End-To-End Research Group
Request for Comments: DRAFT
Category: Informational

C. Partridge
BBN Technologies

Sep 1998

# ACK Spacing for High Delay-Bandwidth Paths with Insufficient Buffering

## Status of this Memo



An argument is made that the correct way to solve buffering shortages in routers on high delay-bandwidth paths is for routers to space out the TCP acks.

This memo presents thoughts from a discussion held at the July 1997 meeting of the End-To-End (E2E) Research Group. The material presented is a half-baked suggestion and should not be interpreted as an official recommendation of the Research Group. Comments are solicited and should be addressed to the author.

## 1. Introduction

Suppose you want TCP implementations to be able to fill a 155 Mb/s path. Further suppose that the path includes a satellite in a geosynchronous orbit, so the round trip delay through the path is at least 500 ms, and the delay-bandwidth product is 9.7 megabytes or more.

If we further assume the TCP implementations support TCP Large Windows and PAWS (many do), so they can manage 9.7 MB TCP window, then we can be sure the TCP will eventually start sending at full path rate (unless the satellite channel is very lossy). But it may

take a long time to get the TCP up to full speed.

One (of several) possible causes of the delay is a shortage of buffering in routers. To understand this particular problem, consider the following idealized behavior of TCP during slow start. During slow start, for every segment ACKed, the sender transmits two new segments. In effect, this behavior means the sender is transmitting at *twice* the data rate of the segments being ACKed. Keep in mind the separation between ACKs represents (in an ideal world) the rate segments can flow through the bottleneck router in the path. So the sender is bursting data at twice the bottleneck rate, and a queue must be forming during the burst. In the simplest case, the queue is entirely at the bottleneck router, and at the end of the burst, the queue is storing half the data in the burst. (Why





half? During the burst, the sender transmitted at twice the bottleneck rate. Suppose it takes one time unit to send a segment on the bottlenecked link. During the burst the bottleneck will receive two segments in every time unit, but only be able to transmit one segment. The result is a net of one new segment queued every time unit, for the life of the burst.)

TCP will end the slow start phase in response to the first lost datagram. Assuming good quality transmission links, the first lost datagram will be lost because the bottleneck queue overflowed. We would like that loss to occur in the round-trip after the slow start congestion window has reached the delay-bandwidth product. Now consider the buffering required in the bottleneck link during the next to last round trip. The sender will send an entire delay-bandwidth worth of data in one-half a round-trip time (because it sends at twice the channel rate). So for half the round-trip time, the bottleneck router is in the mode of forwarding one segment while receiving two. (For the second half of the round-trip, the router is draining its queue). That means, to avoid losing any segments, the router must have buffering equal to half the delay-bandwidth product, or nearly 5 MB.

Most routers do not have anywhere near 5 MB of buffering for a single link. Or, to express this problem another way, because routers do not have this much buffering, the slow start stage will end prematurely, when router buffering is exhausted. The consequence of ending slow start prematurely is severe. At the end of slow start,

TCP goes into congestion avoidance, in which the window size is increased much more slowly. So even though the channel is free, because we did not have enough router buffering, we will transmit slowly for a period of time (until the more conservative congestion avoidance algorithm sends enough data to fill the channel).

## 2. What to Do?

So how to get around the shortage of router buffering?

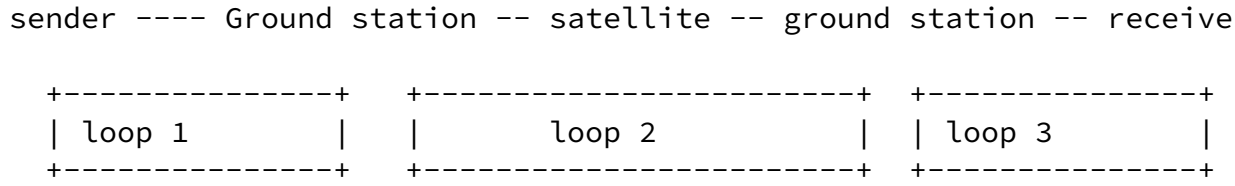
One solution has been proposed, cascading TCPs. We would like to suggest another solution, ACK spacing. Both schemes involve layer violations because they require the router to examine the TCP header.

### 2.1 Cascading TCPs

One approach is to use cascading TCPs, in which we build a custom TCP for the satellite (or bottleneck) link and insert it between the sender's and receiver's TCPs, as shown below:








This approach can work but is awkward. Among its limitations are: the buffering problem remains (at points of bandwidth mismatches, queues will form); the scheme violates end-to-end semantics of TCP (the sender will get ACKs for data that has not and may never reach the receiver); it constrains the reverse path of the TCP connection to pass through points at which the multiple TCP connections are spliced together (a problem if satellite links are unidirectional); and it doesn't work with end-to-end encryption (i.e. if data above the IP layer is encrypted).

### 2.2 ACK Spacing

Another approach is to find some way to spread the bursts, either by

having the sender spread out the segments, or having the network arrange for the ACKs to arrive at the sender with a two segment spacing (or larger).

Changing the sender is feasible, although it requires very good operating system timers. But it has the disadvantage that only upgraded senders get the performance improvement.

Finding a way for the network to space the ACKs would allow TCP senders to transmit at the right rate, without modification. Furthermore, it can be done by a router. The router simply has to snoop the returning TCP ACKs and spread them out. (Note that if the transmissions are encrypted, in many scenarios the router can still figure out which segments are likely TCP ACKs and spread them out).

There are some difficult issues with this approach. The most notable ones are:

1. What algorithm to use to determine the proper ACK spacing.

2. Related to (1), it may be necessary to known when a TCP is in slow-start vs. congestion-avoidance, as the desired spacing between ACKs is likely to be different in the two phases.

3. What to do about assymetric routes (if anything). The scheme works so long as the router sees the ACKs (it does not have to see the related data). However, if the ACKs do not return through the ACK-spacing router, it is not possible to do ACK spacing.





4. How much, if at all, does ack compression between the respacing point and the sender undo the effects of ack spacing?

5. How much per-flow (soft) state is required in the ACK spacing router?

Despite these challenges the approach has appeal. Changing software in a few routers (particularly those at likely bottleneck links) on high delay-bandwidth paths could give a performance boost to lots of TCP connections.

## Security Issues

ACK spacing introduces no new security issues. ACK spacing does not change the contents of any datagram. It simply delays some datagrams in transit, just as a queue might. TCP and other higher layer protocols are already required to work correctly with queueing delays, and indeed, work correctly when encountering far more serious transmission errors such as damage, loss, duplication and reordering [2].

## Credit and Disclaimer

The particular idea of ACK spacing was developed by during the meeting by Mark Handley and Van Jacobson in response to an issue raised by the author, and was inspired, in part by ideas to enhance wireless routers to improve TCP performance [1].

## Intellectual Property Issues

The author has learned from the IETF that parties may be attempting to patent schemes similar to this one. Readers are advised to check with the IETF to learn of any intellectual property rights issues.

## References

1. H. Balakrishnan, V.N. Padmanabhan, S. Seshan and R.H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", Proc. ACM SIGCOMM '96, pp. 256-269.
2. J. Postel, ed. Transmission Control Protocol RFC-793, Internet Requests for Comments, No. 793, September 1981, p. 4.