Improving the performance of wireless links via end-to-end standard TCP tuning techniques

FRANCESCO PALMIERI
Centro Servizi Didattico Scientifico
Università degli studi di Napoli “Federico II”
Complesso Universitario di Monte S. Angelo, Via Cinthia, 45
NAPOLI - ITALY

Abstract: - The increasing popularity of wireless networks indicates that wireless links will play an important role in future internetworks. Wireless networks are characterized by limited bandwidth and frequent packet losses due to transmission errors and handoffs. Consequently wireless and other lossy links suffer from significant non-congestion-related losses due to the reasons above. In fact, reliable transport protocols such as TCP are tuned to perform well in traditional networks where packet losses occur mostly because of congestion. TCP responds to all losses by invoking congestion control and avoidance algorithms, resulting in degraded end-to-end performance and poor bandwidth usage. In this paper we propose and evaluate a basic framework of end-to-end (IETF) standard tuning techniques to optimize communication in this difficult error environment. We argue and demonstrate that most of these mechanisms, largely available in commercial network equipments, if properly configured and combined can achieve a satisfactory communication efficiency even in a highly variable error environment.

Key-Words: - Internet, Mobile Communication, Wireless links, TCP end-to-end tuning

1 Introduction

Wireless networks are an emerging technology of this decade and have become very popular in last few years. Recent developments in this field demonstrate that mobile computers with wireless communication links will form an integral part of the future networks. This technology provides the user with a wireless equipment unlimited access to information and services like video telephony, e-mail, news, stock quotes etc. Future networks will be a combination of high speed wired networks and wireless links. It is required that this wireless link does not become a bottleneck in the network.

Communication over wireless links is characterized by limited bandwidth, high latencies, high bit-error rates and temporary disconnections that must be dealt with by network protocols and applications. Most of these problems are due to a combination of factors such as absorption, scattering, multipath fading, terrain and environmental factors, and interference from other transmissions.

In addition, protocols and applications have to handle user mobility and the handoffs that occur as users move from cell to cell in cellular wireless networks. These handoffs involve transfer of communication state (typically network-level state) from one base station (a router between a wired and wireless network) to another, and typically last anywhere between a few tens to a few hundreds of milliseconds. Moreover, since conditions change over time (due to mobility or intermittent interference sources), the link transmission error environment will also change. Such a variable, high error environment can create problems for transport protocols and applications.

In this paper we focus on the problem of improving communication in point-to-point generic wireless links, exploring and evaluating some of the most common end-to-end IETF standards track mechanisms (or solutions fully compliant with IETF standards), largely available in commercial network equipment, that are able to provide significant performance enhancements in reliable transport connections and applications.

The rest of this paper is organized as follows. In
Section 2, we describe the most common problems and limitation addressing wireless link performance. In Section 3, we describe the end-to-end tuning techniques that can be used on a point-to-point basis to achieve significant performance improvements on wireless TCP links. Then, in section 4 we describe the experimental test platform and present and analyze the results obtained by the application of the tuning framework on this testbed.

2 Common wireless link problems

In this section we introduce the most common problems, limitation and tradeoffs addressing wireless link performance to give a correct perspective of the scenario in which the proposed performance improvement schemas should be applied.

2.1 Bandwidth

Bandwidth is typically a scarce and highly variable resource in wireless networks. The higher protocol layers may have to take this into consideration and use different methods (e.g. compression) to take care of this problem. Furthermore, the Maximum Transfer Unit (MTU) on a wireless link is much smaller than on wired network, consequentially, the use of small packets leads to under-utilization of available bandwidth, furtherly reducing the throughput of a connection [1].

2.2 Mobility

Wireless hosts may move frequently while communicating. During this movement the data sent to mobile wireless hosts may be lost. The TCP/IP stack at the destination often interprets this loss as congestion and invokes congestion control mechanisms, which is unnecessary, because when the movement is complete the wireless host will start receiving data again. This causes the performance of the connections to degrade. There also might be degradation of performance due to frequent recalculation of routes to the moving wireless host.

2.3 Error Rate

Communication over fixed wireless links is often characterized by sporadic high bit-error rates. The Bit error rate on wireless links is usually found to be about 10 or worse, in contrast with an optical fiber link that exhibits a typical error rate of about $10^{-12}$ [2]. Wireless hosts use radio transmission or infrared wave transmission for communication. This mode of communication is vulnerable to interference from the environment that generates frequent transmission errors often causing burst of packets to be lost. Unfortunately, when packets are lost in networks for reasons other than congestion, the congestion mechanisms provided in higher layer protocols such as TCP, that can not distinguish the exact cause of the packet loss, works very poorly causing severe throughput degradation and very high interactive delays. In detail, reliable transport protocols such as TCP have been tuned for traditional networks made up of wired links and stationary hosts. TCP performs very well on such networks by adapting to end-to-end delays and packet losses caused by congestion. TCP provides reliability by maintaining a running average of estimated round-trip delay and mean deviation, and by retransmitting any packet whose acknowledgment is not received within four times the deviation from the average. Due to the relatively low bit-error rates over wired networks, all packet losses are correctly assumed to be because of congestion. In the presence of the high error rates and intermittent connectivity characteristic of wireless links, TCP reacts to packet losses as it would in the wired environment: it drops its transmission window size before retransmitting packets, initiates congestion control or avoidance mechanisms (e.g., slow start [3]) and resets its retransmission timer (Karn's Algorithm [4]). These measures result in an unnecessary reduction in the link's bandwidth utilization, thereby causing a significant degradation in performance in the form of poor throughput and very high interactive delays [5].

