Network Working Group
Request for Comments: 2416
Category: Informational

T. Shepard
C. Partridge
BBN Technologies

September 1998

When TCP Starts Up With Four Packets Into Only Three Buffers

Status of this Memo

Copyright Notice

Abstract

This memo is to document a simple experiment. The experiment showed that in the case of a TCP receiver behind a 9600 bps modem link at the edge of a fast Internet where there are only 3 buffers before the modem (and the fourth packet of a four-packet start will surely be dropped), no significant degradation in performance is experienced by a TCP sending with a four-packet start when compared with a normal slow start (which starts with just one packet).

Background

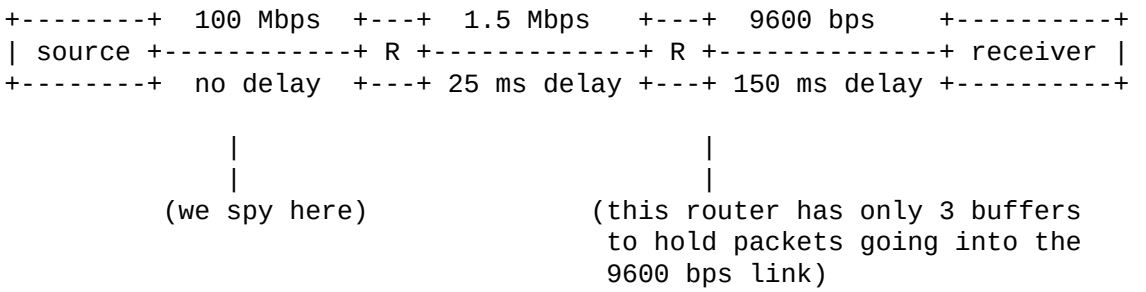
Sally Floyd has proposed that TCPs start their initial slow start by sending as many as four packets (instead of the usual one packet) as a means of getting TCP up-to-speed faster. (Slow starts instigated due to timeouts would still start with just one packet.) Starting with more than one packet might reduce the start-up latency over long-fat pipes by two round-trip times. This proposal is documented further in [1], [2], and in [3] and we assume the reader is familiar with the details of this proposal.

On the end2end-interest mailing list, concern was raised that in the (allegedly common) case where a slow modem is served by a router which only allocates three buffers per modem (one buffer being transmitted while two packets are waiting), that starting with four packets would not be good because the fourth packet is sure to be dropped.

Vern Paxson replied with the comment (among other things) that the four-packet start is no worse than what happens after two round trip times in normal slow start, hence no new problem is introduced by starting with as many as four packets. If there is a problem with a four-packet start, then the problem already exists in a normal slow-start startup after two round trip times when the slow-start algorithm will release into the net four closely spaced packets.

The experiment reported here confirmed Vern Paxson's reasoning.

Scenario and experimental setup



The scenario studied and simulated consists of three links between the source and sink. The first link is a 100 Mbps link with no delay. It connects the sender to a router. (It was included to have a means of logging the returning ACKs at the time they would be seen by the sender.) The second link is a 1.5 Mbps link with a 25 ms one-way delay. (This link was included to roughly model traversing an un-congested, intra-continental piece of the terrestrial Internet.) The third link is a 9600 bps link with a 150 ms one-way delay. It connects the edge of the net to a receiver which is behind the 9600 bps link.

The queue limits for the queues at each end of the first two links were set to 100 (a value sufficiently large that this limit was never a factor). The queue limits at each end of the 9600 bps link were set to 3 packets (which can hold at most two packets while one is being sent).

Version 1.2a2 of the the NS simulator (available from LBL) was used to simulate both one-packet and four-packet starts for each of the available TCP algorithms (tahoe, reno, sack, fack) and the conclusion reported here is independent of which TCP algorithm is used (in general, we believe). In this memo, the "tahoe" module will be used to illustrate what happens. In the 4-packet start cases, the "window-init" variable was set to 4, and the TCP implementations were modified to use the value of the window-init variable only on

connection start, but to set cwnd to 1 on other instances of a slow-start. (The tcp.cc module as shipped with ns-1.2a2 would use the window-init value in all cases.)

The packets in simulation are 1024 bytes long for purposes of determining the time it takes to transmit them through the links. (The TCP modules included with the LBL NS simulator do not simulate the TCP sequence number mechanisms. They use just packet numbers.)

Observations are made of all packets and acknowledgements crossing the 100 Mbps no-delay link, near the sender. (All descriptions below are from this point of view.)

What happens with normal slow start

At time 0.0 packet number 1 is sent.

At time 1.222 an ack is received covering packet number 1, and packets 2 and 3 are sent.

At time 2.444 an ack is received covering packet number 2, and packets 4 and 5 are sent.

At time 3.278 an ack is received covering packet number 3, and packets 6 and 7 are sent.

At time 4.111 an ack is received covering packet number 4, and packets 8 and 9 are sent.

At time 4.944 an ack is received covering packet number 5, and packets 10 and 11 are sent.

At time 5.778 an ack is received covering packet number 6, and packets 12 and 13 are sent.

At time 6.111 a duplicate ack is recieved (covering packet number 6).

At time 7.444 another duplicate ack is received (covering packet number 6).

At time 8.278 a third duplicate ack is received (covering packet number 6) and packet number 7 is retransmitted.

(And the trace continues...)

What happens with a four-packet start

At time 0.0, packets 1, 2, 3, and 4 are sent.

At time 1.222 an ack is received covering packet number 1, and packets 5 and 6 are sent.

At time 2.055 an ack is received covering packet number 2, and packets 7 and 8 are sent.

At time 2.889 an ack is received covering packet number 3, and packets 9 and 10 are sent.

At time 3.722 a duplicate ack is received (covering packet number 3).

At time 4.555 another duplicate ack is received (covering packet number 3).

At time 5.389 a third duplicate ack is received (covering packet number 3) and packet number 4 is retransmitted.

(And the trace continues...)

Discussion

At the point left off in the two traces above, the two different systems are in almost identical states. The two traces from that point on are almost the same, modulo a shift in time of (8.278 - 5.389) = 2.889 seconds and a shift of three packets. If the normal TCP (with the one-packet start) will deliver packet N at time T, then the TCP with the four-packet start will deliver packet N - 3 at time T - 2.889 (seconds).

Note that the time to send three 1024-byte TCP segments through a 9600 bps modem is 2.66 seconds. So at what time does the four-packet-start TCP deliver packet N? At time T - 2.889 + 2.66 = T - 0.229 in most cases, and in some cases earlier, in some cases later, because different packets (by number) experience loss in the two traces.

Thus the four-packet-start TCP is in some sense 0.229 seconds (or about one fifth of a packet) ahead of where the one-packet-start TCP would be. (This is due to the extra time the modem sits idle while waiting for the dally timer to go off in the receiver in the case of the one-packet-start TCP.)

The states of the two systems are not exactly identical. They differ slightly in the round-trip-time estimators because the behavior at the start is not identical. (The observed round trip times may differ by a small amount due to dally timers and due to that the one-packet start experiences more round trip times before the first loss.) In the cases where a retransmit timer did later go off, the additional

difference in timing was much smaller than the 0.229 second difference discribed above.

Conclusion

In this particular case, the four-packet start is not harmful.

Non-conclusions, opinions, and future work

A four-packet start would be very helpful in situations where a long-delay link is involved (as it would reduce transfer times for moderately-sized transfers by as much as two round-trip times). But it remains (in the authors' opinions at this time) an open question whether or not the four-packet start would be safe for the network.

It would be nice to see if this result could be duplicated with real TCPs, real modems, and real three-buffer limits.

Security Considerations

This document discusses a simulation study of the effects of a proposed change to TCP. Consequently, there are no security considerations directly related to the document. There are also no known security considerations associated with the proposed change.

References

1. S. Floyd, Increasing TCP's Initial Window (January 29, 1997). URL ftp://ftp.ee.lbl.gov/papers/draft.jan29.

2. S. Floyd and M. Allman, Increasing TCP's Initial Window (July, 1997). URL http://gigahertz.lerc.nasa.gov/~mallman/share/draft-ss.txt

3. Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", RFC 2414, September 1998.

Authors' Addresses

Tim Shepard
BBN Technologies
10 Moulton Street
Cambridge, MA 02138

EMail: shep@alum.mit.edu

Craig Partridge
BBN Technologies
10 Moulton Street
Cambridge, MA 02138

EMail: craig@bbn.com

Full Copyright Statement