Improving TCP Performance Over Wireless Links

Samir Goel
Computer Science Department
Rutgers University, NJ, USA
gsamir@paul.rutgers.edu

Dheeraj Sanghi
Computer Science & Engg. Department
Indian Institute of Technology Kanpur
dheeraj@iitk.ac.in

Abstract

TCP is the most common transport protocol used in Internet. It was designed primarily for wired networks. The characteristics of wireless links is very different from wired links, particularly in terms of loss behaviour. This results in poor TCP performance over wireless links. In this paper, we have proposed enhancements to TCP so that it can differentiate between a loss due to congestion, and a loss due to noise in the wireless link. With this knowledge, TCP can react differently to the two kinds of losses, leading to improved performance. The modified TCP is compatible with existing implementations.

1 Introduction

Wireless links have fundamentally different characteristics than wired links. They are characterized by low bandwidth and error rates that are high, bursty and time-varying. This difference in error characteristics causes a significant degradation in the performance of TCP that was originally developed for wired networks. For example, TCP misinterprets packet loss over a wireless link as a sign of network congestion, causing poor throughput.

The problem has been studied by several researchers, and many solutions have been proposed [1, 2, 6, 7]. Nanda *et al.* [6] have suggested introducing reliability at the link layer for wireless link using finite number of retransmissions. This approach does not completely shield the source transport layer from all losses on the wireless link. Also, studies [4] have shown that link-layer retransmissions may interfere with TCP's end-to-end retransmissions leading to degraded performance.

Bakre *et al.* [1] and Yavatkar *et al.* [7] advocate splitting end-to-end TCP connection into two separate TCP connections - one over the wired network and the other over the wireless link, with base station serving as the common point for two connections. This approach isolates the transport layer from the erratic behavior of the wireless link. However, this approach does not preserve the semantics of TCP. Also, every packet incurs the overhead of going through the TCP protocol processing twice at the base station. Further, it requires the base station to maintain state for every TCP connection passing through it.

Balakrishnan *et al.* [2] have suggested adding a module called snoop agent to the routing code at the base station. This agent monitors packets flowing on a TCP connection. It maintains a cache of unacknowledged TCP packets on a per-connection basis and performs local retransmission when it detects a packet loss. This approach does not completely shield the sender from losses on a wireless link. Also, it assumes that a wireless link is the last hop in the network path. It also requires a base station to maintain state information and a cache of unacknowledged packets for every TCP connection passing through it.

All proposed solutions are based on the assumption that no changes can be made to existing TCP implementations. However, we note that it is possible that newer hosts run modified TCP implementations. We need to only ensure that the modifications should be backward-compatible. And that the older hosts should not notice a degradation of performance.

In this paper, we propose to augment TCP to respond to control signals sent from wireless gateway nodes. The control signals are such that they are unambiguously discarded by the receiving hosts running older version of TCP. The simulation results show that the scheme provides substantial performance improvement with low cost overhead.

2 A Simple Scheme

We first propose a simple solution. We conduct simulation experiments to show that it works in most circumstances, but our results also show that it does not improve performance under certain conditions. We pro-

pose a refinement to our basic scheme in the next section.

The basic idea is that if TCP can distinguish a packet loss due to congestion from that due to channel noise, algorithms can be developed to deal with such losses efficiently. In this paper, we make the assumption that losses in the wired network are due to congestion, while those in the wireless network are related to noise. These are the same assumptions made by others.

In order that TCP be able to differentiate between two kinds of losses, the network needs to generate some control information and send it to the source TCP entity. We can use ICMP for this purpose. We propose a new message type within ICMP to carry the required information. This has the additional advantage that nodes which do not understand this message-type will simply discard the message. The new message type is called ICMP-DEFER.

2.1 Description of a Simple Scheme

In the proposed scheme, the base station generates an ICMP-DEFER message when the first attempt at transmitting the packet on the wireless link is unsuccessful. This policy ensures that within one round trip time, TCP will receive either an acknowledgment or an ICMP message. This will ensure that end-to-end retransmissions do not start while link-layer retransmissions may be going on. A lack of both will signal a congestion loss. Thus, TCP can distinguish between two kinds of losses. The control information consists of TCP and IP headers (typical ICMP message). This is enough for source TCP to decide which particular packet was lost on the wireless link. ICMP-DEFER messages are not retransmitted, thus the overhead on the network is minimal.

TCP performs the following actions on receipt of an ICMP-DEFER message. If retransmit timer is set for the segment indicated in the ICMP message, it postpones its expiry by the current estimate of retransmission timeout (RTO) value. This will avoid conflict between the link-layer retransmissions and end-to-end retransmissions. We assume that one RTO time is sufficient for the base station to exhaust all local retransmission attempts for a packet. If the wireless link remain in error-state for longer duration, TCP would stop transmission because there would be no acks, and thus send window would not move.

