Network Working Group
Request for Comments: 3390
Obsoletes: 2414
Updates: 2581
Category: Standards Track

M. Allman
BBN/NASA GRC
S. Floyd
ICIR
C. Partridge
BBN Technologies

October 2002

Increasing TCP's Initial Window

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document specifies an optional standard for TCP to increase the permitted initial window from one or two segment(s) to roughly 4K bytes, replacing RFC 2414. It discusses the advantages and disadvantages of the higher initial window, and includes discussion of experiments and simulations showing that the higher initial window does not lead to congestion collapse. Finally, this document provides guidance on implementation issues.

Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1. TCP Modification

This document obsoletes [RFC2414] and updates [RFC2581] and specifies an increase in the permitted upper bound for TCP's initial window from one or two segment(s) to between two and four segments. In most cases, this change results in an upper bound on the initial window of roughly 4K bytes (although given a large segment size, the permitted initial window of two segments may be significantly larger than 4K bytes).

The upper bound for the initial window is given more precisely in (1):

```
      min (4*MSS, max (2*MSS, 4380 bytes))                   (1)
```

Note: Sending a 1500 byte packet indicates a maximum segment size (MSS) of 1460 bytes (assuming no IP or TCP options). Therefore, limiting the initial window's MSS to 4380 bytes allows the sender to transmit three segments initially in the common case when using 1500 byte packets.

Equivalently, the upper bound for the initial window size is based on the MSS, as follows:

```
    If (MSS <= 1095 bytes)
        then win <= 4 * MSS;
    If (1095 bytes < MSS < 2190 bytes)
        then win <= 4380;
    If (2190 bytes <= MSS)
        then win <= 2 * MSS;
```

This increased initial window is optional: a TCP MAY start with a larger initial window. However, we expect that most general-purpose TCP implementations would choose to use the larger initial congestion window given in equation (1) above.

This upper bound for the initial window size represents a change from RFC 2581 [RFC2581], which specified that the congestion window be initialized to one or two segments.

This change applies to the initial window of the connection in the first round trip time (RTT) of data transmission following the TCP three-way handshake. Neither the SYN/ACK nor its acknowledgment (ACK) in the three-way handshake should increase the initial window size above that outlined in equation (1). If the SYN or SYN/ACK is lost, the initial window used by a sender after a correctly transmitted SYN MUST be one segment consisting of MSS bytes.

TCP implementations use slow start in as many as three different ways: (1) to start a new connection (the initial window); (2) to restart transmission after a long idle period (the restart window); and (3) to restart transmission after a retransmit timeout (the loss window). The change specified in this document affects the value of the initial window. Optionally, a TCP MAY set the restart window to the minimum of the value used for the initial window and the current value of cwnd (in other words, using a larger value for the restart window should never increase the size of cwnd). These changes do NOT change the loss window, which must remain 1 segment of MSS bytes (to permit the lowest possible window size in the case of severe congestion).

2. Implementation Issues

When larger initial windows are implemented along with Path MTU Discovery [RFC1191], and the MSS being used is found to be too large, the congestion window 'cwnd' SHOULD be reduced to prevent large bursts of smaller segments. Specifically, 'cwnd' SHOULD be reduced by the ratio of the old segment size to the new segment size.

When larger initial windows are implemented along with Path MTU Discovery [RFC1191], alternatives are to set the "Don't Fragment" (DF) bit in all segments in the initial window, or to set the "Don't Fragment" (DF) bit in one of the segments. It is an open question as to which of these two alternatives is best; we would hope that implementation experiences will shed light on this question. In the first case of setting the DF bit in all segments, if the initial packets are too large, then all of the initial packets will be dropped in the network. In the second case of setting the DF bit in only one segment, if the initial packets are too large, then all but one of the initial packets will be fragmented in the network. When the second case is followed, setting the DF bit in the last segment in the initial window provides the least chance for needless retransmissions when the initial segment size is found to be too large, because it minimizes the chances of duplicate ACKs triggering a Fast Retransmit. However, more attention needs to be paid to the interaction between larger initial windows and Path MTU Discovery.

The larger initial window specified in this document is not intended as encouragement for web browsers to open multiple simultaneous TCP connections, all with large initial windows. When web browsers open simultaneous TCP connections to the same destination, they are working against TCP's congestion control mechanisms [FF99], regardless of the size of the initial window. Combining this behavior with larger initial windows further increases the unfairness to other traffic in the network. We suggest the use of HTTP/1.1 [RFC2068] (persistent TCP connections and pipelining) as a way to achieve better performance of web transfers.

