TCP is used from the SH to the mobile host. Unlike I-TCP, the M-TCP implementations use TCP-like acknowledgements. The SH always acknowledges all but the last byte of data that it received from the sender. The last byte is acknowledged only after it has been successfully sent to the mobile host by the SH. The drawbacks of this scheme are that the semantics of M-TCP acknowledgments have problems when the SH crashes, and the long handoff latency that is seen in the I-TCP implementation is still present [2].

Protocol performance

This section takes a look at the performance of the various techniques presented earlier to deal with TCP performance on wireless links. The simulations and the results presented here are from [1]. As described in this section, the actual protocols implemented for the simulation are not the same as the one's mentioned in the previous section. Instead, the protocols simulated take a particular characteristic used in each of the earlier techniques and simulate it to determine how effective that technique is in combating the problems TCP has on wireless links. Table 1 summarizes these simulated protocols and identifies the category and the particular characteristic being tested through the protocol.

Name	Category	Characteristic
E2E	End-to-end	Standard TCP Reno
E2E-NEWRENO	End-to-end	Fast recovery mode

E2E-SMART	End-to-end	SMART-based selective acks
E2E-ELN	End-to-end	Explicit loss notification
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 simulated [1]

The following is now a more detailed explanation of the protocols mentioned in Table 1.

End-to-end protocols

The current de facto standard for TCP implementation on the Internet is TCP Reno [1]. This paper refers to it as the E2E protocol. The E2E-NEWRENO protocol improves the performance of TCP-Reno. It does so by remaining in fast recovery mode if the first new acknowledgement received after a fast transmission is less than the value of the last byte transmitted. This helps the recovery when there are multiple packet losses in a window as remaining in the fast recovery mode enables the sender to recover from losses quicker rather than blocking until a timeout as TCP-Reno would.

The E2E-SMART protocol adds SMART based selective acknowledgments[1] to the standard TCP Reno stack. This allows the sender to handle multiple losses within a window of outstanding data more efficiently. The sender still assumes however that the losses are a result of congestion and invokes congestion control procedures. This scheme is well suited to situations where there is little reordering of packets, which is true for one-hop wireless systems. The sender retransmits a packet when it receives a SMART acknowledgement if the same packet was not retransmitted within the last round-trip time. If no further SMART acknowledgements arrive, the sender falls back to the timeout mechanism to recover from the loss.

The E2E-ELN protocol adds an Explicit Loss Notification option to TCP acknowledgements. Whenever a packet is lost, future cumulative acknowledgements corresponding to the lost packet are marked to identify that a non-congestion related loss has occurred. When the sender receives this acknowledgement with the duplicate

[1] SMART uses acknowledgements that contain the cumulative acknowledgement and the sequence number of the packet that caused the receiver to generate the acknowledgement. The sender uses this information to create a bitmask of packets that have been delivered successfully. When the sender detects a gap in the bitmask, it automatically assumes that the missing packets have been lost. It thus trades off some resilience to reordering and lost acknowledgements in exchange for a reduction in overhead to generate and transmit acknowledgements [1].

acknowledgements, the sender retransmits without invoking the congestion-control procedures.

It is important to note here that an implicit assumption is that sufficient knowledge is available at the receiver to be able to identify which packets are lost due to errors on a lossy link. This may not always be true in practice.

Link-layer protocols

Unlike TCP for the transport layer, there is no de facto standard for link layer protocols [1]. Some of the existing link-layer protocols are Stop-and-Wait, Go-Back-N, and Selective Repeat. The base link-layer algorithm called LL that was studied in [1] uses cumulative acknowledgements to determine lost packets that are retransmitted locally from the base station to the mobile. The LL scheme uses a shorter timeout interval, which allows it to retransmit several times before the TCP transmitter times out. It does not suppress any duplicate acknowledgements, and does not attempt in-order delivery of packets on the link.

LL-SMART is a more sophisticated link-layer protocol that uses selective retransmissions to improve performance. This protocol uses SMART-based acknowledgements. It processes selective acknowledgements but does not suppress duplicate acknowledgements.

