Measurements of TCP Performance over Wireless Connections

Venkata K. Josyula, Richard B. Bunt
Department of Computer Science
University of Saskatchewan
and
Janelle J. Harms
Department of Computing Science
University of Alberta

Abstract

As wireless links work their way into the Internet fabric, it becomes increasingly important to understand how to support the additional requirements presented by wireless connections and how to accommodate the different characteristics of these connections. As a protocol designed for wired connections, the Transmission Control Protocol (TCP) operates under some basic assumptions that may not be true for wireless connections, for example that packet losses are due to congestion in the network. Because TCP is the foundation of so many applications on the Internet, it will be necessary to examine its operation in the presence of wireless links very carefully. This paper reports on a preliminary investigation of TCP performance across wireless links. Our results are derived from measurements taken on an experimental testbed, and our work differs from earlier work in its consideration of matters internal to TCP, such as the behaviour of the retransmission timeout and windowing mechanisms in the presence of a wireless link. This helps us to understand in detail the impact of a wireless link on end-to-end TCP performance.

1 Introduction

An increased need to maintain access to information from all locations and at all times has led to an increased interest in wireless network connections for mobile users. Wireless links exhibit different properties from wired ones and, therefore, some of the assumptions made by higher level protocols designed for wired links may be invalid. Since many applications on the Internet use the Transmission Control Protocol (TCP) as the transport layer protocol (it has been reported [3] that TCP packets make up 80% of all wide-area network traffic), TCP must perform well in the wireless environment if wireless links are to be integrated seamlessly. Our research focuses on the performance of TCP in a wireless environment.

We begin by reviewing important design features of the TCP protocol in Section 2, paying particular attention to issues that might be affected by wireless connections. This is followed by a brief review of some related work in Section 3. We describe our experimental work in Section 4, and present conclusions from this preliminary study in Section 5.

2 TCP Overview

TCP [7] is a reliable connection-oriented transport layer protocol in the Internet Protocol (IP) suite. Reliable delivery means that all data packets arrive at the destination, where they are delivered to the application program in the correct order (regardless of how they may have been routed). Providing reliable delivery implies the presence of mechanisms to recover from lost packets, and mechanisms to regulate packet flow and control congestion in the network. TCP was designed for wired networks where it is assumed that packet loss is due primarily to congestion and that losses due to transmission errors on the medium are negligible. Good overviews of TCP and its component mechanisms can be found in [4] and [8].

To ensure delivery, TCP requires that the receiver acknowledge receipt of packets by sending an acknowledgement back to the sender (*positive acknowledgement*). Acknowledgements may be *cumulative* in the sense that an acknowledgement may not be sent immediately, and when sent can acknowledge more than a single packet. The sender is responsible for retransmitting packets that are not received. There are several elements to this process. Sequence numbers associated with the packets are used by the receiver to identify out-of-order packets, and by the sender to identify which packets are being acknowledged. The sender sets a retransmission timer at the time a packet is transmitted. Should this timer expire before receipt is acknowledged, the packet is assumed to be lost and is retransmitted. This may result in unnecessary duplication if the packet was received but the acknowledgement was delayed or lost (due to network congestion, for example).

With a view to performance, TCP uses a sliding window mechanism to control the flow of packets across the network and to reduce the rate of transmission when it believes there is congestion in the network. The size of the window determines the maximum amount of unacknowledged data that the sender can have outstanding. The receiver can adjust this window size by sending a *window advertisement* that indicates the amount of data it is currently able to handle. An increased window advertisement signals to the sender that the receiver can handle more data, while a decreased advertisement is used to reduce the rate of transmission until sufficient buffer space is available.