3. Advantages of Larger Initial Windows

1. When the initial window is one segment, a receiver employing delayed ACKs [RFC1122] is forced to wait for a timeout before generating an ACK. With an initial window of at least two segments, the receiver will generate an ACK after the second data segment arrives. This eliminates the wait on the timeout (often up to 200 msec, and possibly up to 500 msec [RFC1122]).

2. For connections transmitting only a small amount of data, a larger initial window reduces the transmission time (assuming at most moderate segment drop rates). For many email (SMTP [Pos82]) and web page (HTTP [RFC1945, RFC2068]) transfers that are less than 4K bytes, the larger initial window would reduce the data transfer time to a single RTT.

3. For connections that will be able to use large congestion windows, this modification eliminates up to three RTTs and a delayed ACK timeout during the initial slow-start phase. This will be of particular benefit for high-bandwidth large-propagation-delay TCP connections, such as those over satellite links.

4. Disadvantages of Larger Initial Windows for the Individual Connection

In high-congestion environments, particularly for routers that have a bias against bursty traffic (as in the typical Drop Tail router queues), a TCP connection can sometimes be better off starting with an initial window of one segment. There are scenarios where a TCP connection slow-starting from an initial window of one segment might not have segments dropped, while a TCP connection starting with an initial window of four segments might experience unnecessary retransmits due to the inability of the router to handle small bursts. This could result in an unnecessary retransmit timeout. For a large-window connection that is able to recover without a retransmit timeout, this could result in an unnecessarily-early transition from the slow-start to the congestion-avoidance phase of the window increase algorithm. These premature segment drops are unlikely to occur in uncongested networks with sufficient buffering or in moderately-congested networks where the congested router uses active queue management (such as Random Early Detection [FJ93, RFC2309]).

Some TCP connections will receive better performance with the larger initial window even if the burstiness of the initial window results in premature segment drops. This will be true if (1) the TCP connection recovers from the segment drop without a retransmit timeout, and (2) the TCP connection is ultimately limited to a small congestion window by either network congestion or by the receiver's advertised window.

5. Disadvantages of Larger Initial Windows for the Network

In terms of the potential for congestion collapse, we consider two separate potential dangers for the network. The first danger would be a scenario where a large number of segments on congested links were duplicate segments that had already been received at the receiver. The second danger would be a scenario where a large number of segments on congested links were segments that would be dropped later in the network before reaching their final destination.

In terms of the negative effect on other traffic in the network, a potential disadvantage of larger initial windows would be that they increase the general packet drop rate in the network. We discuss these three issues below.

Duplicate segments:

As described in the previous section, the larger initial window could occasionally result in a segment dropped from the initial window, when that segment might not have been dropped if the sender had slow-started from an initial window of one segment. However, Appendix A shows that even in this case, the larger initial window would not result in the transmission of a large number of duplicate segments.

Segments dropped later in the network:

How much would the larger initial window for TCP increase the number of segments on congested links that would be dropped before reaching their final destination? This is a problem that can only occur for connections with multiple congested links, where some segments might use scarce bandwidth on the first congested link along the path, only to be dropped later along the path.

First, many of the TCP connections will have only one congested link along the path. Segments dropped from these connections do not "waste" scarce bandwidth, and do not contribute to congestion collapse.

However, some network paths will have multiple congested links, and segments dropped from the initial window could use scarce bandwidth along the earlier congested links before ultimately being dropped on subsequent congested links. To the extent that the drop rate is independent of the initial window used by TCP segments, the problem of congested links carrying segments that will be dropped before reaching their destination will be similar for TCP connections that start by sending four segments or one segment.

An increased packet drop rate:

For a network with a high segment drop rate, increasing the TCP initial window could increase the segment drop rate even further. This is in part because routers with Drop Tail queue management have difficulties with bursty traffic in times of congestion. However, given uncorrelated arrivals for TCP connections, the larger TCP initial window should not significantly increase the segment drop rate. Simulation-based explorations of these issues are discussed in Section 7.2.

These potential dangers for the network are explored in simulations and experiments described in the section below. Our judgment is that while there are dangers of congestion collapse in the current Internet (see [FF99] for a discussion of the dangers of congestion collapse from an increased deployment of UDP connections without end-to-end congestion control), there is no such danger to the network from increasing the TCP initial window to 4K bytes.

