Early Retransmit for TCP and SCTP

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of [RFC2026].

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This document proposes a new mechanism for TCP and SCTP that can be used to more effectively recover lost segments when a connection’s congestion window is small. The "Early Retransmit" mechanism allows the transport to reduce, in certain special circumstances, the number of duplicate acknowledgments required to trigger a fast retransmission. This allows the transport to use fast retransmit to recover packet losses that would otherwise require a lengthy retransmission timeout.

1 Introduction

A number of researchers have pointed out that the loss recovery strategies employed by TCP [RFC793] and SCTP [RFC2960] do not work well when the congestion window at a TCP sender is small. This can happen in a number of situations, such as:

(1) The connection is "application limited" and has only a limited amount of data to send. This can happen any time the...
application does not produce enough data to fill the congestion window. A particular case when all connections become application limited is as the connection ends.

(2) The connection is limited by the receiver-advertised window.

(3) The connection is constrained by end-to-end congestion control when the connection’s share of the path is small, the path has a small bandwidth-delay product or the transport is ascertaining the available bandwidth in the first few round-trip times of slow start.

Many researchers have studied problems with TCP when the congestion window is small and have outlined possible mechanisms to mitigate these problems (e.g., [Mor97,BPS+98,Ba198,LRK98,RFC3150,AA02]). SCTP’s loss recovery and congestion control mechanisms are based on TCP and therefore the same problems impact the performance of SCTP connections. When the transport detects a missing segment, the connection enters a loss recovery phase using one of two methods. First, if an acknowledgment (ACK) for a given segment is not received in a certain amount of time a retransmission timer fires and the segment is resent [RFC2988]. Second, the ‘‘Fast Retransmit’’ algorithm resends a segment when three duplicate ACKs arrive at the sender [Jac88,RFC2581]. However, because duplicate ACKs from the receiver are also triggered by packet reordering in the Internet, the sender waits for three duplicate ACKs in an attempt to disambiguate segment loss from packet reordering. When using small windows it may not be possible to generate the required number of duplicate ACKs to trigger Fast Retransmit when a loss does happen.

Once in a loss recovery phase, a number of techniques can be used to retransmit lost segments. TCP can use slow start based recovery or Fast Recovery [RFC2581], NewReno [RFC2582], and loss recovery based on selective acknowledgments (SACKs) [RFC2018,FF96,RFC3517]. SCTP’s loss recovery is not as varied due to the built-in selective acknowledgments.

The transport’s retransmission timeout (RTO) is based on measured round-trip times (RTT) between the sender and receiver, as specified in [RFC2988] (for TCP) and [RFC2960] (for SCTP). To prevent spurious retransmissions of segments that are only delayed and not lost, the minimum RTO is conservatively chosen to be 1 second. Therefore, it behooves TCP senders to detect and recover from as many losses as possible without incurring a lengthy timeout during which the connection remains idle. However, if not enough duplicate ACKs arrive from the receiver, the Fast Retransmit algorithm is never triggered---this situation occurs when the congestion window is small, if a large number of segments in a window are lost or at the end of a transfer as data drains from the network. For instance, consider a congestion window (cwnd) of three segments. If one segment is dropped by the network, then at most two duplicate ACKs will arrive at the sender, assuming no ACK loss. Since three duplicate ACKs are required to trigger Fast Retransmit, a timeout
will be required to resend the dropped packet.

[BPS+98] shows that roughly 56% of retransmissions sent by a busy web server are sent after the RTO timer expires, while only 44% are handled by Fast Retransmit. In addition, only 4% of the RTO timer-based retransmissions could have been avoided with SACK, which has to continue to disambiguate reordering from genuine loss. Furthermore, [Allo00] shows that for one particular web server the median transfer size is less than four segments, indicating that more than half of the connections will be forced to rely on the RTO timer to recover from any losses that occur. Thus, loss recovery without relying on the conservative RTO is beneficial for short TCP transfers.

The Limited Transmit mechanism introduced in [RFC3042] allows a TCP sender to send previously unsent data upon the reception of each of the two duplicate ACKs that precede a fast retransmit. SCTP [RFC2960] uses SACK information to calculate the number of outstanding segments in the network. Hence, when the first two duplicate ACKs arrive at the sender they will indicate that data has left the network and allow the sender to transmit new data (if available) similar to TCP’s Limited Transmit algorithm.

By sending these two new segments the TCP sender is attempting to induce additional duplicate ACKs (if appropriate) so that Fast Retransmit will be triggered before the retransmission timeout expires. The "Early Retransmit" mechanism outlined in this document covers the case when previously unsent data is not available for transmission.

The next section of this document outlines a small change to TCP and SCTP senders that will decrease the reliance on the retransmission timer, and thereby improve performance when Fast Retransmit cannot otherwise be triggered.

2 Reduction of the Retransmission Threshold

The Early Retransmit algorithm calls for lowering the duplicate ACK threshold when the amount of outstanding data is small and when no unsent data segments are enqueued. In particular, if the following two conditions hold the sender can use Early Retransmit.