TCP also seeks to limit congestion in the network by employing algorithms called *congestion avoidance* [6] and *slow start*. These add two things: another window to the sender, called the *congestion window*, and another parameter called the *slow start threshold (ssthresh)*. The amount of data the sender is allowed to transmit is limited by the minimum of the congestion window and the advertised window of the receiver. Under slow start, the congestion window starts small and increases exponentially as acknowledgements are received, until the ssthresh threshold is reached. At this point congestion avoidance takes over and the rate of increase becomes much slower. Ssthresh is reduced (by half) whenever congestion is detected. Since TCP assumes an essentially loss-free environment, congestion is assumed when duplicate acknowledgements arrive or when timeouts occur. A timeout also reinitializes the congestion window. In effect, TCP reacts to timeouts by reducing the rate of packet transmission. There is one additional control of interest to this investigation. The sender keeps track of the round trip times of the packets it sends and adjusts the retransmission timeout variable, RTO, dynamically. As acknowledgements arrive, the RTO is adjusted, upwards or downwards, based on measured round trip time (RTT). When a timeout occurs, the RTO value is doubled. This control is designed to prevent excessive packet retransmission when acknowledgements are delayed by congestion in the network. Once again, this action is based on an assumption that acknowledgements are only delayed and not lost.

TCP's actions are effective in a wired environment where loss of a packet is rare, but when this is not the case (as in a network with wireless links) these actions need to be reconsidered. Even when there is packet loss due to the lossy nature of the medium, TCP reduces the transmission window size to reduce the amount of data sent, activates the slow start algorithm, and doubles the retransmission timeout value with each timeout. The effect of such actions on end-to-end performance needs to be examined carefully.

3 Related Work

Work related to this research is of two types: studies of TCP performance in wireless environments, and proposals for modifications to TCP for use in wireless environments. An example of work of the first type is a study by De Simone *et al.* [1] that investigated the effect of link level retransmission on transport layer performance through simulation and analytical techniques. They showed that the interaction between the link level and transport layer retransmissions can lead to degraded TCP performance especially as the error rates become more significant.

There have been several proposals to improve TCP performance in networks with wireless links and in the presence of mobility. One such proposal, referred to as Indirect-TCP (I-TCP) [2], splits the TCP connection between the wired and the wireless parts. Problems relating to the presence of wireless links (or the movement of a mobile user) are handled by the wireless part; no new software is required on fixed hosts. Experiments conducted on an experimental testbed showed substantial improvements over standard TCP in this sort of environment. A second group suggests caching packets at the base station and performing local retransmissions of TCP packets across the wireless link [5]. They introduced a new "snoop" module at the base station which keeps track of the acknowledgements going from the receiver to the sender. When a packet loss is detected, the packet is retransmitted from the cache thus hiding the packet loss from the sender. This prevents the invocation of the congestion control mechanism at the sender.

4 The Experiments

The purpose of these experiments is to determine how TCP responds to the packet loss on a wireless link. We take steps to increase the error rate on the wireless link by increasing the wireless distance. This requires that TCP must retransmit the packets that are lost. We keep the overall load on the network low so that congestion is not an issue, even with the retransmissions, so that the only problem

