Contents

Extending TCP for Wireless Networks

Suman Banerjee *Janaki Ram Goteti*
Ganapathy Krishnamoorthy
Department of Computer Science
University of Maryland, College Park
{suman,gjr,gansub}@cs.umd.edu

May 21, 1997

Abstract

Reliable transport protocols like TCP have been adapted to work well for traditional wireline networks where packet losses occur mostly due to congestion. The implicit assumption in these networks is that the links are lossless, and hence they do not cause significant amounts of packet corruption. TCP responds to packet losses by invoking some congestion control and avoidance techniques. However, such an assumption is not valid for wireless networks where packets may also be lost due to channel noise. In this report we suggest modifications to TCP to improve end-to-end delay and latency characteristics in wireless networks, while maintaining end-to-end semantics. Through simulations we compare the performance of unmodified TCP and compare its performance for wireless networks to TCP extended with our modifications.

1 Introduction

Recent activity in mobile computing and wireless networks give a strong indication that mobile computers with wireless communication links will be an integral part of future internetworks. Communication over wireless links is characterized by limited capacity, sporadic high bit-error rates and temporary disconnections. The problem of mobile hosts with unreliable channels connecting them to their base stations has been attacked by various researchers in different levels.

Several Mobile-IP proposals [4,5] have addressed the problem of delivering IP packets to mobile hosts regardless of their location. In theory, one can use existing fixed network transport protocols such as UDP and TCP on the mobile hosts to communicate with the fixed network. Such proposals attempt to keep the mobility, disconnection and other features of wireless and mobile hosts transparent to the layers above the network layer, which might lead to degraded performance at these layers. On the other hand, use of a new protocol stack leads to interoperability problems.

One other approach has been to been to hide any non-congestion related losses from the TCP sender by making the low quality wireless link appear as a high quality one. The intuition behind this approach is that since the problem is local, it should be handled locally. Example of this approach include wireless links with reliable link-layer protocols such as AIRMAIL [1]. In this mechanism, the transport layer need not be aware of the characteristics of the individual links. The link quality is improved by repeated re-transmissions in the link layer level of corrupted or lost frames, or using more bits for the frame error checksum. However, since TCP uses its own end-to-end retransmission protocol, such independent re-transmission protocols might lead to degraded performance, specially as error rates become significant [3].

Similar work has been reported in the implementation of I-TCP [2], where a TCP connection is split into two separate interactions – on between the mobile host and its support router (base station) over the wireless medium, and the other between the base station and the stationary host across a wire-lined network. This allows the separation of flow control and congestion control on the wireless link from that on the fixed network. This separation is desirable because of the vastly different characteristics of the two kinds of links – the fixed links are becoming faster and more reliable every day, where as the mobile, wireless link is very slow and is extremely vulnerable to noise and loss of signal due to fading which result in high error rates. The problems associated with this approach include loss of semantics – where packets may be acknowledged before they are received by the mobile host, and software overhead – a packet needs to go through the TCP protocol stack and the associated overhead four times, once at the sender, twice at the base station and once at the mobile host.

2 TCP Congestion Control

TCP uses a round-trip delay estimate for its adaptive windowing scheme to transmit data reliably over an unreliable network with time varying bandwidth. An estimate of the round trip time is maintained as a smoothed moving average [6] using the relation –

$$srtt = \alpha \times srtt + (1 - \alpha) \times rtt$$

Similarly a smoothed variance (estimated as mean difference to avoid square root calculations in the kernel) is also maintained (Tahoe TCP). A initial re-transmission timeout is then set as –

$$timeout = srtt + 2 \times variance$$

If an acknowledgement for a segment is not received within the timeout, it is re-transmitted.

TCP uses a congestion window in the sender side to do congestion avoidance. The congestion window indicates the maximum amount of data that can be sent out on a connection without being acknowledged. TCP detects congestion when it fails to receive an acknowledgement for a packet within the estimated timeout. In such a situation, it decreases the congestion window to one maximum segment size (MSS), and under other cases it increases the congestion window by one MSS. There also exists a congestion window threshold, which is set to half the congestion window size at the time when a re-transmit was required.

The inherent assumption in this mechanism is that lack of an acknowledgement is due to network congestion. If a packet, however, is lost by the network for reasons other than network congestion, then waiting for the timer to run out is wasteful. This is a situation that may happen quite frequently in wireless networks, and so to improve TCP performance, it is needed to pre-empt re-transmissions before waiting for the timer to run out.

To guard against this scenario, Reno TCP [7] uses Fast Re-transmit and Fast Recovery algorithms. Both these algorithms depend on counting duplicate acknowledgements sent by the data receiver in response to each additional segment received following some missing data. Fast Re-transmit detects loss of a segment when three duplicate acknowledgements are received, and re-transnits it. Fast Recovery algorithm attempts to estimate how much data is outstanding in the network by counting duplicate acknowledgements.