If a segment needs to be retransmitted, TCP checks if it had received an ICMP-DEFER message for this segment. If it had, it would retransmit the segment but it does not reduce the *cwnd*. Also, when coming out of fast recovery algorithm, it resets *cwnd* back to its value before fast retransmission and fast recovery.

2.2 Simulation Experiments

We conducted several simulation experiments to test our solution. We describe our setup, the parameters for the simulation, and present the results below.

2.2.1 Setup

We used a simple network topology for conducting simulation experiments, as shown in Figure 1. The link *L1* is representative of the entire path over the wired network. The propagation delay of this link represents the sum of propagation delay, queuing delay and the processing time of each node in the path. We assume that the wired link is reliable. We further assume that the base station does not drop packets because of insufficient buffers.

Figure 1. Configuration used in Simulations

We simulated wireless link by a 2-state Markov model with the stay period in each state (GOOD, BAD) characterized by exponential distribution [3]. The propagation delay of the wireless link is representative of the processing time at the nodes as well.

2.2.2 Parameters

The following parameters are of interest in the simulation: Packet size, size of file transfer, and receiver's buffer size (or advertized window size). For each link, we could set bandwidth and propagation delay. For wireless link, the loss characteristics was described by two states: GOOD and BAD. We set the duration of each state and the loss probability in that state. Duration was assumed to be exponentially distributed.

Varying the mean value of duration and loss probability for GOOD and BAD states allow us to simulate wireless link of different error characteristics. Varying the receiver's advertised window allowed us to control the maximum amount of data that may be in-transit. Varying the propagation delay of the wired link allowed us to simulate various kinds of networks. A WAN can be simulated by keeping the propagation delay value large. A small value simulates a LAN.

Table 1 shows the values of various parameters that we used in the simulation experiments described in this section.

Parameters	Value
Wired link	
- Bandwidth	8 Mbps
- Propagation delay	492 ms.
Wireless link	
- Bandwidth	1 Mbps
- Propagation delay	1 ms.
- Error characteristics -	
Λ GOOD state (expo. distr.)	
- mean value	5 seconds
- loss probability	0 (reliable)
Λ BAD state (expo. distr.)	
- mean value	0.1 second
- loss probability	1
Packet size	1 KB
Size of file transfer	10 MB
Receiver's adv. window size	16 to 64 KB

Table 1. Parameter values used in simulations

2.2.3 Results

We have conducted several experiments. A detailed discussion of them is available in [5]. In this paper, we present only the most interesting results. The performance criterion we chose to compare our scheme with existing TCP was throughput.

In one set of experiments, we compared the performance of TCP over wired network *No loss case*, normal TCP in presence of a wireless link *Unmodified TCP*, and TCP with our enhancements *Simple Scheme*. The experiment was repeated for different window size to control the amount of data in-transit. The propagation delay is set to 492 milli-seconds to highlight the effects of the clash between link layer retransmissions and end-to-end retransmissions.

Results show that there is degradation in performance of *Unmodified TCP* and *Simple Scheme* over the *No loss case*. Further, we see that the degradation in performance of *Unmodified TCP* increases with the increase in window size. This is because it treats losses on the wireless link as signs of congestion and reduces its *cwnd* and *ssthresh*. The effect of this reduction becomes more prominent at larger values of window size. The performance of *Simple Scheme* does not suffer much degradation.

For figures, and more discussion, refer to [5].

2.3 Problem

In *Simple Scheme*, TCP reacts to a packet loss on a wireless link by retransmitting the packet while not reducing its *cwnd* and *ssthresh*). This results in substantial performance improvement. But, it also introduces a problem. For every packet lost on the wireless link (that is, all local retransmissions failed), TCP now recovers it by suffering either a fast retransmit or a retransmission timeout. Since, fast retransmit and fast recovery algorithms were designed to recover from single packet loss, in case when multiple packets in a transmission window are lost on the wireless link, TCP recovers them by suffering retransmission timeouts as many times as the number of losses. This is in contrast to unmodified TCP which recovers from multiple consecutive packet losses in a window by suffering at most one retransmission timeout.

In Simple Scheme, this large delay in recovering from multiple packet losses, detracts the advantage gained by not reducing the *cwnd* and *ssthresh*. Note that this situation is not specific to a particular value of window size.

3 Refined Scheme

In this section, we propose refinement to the Simple Scheme for better performance.