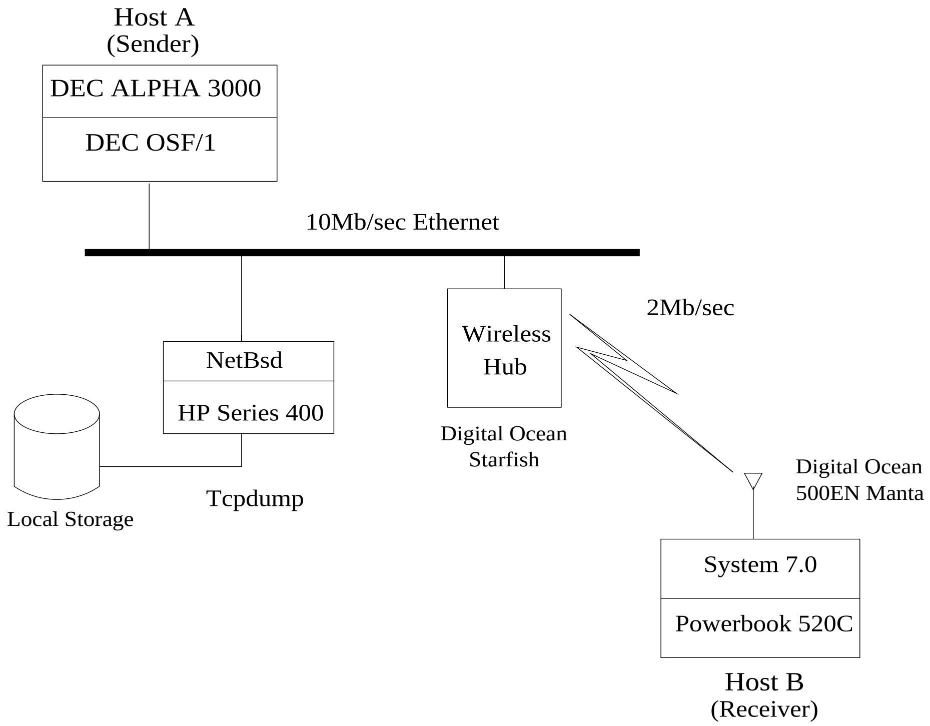

Figure 1: The Experimental Testbed

with our connection is packets lost on the wireless link.

In this section, we give details of the experimental configuration, workload, and measurement tools that we used. We then describe the performance measures that we employed and present our experimental results.

4.1 The Experimental Testbed

Our experimental testbed consists of a Host A (a DEC Alpha 3000 workstation running the DEC OSF/1 operating system) on a 10 Mb/sec Ethernet communicating with a Host B (an Apple PowerBook 520C laptop computer running the MacOS operating system, version 7.0) through a wireless connection as shown in Figure 1. The wireless link is rated at a raw speed of 2 Mb/sec and uses spread spectrum communication operating in the 902-928 MHz range following the IEEE 802.11 standard. Host B is provided with a Digital Ocean 500EN Manta wireless interface which communicates with a Digital Ocean Starfish wireless hub. Host A acts as the sender and Host B as the receiver.

4.2 The Workload

A TCP connection is set up between Host A (the *sender*) and Host B (the *receiver*) and a 10 Mbyte data file is transferred using the TCP streams socket. This translates to approximately 9800 TCP packets.

4.3 Measurement Tools

Three different tools were used to collect data. So that no additional traffic was placed on the network, all data was stored locally and analysed off-line. The network monitoring tool *tcpdump* was used to collect traces of the TCP packets flowing between the two hosts. Tcpdump ran on a separate machine attached to the Ethernet as shown in Figure 1 and provided information on packet sequence numbers, acknowledgements, and window advertisements. A program called *netstat* was run at both the sender and the receiver before and after each run of the experiment, and provided information such as number and type of packets sent and received. Some parameters of interest, such as the congestion window and timeout values at the sender, reside in the operating system kernel (specifically, in the TCP Protocol Control Block) and so cannot be accessed directly. To get this information the socket debug (SODEBUG) option was set in the application program to dump this information into a buffer within the kernel (*/dev/kmem*). At the end of the run a program called *trpt* was used to read the information for subsequent analysis.

4.4 Performance Metrics

The following metrics were used in this study:

- **Completion time** is the time elapsed from the setting up of the TCP connection to its tearing down.

- End-to-end **throughput** is the data transfer rate over the life of the connection, and is computed by dividing the total number of data bits transmitted by the completion time.

- The **number of retransmissions** is determined from the trace by counting the number of re-occurring sequence numbers. Since we assume the Ethernet segment to be loss-free, it is assumed that this is a measure of packet loss on the wireless link.
- **Retransmission timeout (RTO)** values are obtained for the last 100 data packets transmitted. An examination of these values gives an indication of TCP's response to perceived congestion in the network.
- **Zero window advertisements** are extracted from the trace of TCP packets. These indicate that the receiver is unable to keep up with the data flow from the sender and tells the sender to stop transmitting data. The sender waits until the receiver sends a **zero window update**. Since zero window updates could fail to arrive, the sender will probe the receiver by means of **zero window probes**. The number of zero window probes is obtained through *netstat*.