3 Design Assumptions

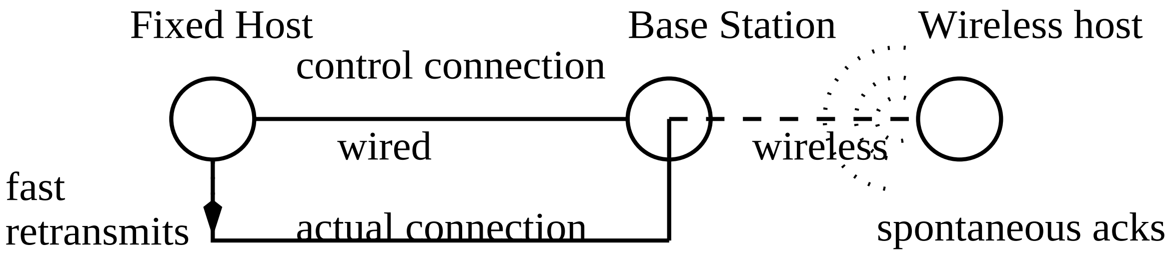
The basic wireless model that we assume for our analysis and design comprises of a source host which is connected to a base station by a fixed wireline network, and a wireless host, which is just a single hop from the base station. We also assume that the mobile, wireless host receives all its packets through the base station only (ie. it has no other network connectivity). This assumption is not overtly optimistic in the light of the suggestions made in Mobile IP extensions [8] where each mobile host is maintained by a *foreign* or a *home agent* in its geographical location, which is responsible for forwarding all packets to it.

Any packets lost between the source and the base station is solely due to network congestion, and any packets dropped by the wireless link is due to the channel noise and bit-error rates. For our purposes, we make the assumption that the connecting base station of the wireless host does not change. In this project we do not look at the changes that are needed to ensure proper handoff between base stations. Also, we assume that the source is aware that the other end of the connection is a wireless host. This helps in making better optimizations since the higher layers (like TCP) can now adapt using this extra knowledge.

4 TCP Extensions

We make two basic design modifications in TCP for the purpose of better throughput when connected to mobile hosts. The changes at the source side is to ensure that the source can distinguish between congestion in the fixed network and noise in the wireless link as the cause for packet loss. The changes in the receiving side (the wireless host) is to ensure that in case packets are dropped due to noise in the wireless channel, the source starts to re-transmit packets earlier than its timeout deadline, for which it normally waits.

When a TCP connection is to be opened between a fixed host on a network and a wireless host, it opens two independent connections – one to the base station and the other to the mobile host. The call setup functionality is left unchanged as is in original TCP. The connection between the source and the base station basically serves as a control, to estimate the network congestion that occurs between the two ends. With the assumption that the mobile host receives all packets through the base station, we expect that packets on both these connections should be routed by almost the same route. and would see the same network congestion. Packets on the control



Model of Proposed TCP Extensions

connection are sent at some regular interval, but at rates sufficiently low so as to not cause too much overhead in the network because of these control packets. The TCP source will also be sending packets on the connection with the mobile host as the application hands down packets to it. By comparing the fraction of the packets that are acknowledged within the timeout on the two connections the source will determine if there is congestion in the network which is causing packets to be lost, or is it the channel noise. If the acknowledged fraction is *significantly* different in the two cases, then the conclusion is that the channel noise is causing the packet loss. In this situation, the source will be prevented from reducing the congestion window size, and it will continue its linear increment at the same rate. If the two fractions are the same, then the normal congestion control mechanisms are allowed to proceed as it would in ordinary TCP.

At the wireless host side, to pre-empt the source to re-send packets that are lost due to the channel noise (or even otherwise), we employ the technique of fast re-transmissions [9]. The source on seeing a duplicate acknowledgement decides that its previous packet sent have been lost or corrupted. Hence, it re-transmits the subsequent packet even before the timer expires and re-starts the timer. On sending a packet the source at time t_s, it waits upto time $(t_s + sRTT + \frac{variance}{2})$ before recognizing another duplicate acknowledgement for a subsequent re-transmission. The need to await for at least the sRTT time is to ensure that these new re-transmissions had a chance to reach the destination before another re-transmission is done.

Also, to ensure that the duplicate acknowledgements manage to get through the wireless channel even under poor channel quality, the wireless host sends multiple duplicate acknowledgements on receiving a new packet at time t_r. This period extends from the time a new packet is received upto time $(t_r + timeout - sRTT)$ more. Any acknowledgements sent after this time will reach around the timeout period, and the source will automatically re-transmit after that, and so further acknowledgements are wasted.

If no further packets are received upto $(t_r + timeout)$, further automatic acknowledgements are made upto time $(t_r + 2 \times timeout - sRTT)$. Hence, it absence of further packets being received, automatic acknowledgements are sent in regular intervals in the periods $[(t_r + k \times timeout)$, $(t_r + k \times timeout - sRTT)]$, where t_r is the time when the last packet was received by the wireless host. The value t_r is reset each time a new packet is received.

The wireless host continuously receives the values of timeout and sRTT from the source, since it needs the estimation of the source for both these values to figure out its re-transmission period.