6. Interactions with the Retransmission Timer

Using a larger initial burst of data can exacerbate existing problems with spurious retransmit timeouts on low-bandwidth paths, assuming the standard algorithm for determining the TCP retransmission timeout (RTO) [RFC2988]. The problem is that across low-bandwidth network paths on which the transmission time of a packet is a large portion of the round-trip time, the small packets used to establish a TCP connection do not seed the RTO estimator appropriately. When the first window of data packets is transmitted, the sender's retransmit timer could expire before the acknowledgments for those packets are received. As each acknowledgment arrives, the retransmit timer is generally reset. Thus, the retransmit timer will not expire as long as an acknowledgment arrives at least once a second, given the one-second minimum on the RTO recommended in RFC 2988.

For instance, consider a 9.6 Kbps link. The initial RTT measurement will be on the order of 67 msec, if we simply consider the transmission time of 2 packets (the SYN and SYN-ACK), each consisting of 40 bytes. Using the RTO estimator given in [RFC2988], this yields an initial RTO of 201 msec (67 + 4*(67/2)). However, we round the RTO to 1 second as specified in RFC 2988. Then assume we send an initial window of one or more 1500-byte packets (1460 data bytes plus overhead). Each packet will take on the order of 1.25 seconds to transmit. Therefore, the RTO will fire before the ACK for the first packet returns, causing a spurious timeout. In this case, a larger initial window of three or four packets exacerbates the problems caused by this spurious timeout.

One way to deal with this problem is to make the RTO algorithm more conservative. During the initial window of data, for instance, the RTO could be updated for each acknowledgment received. In addition, if the retransmit timer expires for some packet lost in the first window of data, we could leave the exponential-backoff of the retransmit timer engaged until at least one valid RTT measurement, that involves a data packet, is received.

Another method would be to refrain from taking an RTT sample during connection establishment, leaving the default RTO in place until TCP takes a sample from a data segment and the corresponding ACK. While this method likely helps prevent spurious retransmits, it also may slow the data transfer down if loss occurs before the RTO is seeded. The use of limited transmit [RFC3042] to aid a TCP connection in recovering from loss using fast retransmit rather than the RTO timer mitigates the performance degradation caused by using the high default RTO during the initial window of data transmission.

This specification leaves the decision about what to do (if anything) with regards to the RTO, when using a larger initial window, to the implementer. However, the RECOMMENDED approach is to refrain from sampling the RTT during the three-way handshake, keeping the default RTO in place until an RTT sample involving a data packet is taken. In addition, it is RECOMMENDED that TCPs use limited transmit [RFC3042].

7. Typical Levels of Burstiness for TCP Traffic.

Larger TCP initial windows would not dramatically increase the burstiness of TCP traffic in the Internet today, because such traffic is already fairly bursty. Bursts of two and three segments are already typical of TCP [Flo97]; a delayed ACK (covering two previously unacknowledged segments) received during congestion avoidance causes the congestion window to slide and two segments to be sent. The same delayed ACK received during slow start causes the window to slide by two segments and then be incremented by one segment, resulting in a three-segment burst. While not necessarily typical, bursts of four and five segments for TCP are not rare. Assuming delayed ACKs, a single dropped ACK causes the subsequent ACK to cover four previously unacknowledged segments. During congestion avoidance this leads to a four-segment burst, and during slow start a five-segment burst is generated.

There are also changes in progress that reduce the performance problems posed by moderate traffic bursts. One such change is the deployment of higher-speed links in some parts of the network, where a burst of 4K bytes can represent a small quantity of data. A second change, for routers with sufficient buffering, is the deployment of queue management mechanisms such as RED, which is designed to be tolerant of transient traffic bursts.

8. Simulations and Experimental Results

8.1 Studies of TCP Connections using that Larger Initial Window

This section surveys simulations and experiments that explore the effect of larger initial windows on TCP connections. The first set of experiments explores performance over satellite links. Larger initial windows have been shown to improve the performance of TCP connections over satellite channels [All97b]. In this study, an initial window of four segments (512 byte MSS) resulted in throughput improvements of up to 30% (depending upon transfer size). [KAGT98] shows that the use of larger initial windows results in a decrease in transfer time in HTTP tests over the ACTS satellite system. A study involving simulations of a large number of HTTP transactions over hybrid fiber coax (HFC) indicates that the use of larger initial windows decreases the time required to load WWW pages [Nic98].

A second set of experiments explored TCP performance over dialup modem links. In experiments over a 28.8 bps dialup channel [All97a, AHO98], a four-segment initial window decreased the transfer time of a 16KB file by roughly 10%, with no accompanying increase in the drop rate. A simulation study [RFC2416] investigated the effects of using a larger initial window on a host connected by a slow modem link and a router with a 3 packet buffer. The study concluded that for the scenario investigated, the use of larger initial windows was not harmful to TCP performance.