4.5 Experimental Design

Since we were interested in the behaviour of TCP for different error rates of the wireless link, this was an important factor in the design of our experiments. We observed quickly that the error rate was a function of the distance between the receiver node (the Powerbook with the wireless interface attachment) and the wireless hub. For our experiments different error rates were obtained by placing the receiver node at different distances from the wireless hub. The distance referred to as "close" was achieved by placing the Powerbook on the same bench, at a distance of roughly .25 m from the wireless hub; the distance referred to as "medium" was achieved by taking the Powerbook roughly 15 m from the wireless hub; and the distance referred to as "far" was achieved by taking the Powerbook roughly 25 m from the wireless hub. The different error rates at these distances are shown clearly by the results in Table 1. We also experimented with a distance greater than 30 m (referred to as "very far"). At this distance a very high error rate caused serious difficulty. In fact, the TCP connection did not terminate, even after 45 minutes. This limited the results that could be obtained from this case. It should be noted that the presence of additional obstacles between the receiver and the hub at long distances (walls, doors, people, etc.) also has an effect on error rates. To provide some measure of the effect of these uncontrolled factors, each run was repeated 5 times, and the values reported are the averages across the five runs.

Range	Completion Time (seconds)		Throughput (Mbits/sec)		Retransmissions
	Average	Std.dev	Average	Std.dev	Average
Close	347.67	9.49	0.228	0.007	4.0
Medium	616.83	42.08	0.130	0.008	31.6
Far	1063.78	40.89	0.078	0.002	71.0

Table 1: Results From the Regular Load

The experiments were conducted during hours of light load in the evenings so that the effect of other network traffic on the wired portion is minimal. In the absence of congestion, we feel it is safe to assume that any packet "loss" occurs as a result of transmission errors on the wireless part of the connection.

4.6 Results

The primary goal of these experiments was to investigate the behaviour of TCP for different error rates on the wireless link. The effect of increasing the rate of errors (by increasing distance) is shown in Table 1. As expected, the loss of packets causes retransmissions to occur and this results in increased completion time and decreased throughput. We believe these effects are due to the loss of packets, but of course the same effects could result from congestion caused by the retransmissions.

TCP's actions with respect to the retransmission timeout variable are shown in Figures 2 and 3, for the medium range case and for the very far case, respectively. Again we see a response consistent with congestion (oscillating behaviour) even though the root cause is packet losses on the wireless link. Especially in the "very far" case, the retransmission timeout values grow very large, effectively shutting down the sender. This is reasonable as a response to congestion, where sending more packets will exacerbate the situation, but it is precisely the wrong response to packet loss. In this case the sender is actually waiting a longer time to retransmit the lost packet instead of trying to recover faster from the loss. In addition, this also affects the windowing mechanism by preventing the window from sliding forward in the data stream.

To investigate the window advertising behaviour, it was necessary to increase the load on the link. This was effected by adding a second connection to run

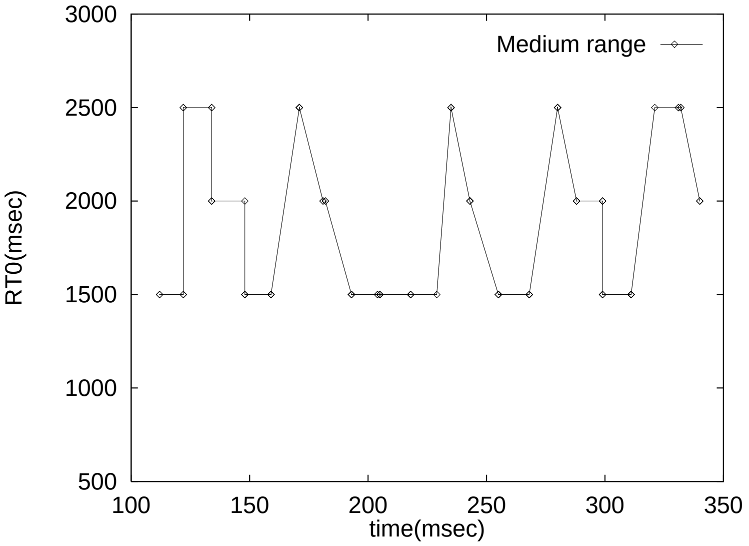

Figure 2: Behaviour of the Retransmission Timeout Variable at the "Medium Range" Distance