LL-TCP-AWARE and LL-SMART-TCP-AWARE are modified versions of the LL and LL-SMART protocol with TCP awareness. These protocols suppress duplicate acknowledgements as well, thus further improving the TCP performance by preventing the TCP sender from entering fast transmission due to these duplicate acknowledgements.

Split connection protocols

Similar to I-TCP, the SPLIT protocol studied uses an intermediate host to divide a TCP connection into two separate TCP connections. A variant of the SPLIT protocol with a SMART-based selective acknowledgement called SPLIT-SMART was also used to perform selective retransmissions.

Results

Results from the experiments on the various protocols performed by [1] are presented here. The experimental setup consisted of IBM Thinkpad laptops and Pentium based personal computers running BSD/OS 2.1. The machines were interconnected using a 10 Mbps Ethernet and 915 MHz WaveLAN with a bandwidth of 2 Mbps. Errors were generated on the lossy link using an exponentially distributed bit-error model. Losses were generated in both directions of the wireless channel. The TCP data size used was 1400 bytes. The performance of the protocols was analyzed across a range of error rates, from one every 16 KB to one every 256 KB. Each run in the experiment consist of an 8 Mbyte transfer from the source to receiver across the wired net and the WaveLAN link.

During each run, the throughput in Mbps and the wired and wireless goodputs[2] are measured. The WAN experiments were performed across 16 Internet hops (with minimum congestion) to study the impact of large delay-bandwidth products.

Link-layer protocols

Link layer protocols operate independently of the higher layer protocol, and do not necessarily shield the sender from the lossy link. Thus, even though there are local retransmissions, TCP performance could be poor for one of two reasons: (i) competing transmissions caused by an incompatible setting of timers at the two layers (ii) unnecessary invocations of the TCP fast retransmission mechanism due to out-of-order delivery of data.

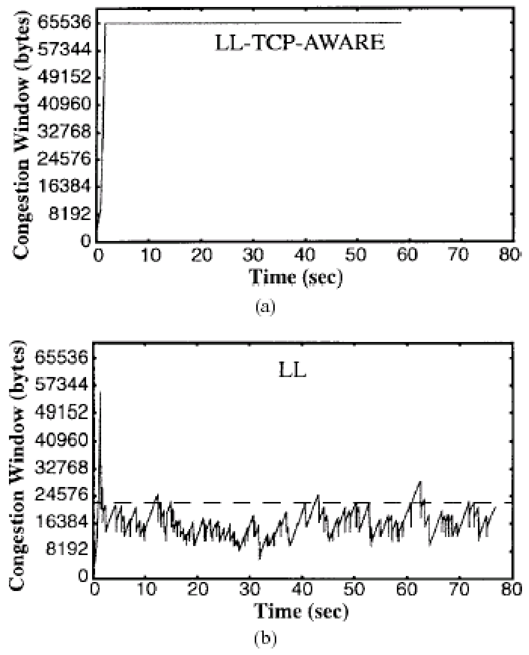

Figure 1 – Congestion window size for link-layer protocols. The dashed line in (b) shows the 23000 byte WAN bandwidth delay product [1].

The results from the simulations run on the LL and LL-TCP-AWARE protocol are shown in Figure 1. 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%, which is lower than LL-TCP-AWARE's wireless goodput of 97.6%. This is close to the maximum achievable goodput for the experiment run, as with a packet size of 1400 bytes and an error rate of 1 in 64 KB (bit error rate 1.9x10^-6), the packet error rate is about 2.3%. The throughput for the LL-TCP-AWARE scheme was 1.35 Mbps.

When a loss occurs, the LL protocol performs a local retransmission relatively quickly. However, enough packets are typically in transit to create more than three duplicate acknowledgements. These duplicates cause the sender to go into fast retransmission mode and invoke congestion control procedures. This results in a reduced throughput as about 90% of the lost packets are retransmitted by both the source and the base destination. These effects are much more pronounced in the wide-area experiments. The LL scheme causes the sender to invoke congestion control procedures often due to duplicate acknowledgements and causes the average window size of the transmitter to be lower than for LL-TCP-AWARE. This is shown in Figure 1(b). Another issue in the WAN tests is the fact that the bandwidth-delay product is large, and the congestion window often drops below this value, thus leading to the "data pipe" not being full all the time. The LAN tests do not suffer from this problem as its congestion-window size is normally larger than the connection's delay-bandwidth product.

[2] Goodput is defined as the ratio of the actual number of user bytes to the total number of bytes transmitted.

The 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 is made TCP aware and prevents duplicate acknowledgements caused by wireless losses from reaching the sender and locally retransmits packets achieves significantly better performance.

End-to-end protocols

The performance of TCP Reno, or E2E, shows the problems with TCP over lossy links. At a packet loss rate of 2.3% that was mentioned in the previous section, the E2E protocol achieves a throughput of less than 50% of the maximum in LANs, and less than 25% in the wide-area experiments. This reduced throughput is due to the large number of timeouts that occur during the transfer. The average window size remains small as a result, thus reducing the effectiveness of the fast retransmission mechanism.

The modified end-to-end protocols achieve better throughput by retransmitting packets that have been lost earlier than they would have by the E2E protocol. This helps in reducing the fluctuation in the window size that would occur if the packet was not retransmitted and the sender timed-out. These modified protocols use more sophisticated acknowledgement techniques to improve the speed and accuracy of identifying and retransmitting lost packets. These techniques include: partial acknowledgements, explicit loss notifications, and selective acknowledgements.

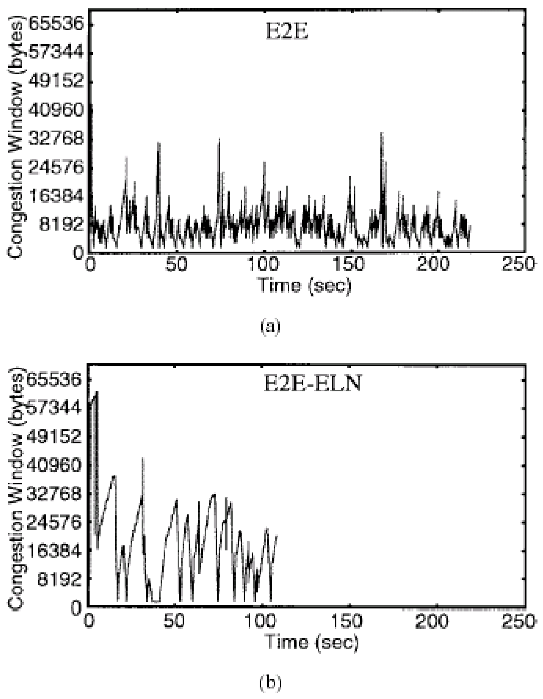

Figure 2 – Congestion window size as a function of time for E2E and E2E-ELN [1].

E2E-NEWRENO uses partial acknowledgement information to recover from multiple losses in a window. It performs between 10 and 25% better than E2E over a LAN and about 2 times better than E2E in the WAN experiments.

E2E-NEWRENO remains in fast recovery if the new acknowledgement is only partial, but reduces the window size to half its original value upon the arrival of the first new acknowledgement. The E2E-ELN protocol prevents the sender from reducing the size of the congestion window in response to a wireless loss through the use of ELN information (Figure 2). This helps maintain a larger average congestion window size thus helping improve the throughput.

Experiments were also conducted with the SACK schemes. A simple SACK scheme based on a subset of the SMART proposal was found to be the best of the end-to-end protocols. It achieved a throughput of 1.25 Mbps.

Thus, E2E-NEWRENO is better than E2E. Adding ELN to TCP improves the throughput significantly by successfully preventing unnecessary fluctuations in the transmission window. Also, SACKs provide a 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 than the WAN experiments. This suggests that an end-to-end protocol that has both ELN and SACKs will result in good performance. All the end-to-end protocols achieve goodputs close to the optimal value of 97.7%.

Split-connection protocols

Split connection protocols isolate the TCP source from the wireless losses. The TCP sender on the wireless connection performs all the retransmissions in response to the wireless losses. Results are presented for two cases – when the wireless connection uses TCP Reno (labeled SPLIT) and when it uses the SMART-based selective acknowledgement scheme described earlier (labeled SMART-SPLIT). The throughput achieved by SPLIT is similar to TCP Reno and is on the order of about 0.6 Mbps. The reason for this is the low congestion size window size for the wireless connection as shown in Figure 3. The throughput for the SPLIT-SMART scheme is much higher. It achieves a throughput of about 1.3 Mbps in the LAN case. The SMART-based selective acknowledgement scheme operates well especially since there are no reordering of packets on the wireless hop. Thus, 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.

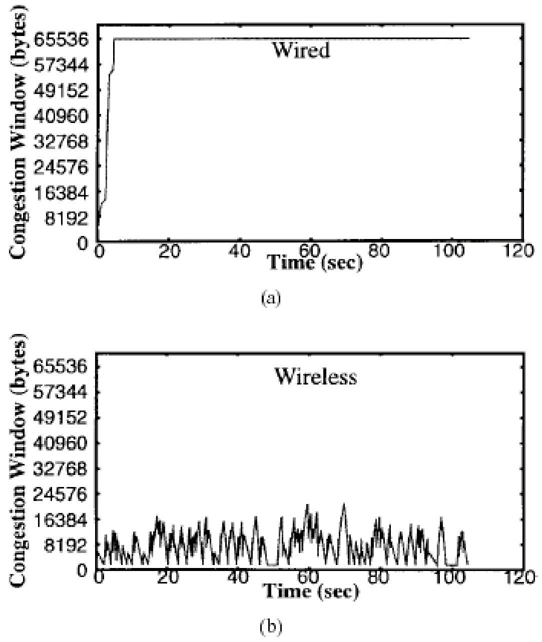

Figure 3 – Congestion window size as a function of time for wired and wireless parts of the split TCP connection [1].

Additional performance parameters

In addition to the protocols mentioned in the previous sections, there are other issues that can contribute to the performance of TCP on wireless links. These include TCP header compression, adaptive MAC frame size as well as handoff algorithms. This section takes a brief look at these issues and how they impact TCP performance.

Handoffs

Depending on the details of the handoff algorithm being used, handoffs can often lead to packet losses and reordering, which in turn can cause significant deterioration in the performance of TCP. As a result, several proposals have been made for achieving fast-handoffs. These schemes restrict updates to the immediate vicinity of the mobile host. One such scheme involves a multicast based solution [2]. In this approach, the packets for a mobile host are multicast to the base stations of the neighboring cells so that when the mobile host moves to a new cell, there are packets waiting for it. While this scheme provides seamless communications, it is not bandwidth effective. As the number of connections increases, the amount of network bandwidth used by multicast is high. As an alternative to this, [2] proposes a scheme where traffic is separated into two classes – real time and non-real time traffic. As non-real time traffic can tolerate some delay, a forward based scheme is used instead of a multicast based scheme. When the new base station receives a reply from the home agent, the old base station would be notified at the same time. The old base station would then open a connection to the new base station and would send all information about the TCP connection to the new base station. The new base station then buffers any packets it receives till the handoff is completed. This procedure incurs a long latency, but it saves network bandwidth. On the other hand, real-time traffic is sensitive to delays. As a result, [2] proposes a direction-based selective multicast scheme to reduce latency and packet loss due to handoff. The direction-based selective multicast is initiated by taking into account the measurements of signal strength and direction of movement of the mobile host. Based on this information, the base station would calculate which base stations the mobile might move to and would multicast packets to these base stations.