Finally, [All00] illustrates that the percentage of connections at a particular web server that experience loss in the initial window of data transmission increases with the size of the initial congestion window. However, the increase is in line with what would be expected from sending a larger burst into the network.

8.2 Studies of Networks using Larger Initial Windows

This section surveys simulations and experiments investigating the impact of the larger window on other TCP connections sharing the path. Experiments in [All97a, AHO98] show that for 16 KB transfers to 100 Internet hosts, four-segment initial windows resulted in a small increase in the drop rate of 0.04 segments/transfer. While the drop rate increased slightly, the transfer time was reduced by roughly 25% for transfers using the four-segment (512 byte MSS) initial window when compared to an initial window of one segment.

A simulation study in [RFC2415] explores the impact of a larger initial window on competing network traffic. In this investigation, HTTP and FTP flows share a single congested gateway (where the number of HTTP and FTP flows varies from one simulation set to another). For each simulation set, the paper examines aggregate link utilization and packet drop rates, median web page delay, and network power for the FTP transfers. The larger initial window generally resulted in increased throughput, slightly-increased packet drop rates, and an increase in overall network power. With the exception of one scenario, the larger initial window resulted in an increase in the drop rate of less than 1% above the loss rate experienced when using a one-segment initial window; in this scenario, the drop rate increased from 3.5% with one-segment initial windows, to 4.5% with four-segment initial windows. The overall conclusions were that increasing the TCP initial window to three packets (or 4380 bytes) helps to improve perceived performance.

Morris [Mor97] investigated larger initial windows in a highly congested network with transfers of 20K in size. The loss rate in networks where all TCP connections use an initial window of four segments is shown to be 1-2% greater than in a network where all connections use an initial window of one segment. This relationship held in scenarios where the loss rates with one-segment initial windows ranged from 1% to 11%. In addition, in networks where connections used an initial window of four segments, TCP connections spent more time waiting for the retransmit timer (RTO) to expire to resend a segment than was spent using an initial window of one segment. The time spent waiting for the RTO timer to expire represents idle time when no useful work was being accomplished for that connection. These results show that in a very congested environment, where each connection's share of the bottleneck bandwidth is close to one segment, using a larger initial window can cause a perceptible increase in both loss rates and retransmit timeouts.

9. Security Considerations

This document discusses the initial congestion window permitted for TCP connections. Changing this value does not raise any known new security issues with TCP.

10. Conclusion

This document specifies a small change to TCP that will likely be beneficial to short-lived TCP connections and those over links with long RTTs (saving several RTTs during the initial slow-start phase).

11. Acknowledgments

We would like to acknowledge Vern Paxson, Tim Shepard, members of the End-to-End-Interest Mailing List, and members of the IETF TCP Implementation Working Group for continuing discussions of these issues and for feedback on this document.

12. References

[AHO98] Mark Allman, Chris Hayes, and Shawn Ostermann, An Evaluation of TCP with Larger Initial Windows, March 1998. ACM Computer Communication Review, 28(3), July 1998. URL "http://roland.lerc.nasa.gov/~mallman/papers/initwin.ps".

[All97a] Mark Allman. An Evaluation of TCP with Larger Initial Windows. 40th IETF Meeting -- TCP Implementations WG. December, 1997. Washington, DC.

[All97b] Mark Allman. Improving TCP Performance Over Satellite Channels. Master's thesis, Ohio University, June 1997.

[All00] Mark Allman. A Web Server's View of the Transport Layer. ACM Computer Communication Review, 30(5), October 2000.

[FF96] Fall, K., and Floyd, S., Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. Computer Communication Review, 26(3), July 1996.

[FF99] Sally Floyd, Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking, August 1999. URL "http://www.icir.org/floyd/end2end-paper.html".

[FJ93] Floyd, S., and Jacobson, V., Random Early Detection gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.

[Flo94] Floyd, S., TCP and Explicit Congestion Notification. Computer Communication Review, 24(5):10-23, October 1994.

[Flo96] Floyd, S., Issues of TCP with SACK. Technical report, January 1996. Available from http://www-nrg.ee.lbl.gov/floyd/.

[Flo97] Floyd, S., Increasing TCP's Initial Window. Viewgraphs, 40th IETF Meeting - TCP Implementations WG. December, 1997. URL "ftp://ftp.ee.lbl.gov/talks/sf-tcp-ietf97.ps".

[KAGT98] Hans Kruse, Mark Allman, Jim Griner, Diepchi Tran. HTTP Page Transfer Rates Over Geo-Stationary Satellite Links. March 1998. Proceedings of the Sixth International Conference on Telecommunication Systems. URL "http://roland.lerc.nasa.gov/~mallman/papers/nash98.ps".