(2.a) The amount of outstanding data (ownd) is less than 4*SMSS bytes.

(2.b) There is either no unsent data ready for transmission at the sender or the advertised window does not permit new segments to be transmitted.

When the above two conditions hold the duplicate ACK threshold used to trigger Fast Retransmit MAY be reduced to:

\[
\text{ER\_thresh} = \text{ceiling (ownd/SMSS)} - 1
\]
duplicate ACKs, where ownd is in terms of bytes. In other words, when ownd is small enough that losing one segment would not trigger Fast Retransmit, the duplicate ACK threshold is reduced to the number of duplicate ACKs expected if one segment is lost. This mitigation is less robust in the face of reordered segments than the standard Fast Retransmit threshold of three duplicate ACKs. Research shows that a general reduction in the number of duplicate ACKs required to trigger fast retransmission of a segment to two (rather than three) leads to a reduction in the ratio of good to bad retransmits by a factor of three [Pax97]. However, this analysis did not include the additional conditioning on the event that the ownd was smaller than 4 segments.

We note two “worst case” scenarios for Early Retransmit:

(1) Persistent reordering of segments, coupled with an application that does not constantly send data, can result in large numbers of needless retransmissions when using Early Retransmit. For instance, consider an application that sends data two segments at a time, followed by an idle period when no data is queued for delivery by TCP. If the network consistently reorders the two segments, the sender will needlessly retransmit one out of every two unique segments transmitted (and one-third of all segments) when using the above algorithm. However, this would only be a problem for long-lived connections from applications that transmit in spurts.

(2) Similar to the above, consider the case of 2 segment transfers that always experience reordering. Just as in (1) above, one out of every two unique data segments will be retransmitted needlessly, therefore one-third of the traffic will be spurious.

Currently this document offers no suggestion on how to mitigate the above problems. Rather, the authors believe that the community’s consensus is that Early Retransmit is scoped enough that the worst case problems are pathological and do not need mitigation at this time. However, Appendix A offers a survey of possible mitigations.

3 Related Work

Deployment of Explicit Congestion Notification (ECN) [Flo94, RFC2481] may benefit connections with small congestion window sizes [RFC2884]. ECN provides a method for indicating congestion to the end-host without dropping segments. While some segment drops may still occur, ECN may allow TCP to perform better with small cwnd sizes because the sender will be required to detect less segment loss [RFC2884].

[Ba198] outlines another solution to the problem of having no new segments to transmit into the network when the first two duplicate ACKs arrive. In response to these duplicate ACKs, a TCP sender transmits zero-byte segments to induce additional duplicate ACKs. This method preserves the robustness of the standard Fast Retransmit algorithm at the cost of injecting segments into the network that do
4 Security Considerations

The security considerations found in [RFC2581] apply to this document. No additional security problems have been identified with Early Retransmit at this time.

Acknowledgments

We thank Sally Floyd for her feedback in discussions about Early Retransmit. We also thank Sally Floyd and Hari Balakrishnan who helped with a large portion of the text of this document when it was part of a separate document. Armando Caro and many members of the tsvwg mailing list provided good discussions that helped shape this document.

References


Author’s Addresses:

Mark Allman
NASA Glenn Research Center/BBN Technologies
Lewis Field

Expires: December 2003
Appendix A: Research Issues in Adjusting the Duplicate ACK Threshold

Decreasing the number of duplicate ACKs required to trigger Fast Retransmit, as suggested in section 2, has the drawback of making Fast Retransmit less robust in the face of minor network reordering. Two egregious examples of problems caused by reordering are given in section 2. This appendix outlines several schemes that have been suggested to mitigate the problems caused to Early Retransmit by reordering. These methods need further research before they are suggested for general use.

MITIGATION A.1: Allow a connection to use Early Retransmit as long as the algorithm is not injecting a "too much" spurious data into the network. For instance, using the information provided by TCP’s DSACK option [RFC2883] or SCTP’s Duplicate-TSN notification, a sender can determine when segments sent via Early Retransmit are needless. Likewise, using Eifel [RFC3522] the sender can detect spurious Early Retransmits. Once spurious Early Retransmits are detected the sender can either eliminate the use of Early Retransmit or limit the use of the algorithm to ensure that an acceptably small fraction of the connection’s transmissions are not spurious.

Alternatively, if a sender cannot reliably determine if an Early Retransmitted segment is spurious or not the sender could simply
limit Early Retransmits either to some fixed number per connection (e.g., Early Retransmit is allowed only once per connection) or to some small percentage of the total traffic being transmitted.

MITIGATION A.2: Allow a connection to trigger Early Retransmit using the number of duplicate ACKs defined in equation (1), in addition to a "small" timeout [Pax97]. For instance, a sender may have to wait for 2 duplicate ACKs and then T msec before Early Retransmitting a segment. The added time gives reordered acknowledgments time to arrive at the sender and avoid a needless retransmit. Designing a method for choosing an appropriate timeout is part of the research that would need to be involved in this scheme.