To solve the problem mentioned in previous section, we decided that the base station should also inform TCP about losses. The base station generates an ICMP message when all the local retransmission attempts have been exhausted. We refer to this as ICMP-RETRANSMIT message. Note that we still need ICMP-DEFER messages to avoid conflict between end-to-end and link level retransmissions.

The TCP response to two types of ICMP messages is as follows. On receiving an ICMP-DEFER message, TCP postpones the retransmission timer if the lost segment is the one for which the timer is active. When TCP receives an ICMP-RETRANSMIT message, it retransmits the segment indicated.

An ICMP-RETRANSMIT message indicates that one segment was lost. When the destination receives subsequent packets, it generates duplicate acks. When the source TCP receives the first of such duplicate acks, it switches to fast recovery algorithm. When it finally receives a new ack, it comes out of the fast recovery algorithm and resets *cwnd* to the value prior to its entering the fast recovery phase.

When source TCP suffers a retransmit timeout, or receives duplicate acks for a segment for which it has not received ICMP-RETRANSMIT message earlier, it follows normal TCP algorithms.

3.1 Simulation Results

We conducted simulation experiments for different values of propagation delay of the wired link and for different error characteristics of the wireless link.

We compared time taken for file transfer for various window sizes. The four schemes compared are: *No Loss Case*, *Unmodified TCP*, *Simple Scheme*, and (*Refined Scheme*).

We note that the *Refined Scheme* performs better than *Simple Scheme* for all values of window size. The percentage improvement of *Refined Scheme* over *Unmodified TCP* reaches 28.96%.

We found that as the duration of BAD state reduces, the improvement of *Refined Scheme* over *Unmodified TCP* is not significant. This is because the number of losses has reduced. The performance in this case is anyway close to that of *No Loss Case*.

The performance of *Simple Scheme* is, many a times, worse than *Unmodified TCP*. This happens especially at higher values of `mean_bad_state`. This is because when the duration of BAD state increases, the possibility of multiple losses increases. As we have discussed earlier, *Simple Scheme* is unable to recover from multiple losses in an efficient fashion.

For detailed results and figures, refer to [5].

3.2 Advantages of Refined Scheme

Below, we summarize the advantages of our scheme.

- Modified TCP performs significantly better than existing TCP over network paths involving wireless links. At the same time, there is no difference in performance when used in wired networks.
- Overhead of the scheme is very low. During simulation runs involving a 10MB file transfer (10,000 packets transfer), the overhead in terms of number of packets mostly remained less than 1%. Overhead in terms of bandwidth used was significantly lower, since ICMP packets are very small.
- ICMP messages are silently discarded by hosts which do not implement these enhancements.
- Our scheme preserves the semantics of TCP.
- In our scheme, generation of ICMP messages is triggered by the link layer for the wireless link. So, in the case when a mobile host is acting as the source of packets, ICMP messages will be triggered by the link layer of the mobile host. This policy makes our scheme symmetric.
- The scheme does not make any assumption about the location of the wireless link in the network path.

4 Conclusions

In this paper, we have proposed an ICMP based scheme to improve the performance of TCP over network paths involving wireless links. In our scheme, we have proposed enhancements to TCP to respond to control information from the network. This enables TCP to differentiate between losses on a wireless link and those on the wired network, resulting in significant improvement in performance. The control information is generated by the base station, and is carried to the source using new message-types in ICMP.

The simulation results show that the proposed scheme achieves significant improvement in performance over unmodified TCP. The results also show that the scheme is robust in face of burst errors on the wireless link and its performance gain improves with the increase in frequency of burst errors on the wireless link.

The scheme is compatible with existing TCP implementations, and there is no difference in performance when used in wired networks.

References

[1] A. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *Proc. 15th Int'l Conf. on Distributed Computing Systems (ICDCS)*, May 1995.

[2] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP performance over wireless networks. In *Proc. 1st ACM Int'l Conf. on Mobile Comp. and Networking (Mobicom)*, November 1995.

[3] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. K. Tripathi. Enhancing throughput over wireless LANs using channel state dependent packet scheduling. In *Proc. IEEE INFOCOM'96*, March 1996.

[4] A. DeSimone, M. C. Chuah, and O. C. Yue. Throughput performance of transport protocols over wireless LANs. In *Globecom'93*, December 1993.

[5] S. Goel. Improving the performance of TCP over wireless links. Master's thesis, Indian Institute of Technology, Kanpur, May 1997.

[6] S. Nanda, R. Ejzak, and B. T. Doshi. A retransmission scheme for circuit mode data on wireless links. *IEEE J. Sel. Areas Comm.*, 12(8), October 1994.

[7] R. Yavatkar and N. Bhagwat. Improving end-to-end performance of TCP over mobile internetworks. In *Proc. Mobile '94*, December 1994.