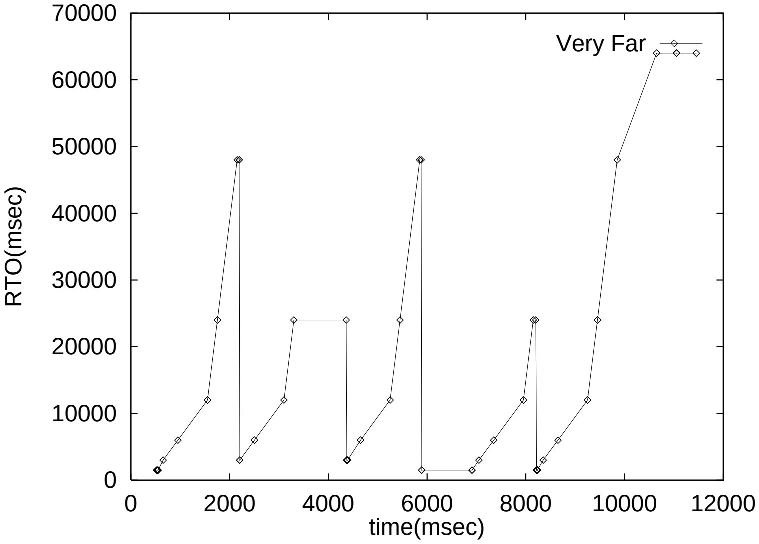

Figure 3: Behaviour of the Retransmission Timeout Variable at the "Very Far" Distance

Range	Completion Time (seconds)		Throughput Mbits/sec		Retransmissions
	Average	Std.dev	Average	Std.dev	Average
Close	776.851	44.561	0.10192	0.0053	4.0
Medium	1363.32	254.588	0.06089	0.0119	63.3

Table 2: Results From the Heavy Load

Range	Zero window advertisements		Zero window probes	
	Light Load	Heavy Load	Light Load	Heavy Load
Close	24.0	108.8	0	0
Medium	52.6	117.6	0	12

Table 3: Windowing Behaviour

simultaneously between the sender and the receiver. Table 2 shows that increasing the load in this way increases the number of retransmissions, and this results in increased completion time and reduced throughput. Table 3 shows that increasing the load results in many more zero window advertisements, indicating that the receiver is having a difficult time keeping up with the sender. At medium range it appears that some of the zero window updates are being lost, since the sender is transmitting zero window probes. Therefore the sender window is staying closed longer than it needs to and the completion time suffers as a result.

5 Conclusions

This is a preliminary study, but the results suggest several things. First, it is clear that TCP is unable to distinguish packet loss on the wireless link (which *did* occur) from congestion in the network (which didn't). TCP reacts in a similar way to lost packets, even though a different response would be more appropriate. Instead of trying to recover from the loss, the sender waits a longer time to retransmit the lost packets. In fact, TCP's response may exacerbate the problem rather than alleviate it.

In summary, we believe that this work provides experimental evidence that additional modifications are needed to the TCP protocol to accommodate the presence of wireless links. Our investigation of these issues is continuing.

References

[1] M. C. Chuah A. DeSimone and O. C. Yue. Throughput Performance of Transport-layer Protocols Over Wireless LANs. In *Proceedings of GLOBE-COM*, pages 542–549, Houston, Texas, November 1993.

[2] A. Bakre and B. R. Badrinath. I-TCP:indirect TCP for Mobile Hosts. In *Proceedings of 15th International Conference on Distributed Computing Systems (ICDCS)*, pages 136–143, Vancouver, BC, May 1995.

[3] Ramon Caceres and Peter B. Danzig. Characteristics of Wide-Area TCP/IP Conversations. In *Proceedings of ACM SIGCOMM '91*, pages 101–112, Zurich, Switzerland, September 1991.

[4] Douglas E. Comer. *Internetworking with TCP/IP : Principles, Protocols and Architecture*. Prentice Hall, second edition, 1991.

[5] Hari Balakrishnan, S. Seshan, Elan Amir and Randy H. Katz. Improving TCP/IP Performance Over Wireless Networks. In *Procceedings of MOBI-COM95, ACM International Conference on Mobile Computing and Networking*, pages 2–12, Berkeley, California, November 1995.

[6] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM '88*, pages 314–329, Stanford, California, August 1988.

[7] J. Postel, editor. *Transmission Control Protocol*, RFC 793, September 1981.

[8] W. Richard Stevens. *TCP/IP Illustrated*. Addison-Wesley Publishing Company, 1994.