Delayed Duplicate-Acknowledgements: A proposal to Improve Performance of TCP on Wireless Links *†

Miten N. Mehta Nitin H. Vaidya

Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
E-mail: {mmehta,vaidya}@cs.tamu.edu
Web: http://www.cs.tamu.edu/faculty/vaidya/mobile.html

February 24, 1998

Abstract

TCP protocol has been designed and tuned to perform well on wired network where the packet loss is mainly due to congestion. On heterogeneous networks where some links can be wireless, this assumption is no more valid. In this paper, we propose a mechanism which takes advantage of link level retransmissions on wireless link while avoiding interference between link level and TCP retransmissions. The idea is to delay the third and subsequent duplicate acknowledgements for some time, during which the packet may reach receiver wireless host due to local link level retransmissions. In case of packet losses due to corruptions on wireless links, delayed dupack scheme may avoid

*This report is revised version of the report released on end2end-interest email list on December 27, 1997.

†Research reported is supported in part by Texas Advanced Technology Program grants 009741-052-C and 010115-248-C.

unnecessary invocation of congestion control mechanisms at the sender. We expect that this scheme may improve performance of TCP over wireless links.

1 Introduction

In this paper, we consider a TCP connection between a two hosts such that the last link on the path from the sender to the receiver is a wireless link. Such a path is illustrated in Figure 1. This scenario is quite typical when communicating with mobile hosts. In Figure 1, FH (fixed host) is the node sending data on the TCP connection, and MH (mobile host) is the intended recipient of the data. BS (base station) provides access to the mobile host MH via a wireless link.

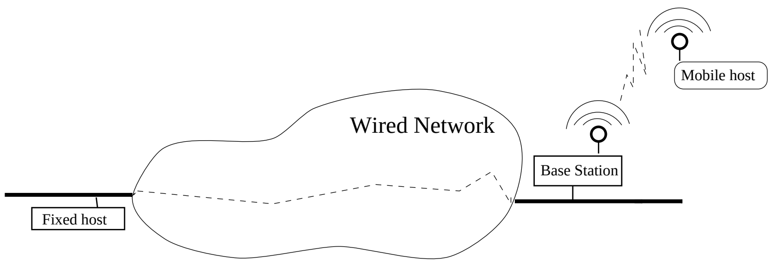

Figure 1: Topology

The wireless links are usually more prone to errors, as compared to wires. A natural approach to improve reliability of a wireless link is to use *link-level retransmission*. In a typical link-level retransmission scheme, MH link layer sends an acknowledgement to the BS link layer for each packet received from the BS. If the BS does not receive an expected ack within a timeout interval, it retransmits the packet. This process is continued until the expected ack is received, or number of retransmissions exceeds a threshold.

Link-level retransmissions provide higher link-level reliability; which benefits all types of traffic, TCP and non-TCP. Unfortunately, link-level retransmissions interfere with retransmissions by the TCP sender [3]. Therefore, researchers have developed several schemes to allow so-called "local recovery" (essentially consisting of retransmissions from the base station) that do not interfere with TCP's retransmission mechanism.

In this paper, we present a new and much simpler scheme to improve performance of TCP using wireless link-level retransmissions. First, in Section 2, we identify the causes for the interference between TCP and link-level retransmissions. Section 3 discusses existing approaches to improve performance of TCP over wireless, and identify their shortcomings. Section 4 discusses the basic idea behind the proposed approach. Several refinements on the proposed approach are feasible, as elaborated in Section 6. Section 7 discusses the implementation details of a wireless testbed. Performance evaluation of the proposed scheme and conclusions will be presented in Section 8 and Section 9 respectively.

2 TCP and Link-level Retransmissions

TCP achieves end-to-end reliability semantics by requiring the receiver to acknowledge receipt of each data packet from the sender. If the sender, for some reason, concludes that a packet is lost before reaching the receiver, the sender retransmits the lost packet. In the modern TCP protocol, there are two mechanisms used to initiate a retransmission from the sender:

- Time-out: If acknowledgement for a certain packet is not received within a *timeout* period (calculated dynamically), then the sender assumes that the packet is lost, and retransmits the packet.
- Fast retransmit: If the sender receives at least 3[1] duplicate acknowledgements (dupacks) for a certain packet, the TCP sender assumes that the packet is lost, and retransmits the packet. This is known as the fast retransmit mechanism [6].

When the packet is lost on wireless link, retransmissions from FH (which may be far away from MH) wastes the bandwidth on wired links. Also, after getting indication of a packet loss, FH may invoke congestion control algorithms, which in turn can reduce the window size and degrade the performance. So in case of packet loss on the wireless link, local retransmission from base station is more advisable. While the retransmission(s) from

[1] 3 is default value of tcprexmtthresh variable.

BS is in progress, the sender may either (i) time-out, or (ii) receive 3 dupacks from the mobile host. In either case, the sender will also retransmit the lost packet, leading to the so-called "interference" phenomenon. Such interference can result in poor throughput, as shown in [3].

3 Related Work

There has been interesting research on improving TCP performance in the presence of transmission errors [1, 2, 3, 4, 5, 7]. Indirect TCP protocols [1, 7] break one TCP connection into two – one for the last wireless hop, and one for the rest of the path. This approach violates the end-to-end semantics of TCP. The Snoop protocol [3, 4] performs even better than indirect TCP, and does not violate the end-to-end semantics. Snoop requires change in intermediate node (*base station*) code to perform retransmission of packets lost due to wireless transmission errors. The Explicit Bad State Notification (EBSN) scheme in [2] requires the *base station* to send a special message to the sender when bursty error conditions are detected on the wireless channel. Similar approaches are suggested in [5].

The existing schemes *require* cooperation from intermediate hosts (typically, the base station) on the path from the sender to the receiver. Also, in three circumstances listed below, the past research is not applicable. Note that these circumstances may occur independently or together.

- IP traffic is encrypted: In this case, intermediate nodes cannot differentiate the TCP packets from the other packets. In this case, any scheme that needs to make use of information (such as sequence number) specific to TCP packets will not work.

- Data packets and TCP acknowledgement (ack) packets traverse different paths. This situation can occur, for instance, when a satellite downlink is being used to send data to a receiver (as in DirecPC [8]), whereas acks are sent over land-based channels. In this situation, the forward and reverse paths may, potentially, not have any nodes in common (except sender and receiver). Therefore, any scheme that requires the same intermediate node to handle data as well as ack packets will not work.

- The link-layer performs local retransmission to recover from wireless transmission losses, but the link layer is **not** aware of the TCP protocol. In this case, link layer retransmission scheme interferes with TCP retransmission scheme, leading to redundant retransmissions, and degraded performance.

In this paper, our interest is in schemes that can work well under the above circumstances while maintaining end-to-end reliability semantics.

4 Proposed Approach: Basic Idea

Of the various protocols proposed so far, the snoop protocol [4] and indirect-TCP protocols [1, 7] perform quite well. These protocols have two features:

- Local recovery: As in a link-level retransmission, these protocols retransmit packets lost on the wireless link.
- Reducing interference between TCP retransmissions and local recovery: Unlike a basic link-level retransmission protocol, these schemes minimize the likelihood that the base station and sender FH, **both** will retransmit the packet.

In these protocols [4, 1, 7] *both* of the above features are accomplished primarily at the base station. The proposed approach also attempts to provide the above two features, as follows:

- Local recovery: Base station does link-level retransmission for packets lost on wireless link. Retransmissions are limited to a finite (typically small) number of retries.
- Reducing interference between TCP retransmissions and local recovery: The mobile host has the responsibility to minimize the interference between TCP and link-level acknowledgements. How the mobile host achieves this is the subject of rest of this section.

As noted in Section 2, interference between retransmission occurs either (i) sender times out while link-level retransmissions are in progress, or (ii) sender receives 3 dupacks. Now, in current TCP implementations, sender time-out is very coarse (multiple of 200 millisec or 500 millisec, in most popular implementations). On the other hand, the link-level retransmission is likely to complete much sooner than 200 millisec in most environments (except, perhaps, some satellite environments). Therefore, the likelihood of sender timing out before link-level retransmissions complete is small. Thus, the primary cause of interference is that the sender may retransmit a packet after receiving 3 dupacks.

A simple (and potentially poor) solution is to switch off the *fast retransmit* mechanism at the sender. This approach can degrade TCP performance when congestion losses are more probable than transmission errors.

This paper proposes an alternative mechanism that requires modifications only at the mobile host MH (sender and base station can continue using normal TCP/IP stack). In our approach, the third[2] dupack is delayed before it is sent by the mobile host. More specifically, the mobile host does not send a third (and subsequent) dupack(s) for a packet, unless the interval since the time when the first dupack[3] is sent exceeds a certain threshold, say, δ.

Now there are two possibilities:

- A packet is lost on the wireless link: If δ is chosen large enough to allow time for a local retransmission (from base station) of a lost packet, then the third dupack will not be sent at all. Thus, the sender will not do fast retransmit (which reduces its window size, in addition to unnecessary retransmission).
- A packet is lost due to congestion before it could reach the base station: In this case, the link-layer retransmission mechanism is not useful. However, the mobile host will delay the third dupack for δ time units. This can lead to a degradation in performance, compared to normal TCP. Normal TCP will send the third dupack without any delay, thus initiating fast retransmission sooner than the proposed protocol.