TCP/IP Header Compression

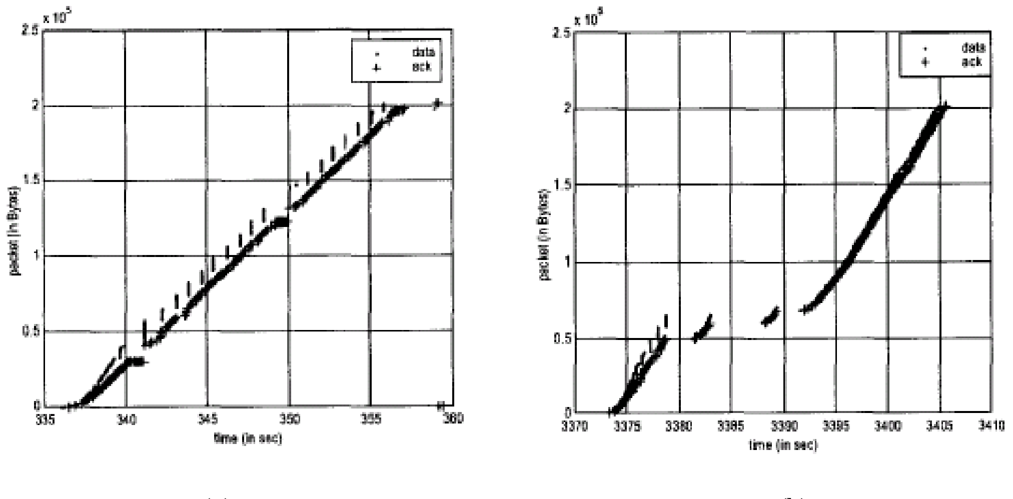

Figure 4 – (a) TCP data flow without header compression (b) TCP data flow with header compression [3]

As the bandwidth of wireless networks is scarce, TCP/IP header compression is a simple way to improve the bandwidth usage. Under this scheme, the sender would transmit a full header for the first time, and then send compressed headers that contain only the changes for successive data packets. Traditional TCP/IP headers are 40 bytes – 20 bytes for the IP layer and 20 bytes for the TCP layer. [2] proposes a scheme where some fields of the header are omitted or shortened over the wireless links. For example, the address and port number fields of the TCP header are the same in all packets that belong to the same TCP session, so they only need to be transmitted in the first packet. The main problem with this is that if a packet loss occurs, or if a packet is received out of order, the receiver may not be able to reconstruct the original headers of subsequent packets. Based on tests performed on Metricom's Ricochet network, [3] suggests removing header compression. Figure 4 shows TCP data flow with and without header compression. It is worth noting that in figure 4(a), TCP recovered very quickly from the error due to fast retransmission. In figure 4(b), the sender only resent the lost packet with an uncompressed header after the time out interval, and hence did not recover as fast.

Adaptive MAC Frame size

The MAC frame size also has a great impact on the performance of wireless networks [2]. In traditional networks, the Maximum Transmission Unit (MTU) is determined at the time that the TCP/IP connection is established. However, the conditions of wireless channels are not stable, so it is not good to transmit fixed MTU over wireless links during the whole life of the TCP/IP session. Shorter packets are less likely to suffer from errors than long packets, but on the other hand, the header overhead is greater for shorter packets, thus leading to reduced goodputs. This is why an adaptive scheme is needed to adjust the MAC frame size based on the channel conditions. Figure 5 shows the normalized goodput versus data length for various BERs based on experiments by [2]. The figure shows that as the channel conditions deteriorate, the sender should use a smaller length of user data. Table 2 gives the optimum user data length for different BERs. The base station can then use a channel estimator to estimate the approximate BER of the wireless links. It can then broadcast the optimal MAC frame length to the mobile hosts periodically by looking in the table.

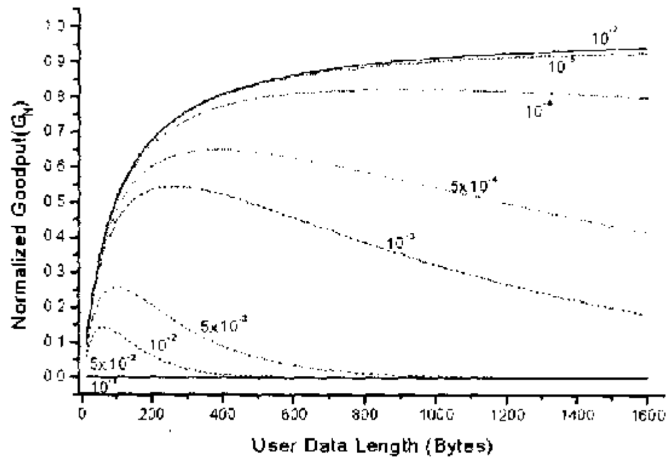

Figure 5 – Normalized goodput vs. User data length for varying BERs [2]

BER	10^{-1}	$5x10^{-2}$	10^{-2}	$5x10^{-3}$	10^{-3}	$5x10^{-4}$	10^{-4}	$<10^{-5}$
L (bytes)	10	20	60	100	260	390	920	1500

Table 2 – Optimum user data length (L) vs. bit error rate (BER)

Conclusions

This paper took a look at various proposed protocols that attempt to improve the performance of TCP on wireless links. These protocols were divided into three broad categories based on the techniques they used. These categories were end-to-end protocols, link-layer protocols, and split-connection protocols. The performance of these protocols was also studied through simulations. It was seen that a reliable link-layer protocol that used selective acknowledgements and knowledge of TCP to shield the sender from duplicate acknowledgements arising from wireless losses performs better than the other protocols studied.

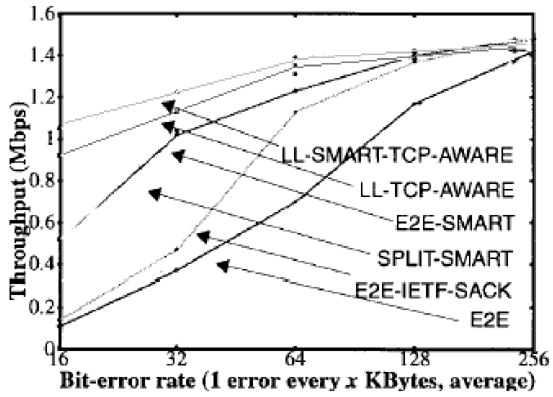

Figure 6 – Performance of the different protocols at different BERs [1]

In addition, the paper looked at some other important issues that can affect TCP performance. These include handoff algorithms, TCP header compression and MAC frame size. Each of these parameters needs to be tuned based on the conditions of the channel and can greatly impact performance.

References

[1] Chen, W and Lee, J. "Some Mechanisms to Improve TCP/IP Performance over Wireless and Mobile Computing Environment," Proceedings of the Seventh International Conference on Parallel and Distributed Systems (ICPADS'00), Iwate, Japan, pp. 437-444, July, 2000.

[2] Balakrishnan, H., Padmanabhan, V., Seshan, S. and Katz, R. "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," IEEE/ACM Transactions on Networking, pp. 756-769, December, 1997.

[3] Srivastava, A., Friday, R., Ritter, M. and Filippo W. "A Study of TCP Performance Over Wireless Data Networks," IEEE VTS 53rd, vol. 3, pp. 2265-2269, 2001.

[4] Cacres, R. and Iftode, L. "Improving the performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE JSAC, vol. 13, pp. 850-857, June, 1995.

[5] Durst, R., Miller, G. and Travis, E. "TCP extension for space communication," Wireless Network, vol. 3, pp. 389-402, 1997.

[6] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A. "Selective Acknowledgement options," Internet Engineering Task Force, October 1996, RFC 2108.

[7] Ayangolu, Paul, S., LaPorta, T., Sabnani, K., Gitlin, R. "AIRMAIL: A Link Layer Protocol for Wireless Networks," Wireless Networks, vol. 1, pp. 47-60, 1995.

[8] Balakrishnan, H., Seshan, S. and Katz, R. "Improving Reliable Transport and Handoff performance in cellular wireless network," Wireless Networks, vol. 1, pp. 469-481, 1995.