5 Modeling the wireless channel

The wireless channel between the base station and the wireless host is modeled to have two states – good and bad. In the good state, the channel is a constant and low bit-error probability. In the bad state, the channel error is in general a complex-value random variable. Its amplitude is given by a Rayleigh distribution [10], with the cumulative probability distribution being given by –

$$P(r \leq R) = 1 - e^{\frac{-R^2}{2\sigma^2}}$$

and its phase is always uniformly distributed between 0 and 2π.

For our purposes, it is however, sufficient to model it as having a high error probability in the bad state. Hence, for our model, the wireless channel is modeled as a two-state Markov modulated process, with one a low error probability and the other a high error probability state, which corresponds to stable high-quality channel conditions and fade conditions.

6 Implementation

For the purpose of this project, we implemented a discrete event simulator using Java. In the simulator, we have events corresponding to packet and acknowledgements sent and received. In each such event further events are scheduled in the future. The program generates a trace of packets received at the wireless host, and notes the corresponding time at which they are received. The source generates packets at a constant rate, which is sent out for the wireless host. To simulate the performance of the extended TCP, we study the performance under all levels of network congestion. The network load (network congestion parameter) is made to vary between low load, and peak load in a quasi-periodic way, with a mean period of 1 second. It must be noted that the period of variation is not very relevant, as long as the period is not too less for the source estimate of the round-trip time to somewhat stabilize, since the basic aim is to study whether the extended TCP can distinguish between network congestion and the channel fade periods.

The smoothing parameter for the sRTT (α) is taken to be 1/8 and that for variance is taken to be 1/4 (as is done in the implementation of the BSD TCP stack).

The packet delay statistics between the source and the base station is taken as a function of the current network congestion, which more specifically is given by –

$$\text{packet delay for a specific packet} = \text{base packet delay} + A \times \text{network congestion parameter} + p$$

where p represents the perturbation in the delay, and is given by a Gaussian distribution, and A is an arbitrary constant. The base packet delay is an estimate of the mean delay of packets to travel between the source and base station, and for our example plots have been chosen at 40 ms. The network congestion parameter results in the un-perturbed packet delay to vary between 30 ms and 50 ms, while the perturbtion adds some randomness in the delays. The network congestion parameter is a time dependent function that is used by the simulator to determine the actual packet delay for a packet being delivered at that time. To simulate peak load and congestion, we keep the time to live in the packets to be less than the maximum delay that is possible.

The propagation and transmission delay over the wireless link is assumed to be 2 ms, and the ratio between the good and the bad states is 100:1.

The source makes its sRTT and variance estimate using the round trip delay characteristics of each packet, that is acknowledged before the timer runs out.

7 Performance Characteristics

The throughput of the modified TCP is observed to be an improvement over the unmodified TCP (Refer to the plots). It may be noticed that the wireless noise is sufficient cause around a mere 20% throughput. The plots show that in the modified TCP (the darker plot) packet transfer occur in larger bursts than the unmodifed TCP case. This is in tune with our expectations, since the congestion window for our modified TCP is prevented from decreasing to 1 MSS each time a packet is lost due to channel noise. The same is not true for unmodified TCP.

Also, the graphs show a very periodic zig-zag pattern which basically follows the periodic packet loss due to the wireless channel error. The congestion window size in unmodified TCP drops to one in a periodic fashion, as a packet is lost by the wireless channel. Hence, there are no large bursts of packet transmissions in the unmodified TCP graphs.

8 Conclusions

As can be observed, there has been some significant performance improvement using our suggested modifications. The results do indicate that the mechanisms that we have suggested are worth pursuing, to lead to an improvement of TCP performance for wireless networks.

To get better estimates of the of the modified system, it might be useful to actually change the kernel TCP implementation and then compare performance of unmodifed and modified TCP performance.

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 Wireless Networks Journal, 1, February 1995.

[2] A. Bakre and B.R. Badrinath. *I-TCP : Indirect TCP for Mobile Hosts.* Proc. Second Usenix Symposium on Mobile and Location-Independent Computing, April 1995.

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

[4] H. Wada, T. Yozawa, T. Ohnishi and Y. Tanaka. *Mobile computing environment based on internet packet forwarding.* Proc. Usenix Winter Technical Conference, January 1993.

[5] J. Ioannidis and G.Q. Maguire. *The design and implementation of a mobile internetworking architecture.* Proc. USENIX Winter Technical Conference, January 1993.

[6] J. Postel. *Transmission Control Protocol. RFC 792.* SRI Network Information Center, Menlo Park, CA, September 1981.

[7] W. Richard Stevens. *TCP Slow Start, Congestion Avoidance, Fast Re-transmit and Fast Recovery Algorithms.* Internet Draft, March 1996. draft-stevens-tcpca-spec-01.txt

[8] D.B. Johnson. *Scalable Support for Transparent Mobile Host Internetworking.* Proc. 9th Annual IEEE Workshop on Computer Communications, IEEE Communications Society, 1994.

[9] R. Caceres and L. Iftnode.*Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments.* IEEE Journal on Selected Areas in Communications, 13(5) June 1994.

[10] Theodore S. Rappaport. Wireless Communications : Principles and Practice. Prentice Hall Inc. 1996.