[2]If the third dupack is delayed, then, by definition, fourth and subsequent dupacks are also delayed.
[3]Alternatively, the threshold may be set on the interval since the packet before the lost packet was received.

From the above discussion, it should be clear that the proposed approach works well for packet losses due to wireless link errors, but performs worse than normal TCP for congestion losses.

The above protocol raises several questions:

- How does the value of δ affect performance?

  Observe that $\delta = 0$ reduces our scheme to normal TCP, whereas $\delta = \infty$ is equivalent to turning fast retransmit off at the sender. Thus, $\delta = 0$ is appropriate if all losses are due to congestion, whereas $\delta = \infty$ is appropriate if all losses are due to errors. In general, a small δ should be suitable when congestion losses are more probable, and large δ should be more suitable when error losses are more probable.

- What is the appropriate threshold δ?

  In general, δ will be a function of several parameters such as error rate, congestion level, and wireless propagation delay.

- How does this approach perform compared to other approaches?

  Note that, as the proposed protocol does not modify the base station, it is unfair to compare it with snoop or split-TCP. A fair comparison is with two versions of normal TCP: (a) with link-level retransmission allowed, and (b) with link-level retransmission suppressed.

5 When is delayed-dupack beneficial ?

As we discussed earlier in this section, our approach tries to achieve better performance by doing local recovery and by reducing interference between TCP retransmissions and local recovery. We also noted that likelihood of sender timing out before link-level retransmissions complete, is small. The reason for interference is mainly due to sender receiving 3 dupacks. Dupacks are generated because of packets received between lost packet and retransmitted packet.

One can categorize the placement of retransmitted packets in 3 cases as shown in Figure 2.

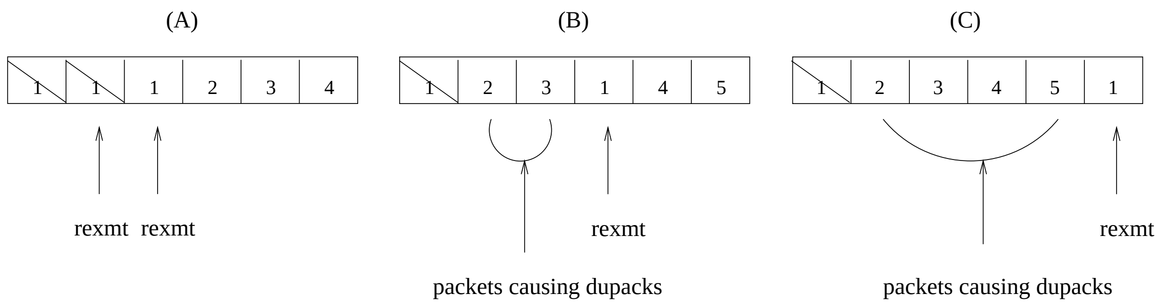

Figure 2: Placement of retransmitted packets

case A: The packets are sent across the channel as First In First Out(FIFO). Hence, till the packet is successfully delivered, BS keeps on retransmitting it. BS transmits packet x+1 only after packet x is successfully received by MH.

case B: BS does not wait for ack before transmitting subsequent packets. Hence before ack for packet x comes, BS transmits packet x+1 and so on. But here the *timeout* value for retransmission and *delay-bandwidth product* are such that, BS gets indication of packet loss before it can transmit packet x+3 or succeeding ones. Hence only 2 or less dupacks are generated before retransmitted packet may reach at MH. Let us see an example to see what is going on. If the bandwidth of the channel is 1.6 Mbps and if the packet size is 1460 bytes, each packet will take approximately 7 ms to reach from BS to MH. If the *timeout* value is 20 ms, then BS will have send at the most 2 packets after sending packet x and before timing out and retransmitting packet x. So MH will generate at the most 2 dupacks.

case C: As in case B, BS does not wait for ack, but here the *timeout* value and *delay-bandwidth product* are such that BS may send more than 3 packets between the time when *it sends packet x* and when *it gets indication of loss of packet x*. Hence more than 3 dupacks may be generated by MH before the retransmitted packet can reach to MH.

In cases A and B, number of packets transmitted between packet x and retransmission of packet x are less than 3. So TCP sender will not trigger fast retransmit and congestion control, if local recovery takes care of wireless losses. Hence in these two cases local link level retransmissions are sufficient, and our scheme is not needed. Also note that other schemes like Snoop [4] are also not needed.

It is the case C, where only local link level retransmissions are not sufficient as the interference (between local recovery and TCP retransmissions) and unnecessarily invocation of congestion control at TCP sender, may degrade the performance. In this case, one needs mechanisms such as dupack hiding [4], delayed dupack to achieve better performance.

6 Refinements and Variations over Basic Scheme

6.1 Determining Relative Error Rate

As noted previously, optimal value of δ is a function of the relationship between congestion losses and error losses. There are three reasons why a packet may appear to the receiver as being lost: (a) congestion loss, (b) wireless error, and (c) out of order delivery on the wired path.

Note that, in cases (b) and (c) a "lost or out of order" packet should arrive at the receiver soon after it was found missing by the receiver. Therefore, we consider both as "error" losses in calculation of relative error rate.

The scheme suggested here is quite simple. The scheme periodically chooses a new value of δ. The idea is to keep track of the fraction, say e, of lost packets that are received (though out of order) within δ (current value of δ) time units from the time when they should have arrived. This fraction e is then considered the fraction of packet losses due to wireless errors. The estimated e is then used to choose an appropriate new value for δ.

Clearly, due to the dependence of estimated e on old δ, and dependence of new δ on e, there is a circular relationship between e and δ. For stability, it may be necessary to impose upper and/or lower bound(s) on chosen value of δ. We have yet to experiment with this scheme.

An alternative would be to collect the error rate information at the link-level. However, some link-level implementations may not be designed to provide such information to upper layers, necessitating some other mechanism, such as the one described above.

6.2 Determining the Retransmission Delay

The appropriate value of δ needs to be a function of the delay incurred for retransmission in the event of a transmission error.

One possibility is to assume a constant value for this. A constant value may be acceptable for environments where wireless link latency is constant.

If the wireless link is being shared by multiple hosts, this latency could be time-varying (depending on the contention). For such environments, it will be useful to have some mechanism to get an estimate of current retransmission delay on wireless link. There are at least two possibilities for this:

- The retransmission delay can be estimated as a function of the round-trip delay on the wireless link. The mobile host can get an estimate of the round-trip, for instance, by using "ping" like operations (or sending packet to an "echo" server on the base station).
- If a packet is retransmitted from BS, the delay will be less compared to the packet being retransmitted by FH. MH can take advantage of this fact to guess whether packet was retransmitted from BS or FH. If the guess is that the packet is retransmitted from BS, then the delay measured provides a point estimate of the current retransmission delay.

The other factor which may affect the latency can be multiple hosts sharing same wireless channel for transmissions. But in the environments where each host has got dedicated share of bandwidth, latency will not be dependent on number hosts.

6.3 Explicit Loss Notification to the Receiver (ELNR messages)

Note that if a packet loss is due to congestion, then the mobile host should not delay a dupack. On the other hand, if packet loss is due to wireless error, it is useful to delay the third dupack to allow a local retransmission to succeed. Thus, if the base station can provide the mobile host information that allows the mobile host to distinguish between congestion loss and transmission error loss, then the mobile host can use this information to determine if a dupack should be delayed or not.

We propose one possible approach for this, named as *Explicit Loss Notification to the Receiver* (ELNR for short). The base station sends a ELNR message to the mobile host, whenever the base station receives an out of order packet. When a mobile host receives ELNR, it knows that the perceived packet loss is not due to transmission errors, therefore, it does not delay any dupack.

ELNR message is similar to the Explicit Loss Notification (ELN) previously proposed [3]. The crucial difference is that, in the previous proposal, ELN was sent to the sender, whereas we propose that ELNR be sent to the receiver. Thus, ELNR can be implemented without any modifications to the sender, whereas ELN scheme requires modification to the sender as well.

6.4 What the BS can do?

Base station can help our scheme with different levels of modifications on it.

- BS does nothing special. In this case BS code is not changed in any way and MH is the only place where the changes occur. MH can be changed as explained in sections 6.1 and 6.2. This is the approach that we are most interested in.

  However, there are potentially other approaches as listed below.

- BS maintains "Expected Sequence Number" variable per session and hence if it gets packet with sequence number other than that, it can tell MH about it by sending ELNR. On getting ELNR, MH switches off the delayed dupack scheme till the point when it received EDDAN (Explicit Delayed Dupack Activation Notification) from BS. The problem with this variant is, how does BS find out whether the congestion has gone or not. Because BS does not store any history, it is difficult for it to find out whether now it is getting packets in proper order or not. This can be solved by next variant.

- BS maintains history of packets received in case of some out of order packet arrives at BS from FH. In normal case of in-order packets, information (sequence number and size of packet) will be maintained for last received packet only. But whenever out of order (OOO) packets come, start maintaining cache from next_recv packet to all the packets BS gets, till the hole is filled. So at the point when BS needs to store more than one entry, it informs MH with ELNR, so that MH can turn off the delayed dupack scheme. And when the hole(s) is filled, BS removes the history and informs MH with EDDAN so that MH can turn on the delayed dupack scheme.

6.5 Variants of local retransmissions

Local retransmissions at the link layer may occur due to two reasons.

- Timeouts: Link layer at BS can time out for link layer ack and can retransmit the packet for which the ack was due and did not come.
- When BS link layer receives dupacks: If BS link layer uses a fast retransmission type of mechanism, it retransmits the packet on receipt of dupacks from the other end on wireless link.

6.6 Determining if Link Layer Does Retransmissions

This is an interesting problem to consider. The proposed scheme is not useful if the link layer does not perform retransmissions. Therefore, some mechanism(for the mobile host) to detect whether the link layer is capable of performing retransmissions is useful. We do not consider this problem at present, and assume that the available link layer is known to be able to perform retransmissions.

6.7 LAN vs. WAN

If FH and MH are on the same LAN, MH may select not to activate delayed dupack scheme, because total rtt may not be much larger than wireless link rtt.

7 Implementation

7.1 Topology

As shown in Figure 1 MH is connected with BS over wireless LAN with 2Mbps and FH is connected to BS on wired links over Internet. Both MH and BS are Pentium based workstations running Free BSD 2.1.5 Release 0. These workstations have 915 Mhz AT& T Wavelan cards for wireless connectivity. Whereas FH is running HP-UX B.10.20.

7.2 Local retransmissions

In actual implementations the wireless cards will/should do local retransmissions as explained in section 6.5. But our implementations do not do that, so we had to simulate local retransmissions at somewhat higher level. We are doing that between IP and TCP layer on BS. (Note that in real implementation this change in BS is not needed.)

7.3 Measurement Mechanism

Three types of application layer programs are written to make measurement process easy and automatic.

- Configuration Daemon: This daemon runs on BS and MH and it gets and sets kernel variable values by using getsockopt and setsockopt functions. It binds to a well known port where our monitor program can send requests of getting/setting values of kernel variables.
- Sender Daemon: This daemon runs on FH and it gets commands from monitor program and can send data to MH. Data size, destination, destination port, how many times data is to be sent, etc. are supplied by monitor to sender daemon as part of *send command* . This daemon makes easier to send data from remote machine on WAN to MH on any place without logging on to sender machine interactively.

- Monitor program: This program is the key part of automatic measurement taking. It reads command file and parses the commands. The command can be *send command* (sent to sender daemon), *configuration command* (sent to configuration daemon) or *parameters command* which can be needed by monitor program for deciding other parameters like sender machine, destination machine, various port addresses etc. One can run monitor program by calling them from scripts which can be run at definite times, when you are even logged on the machine and it reads the commands and instructs other daemons appropriately and hence takes measurements and logs them in files automatically.

8 Performance Evaluation

To be added in a future revision of this paper.

9 Conclusions

This paper proposes an approach for improving performance of TCP using wireless link-level retransmissions and delayed dupacks. It requires no modifications at Base station or Sender, maintains end-to-end reliability, and is simple to implement.

10 Acknowledgements

We would like to thank Luigi Rizzo, Mark Allman and Hari Balakrishnan for their valuable comments and suggestions on the previous version of this report.

References

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

[2] B. Bakshi, P. Krishna, N. Vaidya, and D. Pradhan, "Improving performance of TCP over wireless networks," in *17th Int. Conf. Distributed Computing Systems ICDCS*, May 1997.

[3] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparision of mechanisms for improving TCP performance over wireless links," in *ACM SIGCOMM, Stanford, CA*, Aug. 1996.

[4] H. Balakrishnan, S. Seshan, and R. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks*, vol. 1, Dec. 1995.

[5] R. C. Durst, G. J. Miller, and E. J. Travis, "Tcp extentions for space communications," in *Mobicom*, 1996.

[6] W. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recover algorithmsRFC," Jan. 1997.

[7] R. Yavatkar and N. Bhagwat, "Improving end-to-end performance of TCP over mobile internetworks," in *Workshop on Mobile Computing Systems and Applications*, Dec. 1994.

[8] Y. Zhang and S. Dao, "Integrating direct broadcast satellite with wireless local access," in *Int. Workshop on Satellite-based Information Services*, pp. 24–29, Nov. 1996.