3 End-to-end TCP tuning

In this section, we briefly describe the standard tuning techniques that have been proposed to improve the performance of TCP over wireless links.

3.1 Header Compression
As bandwidth is very limited in wireless channels, a great effort has to be done in order to optimise its use. One method to decrease the used bandwidth is to reduce the headers of the network protocols that are sent to the channel. In the particular case of this study, we have evaluated the Van Jacobson algorithm [6] to reduce the TCP/IP headers.

The advantages of the header compression are:

- The reduction of the response time in interactive applications. A typical TCP/IP header of 55 bytes, will be reduced to 3 or 5 bytes and consequently the response time in application sending single or few characters in each segment is substantially reduced.
- Header compression contributes to independence the segment size and the achievable throughput for a connection, which is important when interactive and bulk data transmissions are concurrent.
- As in wireless environments the MTU is small due to the need of minimise the Packet Loss probability, header compression seems to be appropriate to reduce the packet loss and optimise the bandwidth usage.
- Moreover, the new version of the TCP/IP Protocol (IPv6) has a header length of 60 bytes due to the increase in IP addresses. Also, IP includes some options to allow host mobility and facilitate datagram routing, which furtherly increases the TCP/IP header length [7].

Consequently, Van Jacobson header compression can be surely appropriate in TCP/IP over wireless links, and this feature will be always enabled on our experimental testbed combined with the other techniques to obtain better results in terms of TCP performance.

3.2 Path MTU Discovery

Path MTU Discovery [8] is used to determine the maximum packet size a connection can use on a given network path without being subjected to IP fragmentation. The sender transmits a packet that is the appropriate size for the local network to which it is connected (e.g., 1500 bytes on an Ethernet) and sets the IP "don't fragment" (DF) bit. If the packet is too large to be forwarded without being fragmented to a given channel along the network path, the gateway that would normally fragment the packet and forward the fragments will instead return an ICMP message to the originator of the packet. The ICMP message will indicate that the original segment could not be transmitted without being fragmented and will also contain the size of the largest packet that can be forwarded by the gateway.

Path MTU Discovery allows TCP to use the largest possible packet size, without incurring the cost of fragmentation and reassembly. Large packets reduce the packet overhead by sending more data bytes per overhead byte. Increasing TCP's congestion window is segment based, rather than byte based and therefore, larger segments enable TCP senders to increase the congestion window more rapidly, in terms of bytes, than smaller segments.

The disadvantage of Path MTU Discovery is that it may cause a delay before TCP is able to start sending data. For example, assume a packet is sent with the DF bit set and one of the intervening routers returns an ICMP message indicating that it cannot forward the segment. At this point, the sending host reduces the packet size per the ICMP message returned by the router and sends another packet with the DF bit set. When the packet can be transmitted without fragmentation it will be forwarded to the next hop, but this does not ensure all subsequent routers in the network path will be able to forward the segment. If a second router cannot forward the segment it will return an ICMP message to the transmitting host and the process will be repeated. Therefore, path MTU discovery can spend a large amount of time determining the maximum allowable packet size on the network path between the sender and receiver. Wireless, and particularly satellite, delays can aggravate this problem (consider the case when the channel between the two routers is a satellite link). In practice, Path MTU Discovery does not consume a large amount of time due to wide support of common MTU values. Additionally, caching MTU values may be able to eliminate discovery time in many instances, although the exact implementation of this and the aging of cached values remains an open problem.

However, since the relationship between Bit
Error Rate and segment size is likely to vary depending on the error characteristics of the given channel, the use of Path MTU Discovery is absolutely recommended to allow TCP to use the largest possible MTU over the wireless channel.

### 3.3 Selective Acknowledgements

The standard TCP acknowledgment scheme is cumulative. If a segment is lost, TCP senders will retransmit all data sent starting from the lost segment without regard to the successful transmission of later segments. TCP considers this lost segment as an indication of congestion and reduce its window size by half. Recently the newly defined standard TCP-SACK (Selective ACKnowledge) [9] allows the receiver to explicitly inform the sender of the loss telling exactly which packets have arrived. Consequently, a sender can retransmit the lost segments immediately rather than waiting for a timeout, reacting to supposed congestion, and multiplicatively decreasing its window.