[Mor97] Robert Morris. Private communication, 1997. Cited for acknowledgement purposes only.

[Nic98] Kathleen Nichols. Improving Network Simulation With Feedback, Proceedings of LCN 98, October 1998. URL "http://www.computer.org/proceedings/lcn/8810/8810toc.htm".

[Pos82] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1191] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[RFC1945] Berners-Lee, T., Fielding, R. and H. Nielsen, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[RFC2068] Fielding, R., Mogul, J., Gettys, J., Frystyk, H. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, January 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J. and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[RFC2414] Allman, M., Floyd, S. and C. Partridge, "Increasing TCP's Initial Window", RFC 2414, September 1998.

[RFC2415] Poduri, K. and K. Nichols, "Simulation Studies of Increased Initial TCP Window Size", RFC 2415, September 1998.

[RFC2416] Shepard, T. and C. Partridge, "When TCP Starts Up With Four Packets Into Only Three Buffers", RFC 2416, September 1998.

[RFC2581] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2821] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[RFC3042] Allman, M., Balakrishnan, H. and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit", RFC 3042, January 2001.

[RFC3168] Ramakrishnan, K.K., Floyd, S. and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

Appendix A - Duplicate Segments

In the current environment (without Explicit Congestion Notification [Flo94] [RFC2481]), all TCPs use segment drops as indications from the network about the limits of available bandwidth. We argue here that the change to a larger initial window should not result in the sender retransmitting a large number of duplicate segments that have already arrived at the receiver.

If one segment is dropped from the initial window, there are three different ways for TCP to recover: (1) Slow-starting from a window of one segment, as is done after a retransmit timeout, or after Fast Retransmit in Tahoe TCP; (2) Fast Recovery without selective acknowledgments (SACK), as is done after three duplicate ACKs in Reno TCP; and (3) Fast Recovery with SACK, for TCP where both the sender and the receiver support the SACK option [MMFR96]. In all three cases, if a single segment is dropped from the initial window, no duplicate segments (i.e., segments that have already been received at the receiver) are transmitted. Note that for a TCP sending four 512-byte segments in the initial window, a single segment drop will not require a retransmit timeout, but can be recovered by using the Fast Retransmit algorithm (unless the retransmit timer expires prematurely). In addition, a single segment dropped from an initial window of three segments might be repaired using the fast retransmit algorithm, depending on which segment is dropped and whether or not delayed ACKs are used. For example, dropping the first segment of a three segment initial window will always require waiting for a timeout, in the absence of Limited Transmit [RFC3042]. However, dropping the third segment will always allow recovery via the fast retransmit algorithm, as long as no ACKs are lost.

Next we consider scenarios where the initial window contains two to four segments, and at least two of those segments are dropped. If all segments in the initial window are dropped, then clearly no duplicate segments are retransmitted, as the receiver has not yet received any segments. (It is still a possibility that these dropped segments used scarce bandwidth on the way to their drop point; this issue was discussed in Section 5.)

When two segments are dropped from an initial window of three segments, the sender will only send a duplicate segment if the first two of the three segments were dropped, and the sender does not receive a packet with the SACK option acknowledging the third segment.

When two segments are dropped from an initial window of four segments, an examination of the six possible scenarios (which we don't go through here) shows that, depending on the position of the dropped packets, in the absence of SACK the sender might send one duplicate segment. There are no scenarios in which the sender sends two duplicate segments.

When three segments are dropped from an initial window of four segments, then, in the absence of SACK, it is possible that one duplicate segment will be sent, depending on the position of the dropped segments.

The summary is that in the absence of SACK, there are some scenarios with multiple segment drops from the initial window where one duplicate segment will be transmitted. There are no scenarios in which more than one duplicate segment will be transmitted. Our conclusion is than the number of duplicate segments transmitted as a result of a larger initial window should be small.

Author's Addresses

Mark Allman
BBN Technologies/NASA Glenn Research Center
21000 Brookpark Rd
MS 54-5
Cleveland, OH 44135
EMail: mallman@bbn.com
http://roland.lerc.nasa.gov/~mallman/

Sally Floyd
ICSI Center for Internet Research
1947 Center St, Suite 600
Berkeley, CA 94704
Phone: +1 (510) 666-2989
EMail: floyd@icir.org
http://www.icir.org/floyd/

Craig Partridge
BBN Technologies
10 Moulton St
Cambridge, MA 02138
EMail: craig@bbn.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.