In detail, SACK allows TCP to recover more quickly from lost segments, as well as avoiding needless retransmissions. The fast retransmit algorithm can generally only repair one loss per window of data. When multiple losses occur, the sender generally must rely on a timeout to determine which segment needs to be retransmitted next. While waiting for a timeout, the data segments and their acknowledgments drain from the network. In the absence of incoming ACKs to clock new segments into the network, the sender must use the slow start algorithm to restart transmission. As discussed above, the slow start algorithm can be time consuming over wireless channels. When SACK is employed, the sender is generally able to determine which segments need to be retransmitted in the first RTT following loss detection. This allows the sender to continue to transmit segments (retransmissions and new segments, if appropriate) at an appropriate rate and therefore sustain the ACK clock. This avoids a costly slow start period following multiple lost segments.

Generally SACK is able to retransmit all dropped segments within the first RTT following the loss detection. If lost segments are not caused by congestion, or the congestion is transient, throughput in TCP-SACK should be much better. This will be very helpful in wireless networks because anything that triggers timeouts and window size reduction will force a lengthy recovery in TCP.

### 3.4 TCP Window size

Often, the standard maximum TCP window size (65,535 bytes) is not adequate to allow a single TCP connection to utilize the entire bandwidth available on some wireless channels. TCP throughput is limited by the following formula [10]:

\[
\text{throughput} = \frac{\text{window size}}{\text{RTT}}
\]

Under these assumptions, it may happen that a single standard TCP connection cannot fully utilize the bandwidth of a high speed wireless link such as a satellite or a very high frequency microwave link. TCP has been extended to support larger windows [11] and the window scaling options outlined in [11] should be used in huge bandwidth-high latency wireless environments. Of course for a wireless link constantly shared among many flows, large windows may not be necessary.

This optimisation often requires both client and server applications or TCP stacks to be hand tuned (usually by an expert) to use large windows, allowing TCP applications to better utilize the capacity provided by the underlying wireless link.

### 4 Experimental evaluation

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 measurement results obtained with the various tuning schemes.

#### 4.1 The experimental testbed

The experimental network topology we used to evaluate the TCP tuning features in the point-to-point wireless environment is shown in Figure 1 below.
A minimal low speed/low quality (115.2 Kbps, Siemens Gigaset M101, to easily observe transmission errors and congestion effects) wireless point-to-point link, has been set-up between two Cisco 2610 routers. PPP protocol has been used for the link encapsulation. An artificially simulated constant bit rate TCP traffic flow (long connection - file send) generated using the Chariot [12] tool from NetIQ, traverse the experimental backbone, starting from a first endpoint station, directly wired via 10Mps Ethernet to the first router and reaching an endpoint station wired in the same manner to the other one. The wireless link has been installed in an high density computer room to easily introduce real-environment problems (errors etc.) due to the high electromagnetic interference levels usually present in such environment. Detailed performance measurements have been collected for the TCP stream, first flowing through the backbone with no tuning feature enabled and then progressively with VJ header-compression, Path MTU discovery, tuned TCP window size and Selective-Acknowledge combined together.

4.2 Results analysis
We have conducted the simple set of experiments described above to explore the correlation among VJ compression, TCP Sack, window size, PMTU discovery, network bandwidth, latency, and end-to-end performance (response time and throughput). We also fairly saturated the interconnection link bandwidth to create the worst conditions, but not exacerbating the packet loss, and thus easily observe the effects of the TCP tuning techniques on the maximum achievable link throughput.

We recorded the throughput and response time of each TCP connection. Throughput determines the bandwidth utilization of the link from the system manager's point of view, while the response time reflects performance as perceived by the user. Some of the significant results have been reported in Fig. 1 and Fig. 2 below:

At a first glance it can be seen, as asserted before, that there are significant performance benefits in using the VJ Header Compression combined with the other tuning mechanisms.

PMTU discovery, dynamically detecting an optimal packet size for the whole backbone, substantially reduces the disruptive effects of packet loss on throughput and interactive response time.

TCP Window size also affect the overall performance, but only if finely hand-tuned to find the optimal value (we tested WSIZE values varying from 2K to 64K). In the results obtained in our testing environment with a single persistent TCP connection, the window size should matter most. Finally, as can be noted from the reported graphs, in case of long burst of errors the use of compression and large window size produces, as a trade-off, the retransmission of the whole transmission window, generating perceivable falling peaks in throughput and consequential increments in response time.
TCP SACK performed slightly better, when combined with the above techniques with more stable improvements on end-to-end performance clearly due to SACK benefits on typical burst packet loss.

5 Conclusion

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. None of these solutions is perfect but some interesting results may be obtained by the combination and fine-tuning of the above techniques.

The use of a compression method of the TCP/IP encapsulated information may have positive effects when the used communications link is the mobile radio channel. Packet size optimal tuning via PMTU discovery usually decreases the response time in interactive applications, reduces packet loss probability and increases the bandwidth efficiency. Moreover, the transmission window size when burst errors occur can heavily tax the achieved throughput. TCP SACK, allowing the sender to recover from multiple packet losses in a window, without resorting to a coarse timeout achieves better and more stable performance on the wireless link throughput.

References:


