Internet Engineering Task Force
INTERNET DRAFT
File: draft-ietf-tcpm-early-rexmt-00.txt

Mark Allman
ICSI
Konstantin Avrachenkov
INRIA
Urtzi Ayesta
LAAS-CNRS
Josh Blanton
Ohio University
Per Hurtig
Karlstad University

August 2008
Expires: February 2009

Early Retransmit for TCP and SCTP

Status of this Memo

Copyright Notice

Abstract

This document proposes a new mechanism for TCP and SCTP that can be used to 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.

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 Introduction

Many researchers have studied problems with TCP [RFC793,RFC2581] when the congestion window is small and have outlined possible mechanisms to mitigate these problems [Mor97,BPS+98,Bal98,LK98,RFC3150,AA02]. SCTP's [RFC4960] 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. There are several variants of the loss recovery phase depending on the TCP implemention. 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.

All the above variants have two methods for invoking loss recovery. 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,RFC4960]. Second, the ``Fast Retransmit'' algorithm resends a segment when three duplicate ACKs arrive at the sender [Jac88,RFC2581]. Duplicate ACKs are triggered by out-of-order arrivals at the receiver. However, because duplicate ACKs from the receiver are triggered by both packet loss and packet reordering in the network path, the sender waits for three duplicate ACKs in an attempt to disambiguate packet loss from packet reordering. When using small congestion windows it may not be possible to generate the required number of duplicate ACKs to trigger Fast Retransmit when a loss does happen.

Small windows can occur in a number of situations, such as:

(1) 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.

(2) 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.

(3) The connection is limited by the receiver's advertised window.

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 [RFC4960] (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, [All00] 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 that does not rely on the conservative RTO is likely to be beneficial for short TCP transfers.

The Limited Transmit mechanism introduced in [RFC3042] allows a TCP sender to transmit previously unsent data upon the reception of each of the two duplicate ACKs that precede a Fast Retransmit. SCTP [RFC4960] 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. In the remainder of this document we use "Limited Transmit" to include both TCP and SCTP mechanisms for sending in response to the first two duplicate ACKs. By sending these two new segments the 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 or cannot be transmitted due to an advertised window limitation.

2 Early Retransmit Algorithm

The Early Retransmit algorithm calls for lowering the threshold for triggering Fast Retransmit when the amount of outstanding data is small and when no previously unsent data can be transmitted (such that Limited Transmit could be used). Duplicate ACKs are triggered by each arriving out-of-order segment. Therefore, Fast Retransmit will not be invoked when there are less than four outstanding segments (assuming only one segment loss in the window). However, TCP and SCTP are not required to track the number of outstanding segments, but rather the number of outstanding bytes or messages. Therefore, applying the intuitive notion of a transport with less than four segments outstanding is more complicated than it first appears. In section 2.1 we describe a "byte-based" variant of Early Retransmit that attempts to roughly map the number of outstanding bytes to a number of outstanding packets that is then used when deciding whether to trigger Early Retransmit. In section 2.2 we describe a "packet-based" variant that represents a more precise algorithm for triggering Early Retransmit. The precision comes at the cost of requiring additional state to be kept by the TCP sender. In both cases we described SACK-based and non-SACK-based versions of the scheme (of course, the non-SACK version will not apply to SCTP).

2.1 Byte-based Early Retransmit

A TCP or SCTP sender MAY use byte-based Early Retransmit.

A sender employing byte-based Early Retransmit MUST use the following two conditions to determine when an Early Retransmit is sent:

(2.a) The amount of outstanding data (ownd)---data sent but not yet acknowledged---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 and the connection does not support SACK the duplicate ACK threshold used to trigger a retransmission MUST be reduced to:

```
               ER_thresh = ceiling (ownd/SMSS) - 1                 (1)
```

duplicate ACKs, where ownd is in terms of bytes.

When conditions (2.a) and (2.b) hold and the connection does support SACK, Early Retransmit MUST be used only when "ownd - SMSS" bytes have been SACKed.

When conditions (2.a) and (2.b) do not hold, the transport MUST NOT use Early Retransmit, but rather prefer the standard mechanisms, including Limited Transmit.

As noted above, the drawback of this byte-based variant is precision [HB08]. We illustrate this with two examples:

+ Consider a non-SACK TCP sender that uses an SMSS of 1460 bytes and transmits three segments each with 400 bytes of payload.

This is a case where Early Retransmit could aid loss recovery if one segment is lost. However, in this case ER_thresh will become zero, per equation (1), because the number of outstanding bytes is a poor estimate of the number of outstanding packets.

A similar problem occurs for senders that employ SACK as the expression "ownd - SMSS" will become negative.

+ Next, consider a non-SACK TCP sender that uses an SMSS of 1460 bytes and transmits 10 segments each with 400 bytes of payload. In this case ER_thresh will be two, per equation (1). Thus, even though there are enough segments outstanding to trigger Fast Retransmit with the standard duplicate ACK threshold Early Retransmit will be triggered. This could cause or exacerbate performance problems caused by packet reordering in the network.

2.2 Packet-based Early Retransmit

A TCP or SCTP sender MAY use packet-based Early Retransmit.

A sender employing packet-based Early Retransmit MUST use the following two conditions to determine when an Early Retransmit is sent:

(3.a) The number of outstanding segments (oseg)---segments sent but not yet acknowledged---is less than four.

(3.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 and the connection does not support SACK the duplicate ACK threshold used to trigger a retransmission MUST be reduced to:

$$ER\_thresh = oseg - 1 \qquad (2)$$

duplicate ACKs, where oseg represents the number of outstanding segments. (We discuss tracking the number of outstanding segments below.)

When conditions (3.a) and (3.b) hold and the connection does support SACK, Early Retransmit MUST be used only when "oseg - 1" segments have been SACKed.

When conditions (3.a) and (3.b) do not hold, the transport MUST NOT use Early Retransmit, but rather prefer the standard mechanisms, including Limited Transmit.

This version of Early Retransmit solves the precision issues discussed in the previous section. As noted previously, the cost is that the implementation will have to track packet boundaries to form an understanding as to how many actual segments have been transmitted, but not acknowledged. This can be done by tracking the boundaries of the three segments on the right side of the current window (which involves tracking four sequence numbers in TCP). This could be done by keeping a circular list of the packet boundaries, for instance. Cumulative ACKs that do not fall within this region indicate that at least four segments are outstanding and therefore

Early Retransmit MUST NOT be used. When the outstanding window becomes small enough that Early Retransmit can be invoked, a full understanding of the number of outstanding packets will be available from the four sequence numbers retained.

3 Discussion

The SACK variant of the Early Retransmit algorithm is preferred to the non-SACK variant due to its robustness in the face of ACK loss (since SACKs are sent redundantly) and due to interactions with the delayed ACK timer. Consider a flight of three segments, S1...S3, with S2 being dropped by the network. When S1 arrives it is in-order and so the receiver may or may not delay the ACK, leading to two scenarios:

(A) The ACK for S1 is delayed: In this case the arrival of S3 will trigger an ACK to be transmitted covering segment S1 (which was previously unacknowledged). In this case Early Retransmit without SACK will not prevent an RTO because no duplicate ACKs will arrive. However, with SACK the ACK for S1 will also include SACK information indicating that S3 has arrived at the receiver. The sender can then invoke Early Retransmit on this ACK because only one packet remains outstanding.

(B) The ACK for S1 is not delayed: In this case the arrival of S1 triggers an ACK of previously unacknowledged data. The arrival of S3 triggers a duplicate ACK (because it is out-of-order). Both ACKs will cover the same segment (S1). Therefore, regardless of whether SACK is used Early Retransmit can be performed by the sender (assuming no ACK loss).

Early Retransmit is less robust in the face of reordered segments than when using the standard Fast Retransmit threshold. Research shows that a general reduction in the number of duplicate ACKs required to trigger Fast Retransmit 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 and that no new data was available for transmission.

A number of studies have shown that network reordering is not a rare event across some network paths. Various measurement studies have shown that reordering along most paths is negligible, but along certain paths can be quite prevalent [Pax97,BPS99,BS02,Pir05]. Evaluating Early Retransmit in the face of real packet reordering is part of the experiment we hope to instigate with this document.

Next, 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. If the network consistently reorders the two segments, the sender will needlessly retransmit one out of every two unique segments transmitted when using the above algorithm (meaning that one-third of all segments sent are needless retransmissions). 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. However, the worst cases are likely pathological and part of the experiments that this document hopes to trigger would involve better understanding of whether such theoretical worst case scenarios are prevalent in the network and in general to explore the tradeoff between spurious fast retransmits and the delay imposed by the RTO. Appendix A does offer a survey of possible mitigations that call for curtailing the use of Early Retransmit when it is making poor retransmission decisions.

4 Related Work

Deployment of Explicit Congestion Notification (ECN) [Flo94,RFC3168] 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 a transport to perform better with small cwnd sizes because the sender will be required to detect less segment loss [RFC2884].

[Bal98] 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 not deliver any data (and, therefore are potentially wasting network resources).

5 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. The notion of Early Transmit was originally sketched in an Internet-Draft co-authored by Sally Floyd and Hari Balakrishnan. Armando Caro and many members of the TSVWG and TCPM working groups

provided good discussions that helped shape this document. Our thanks to all!

Normative References

[RFC793] Jon Postel. Transmission Control Protocol. Std 7, RFC 793. September 1981.

[RFC2018] Matt Mathis, Jamshid Mahdavi, Sally Floyd, Allyn Romanow. TCP Selective Acknowledgement Options. RFC 2018, October 1996.

[RFC2581] Mark Allman, Vern Paxson, W. Richard Stevens. TCP Congestion Control. RFC 2581, April 1999.

[RFC2883] Sally Floyd, Jamshid Mahdavi, Matt Mathis, Matt Podolsky. An Extension to the Selective Acknowledgement (SACK) Option for TCP. RFC 2883, July 2000.

[RFC2988] Vern Paxson, Mark Allman. Computing TCP's Retransmission Timer. RFC 2988, April 2000.

[RFC3042] Mark Allman, Hari Balakrishnan, Sally Floyd. Enhancing TCP's Loss Recovery Using Limited Transmit. RFC 3042, January 2001.

[RFC3522] Reiner Ludwig, Michael Meyer. The Eifel Detection Algorithm for TCP. RFC 3522, April 2003.

[RFC4960] R. Stewart. Stream Control Transmission Protocol. RFC 4960, September 2007.

Informative References

[AA02] Urtzi Ayesta, Konstantin Avrachenkov, "The Effect of the Initial Window Size and Limited Transmit Algorithm on the Transient Behavior of TCP Transfers", In Proc. of the 15th ITC Internet Specialist Seminar, Wurzburg, July 2002.

[All00] Mark Allman. A Web Server's View of the Transport Layer. ACM Computer Communications Review, October 2000.

[Bal98] Hari Balakrishnan. Challenges to Reliable Data Transport over Heterogeneous Wireless Networks. Ph.D. Thesis, University of California at Berkeley, August 1998.

[BPS+98] Hari Balakrishnan, Venkata Padmanabhan, Srinivasan Seshan, Mark Stemm, and Randy Katz. TCP Behavior of a Busy Web Server: Analysis and Improvements. Proc. IEEE INFOCOM Conf., San Francisco, CA, March 1998.

[BS02] John Bellardo, Stefan Savage. Measuring Packet Reordering, ACM/USENIX Internet Measurement Workshop, November 2002.

[FF96] Kevin Fall, Sally Floyd. Simulation-based Comparisons of

Tahoe, Reno, and SACK TCP. ACM Computer Communication Review, July 1996.

[Flo94] Sally Floyd. TCP and Explicit Congestion Notification. ACM Computer Communication Review, October 1994.

[HB08] Per Hurtig, Anna Brunstrom. Enhancing SCTP Loss Recovery: An Experimental Evaluation of Early Retransmit. Elsevier Computer Communication, 2008, to appear.

[Jac88] Van Jacobson. Congestion Avoidance and Control. ACM SIGCOMM 1988.

[LK98] Dong Lin, H.T. Kung. TCP Fast Recovery Strategies: Analysis and Improvements. Proceedings of InfoCom, San Francisco, CA, March 1998.

[Mor97] Robert Morris. TCP Behavior with Many Flows. Proceedings of the Fifth IEEE International Conference on Network Protocols. October 1997.

[Pax97] Vern Paxson. End-to-End Internet Packet Dynamics. ACM SIGCOMM, September 1997.

[Pir05] N. M. Piratla, "A Theoretical Foundation, Metrics and Modeling of Packet Reordering and Methodology of Delay Modeling using Inter-packet Gaps," Ph.D. Dissertation, Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, Fall 2005.

[RFC2582] Sally Floyd, Tom Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582, April 1999.

[RFC2884] Jamal Hadi Salim and Uvaiz Ahmed. Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks. RFC 2884, July 2000.

[RFC3150] Spencer Dawkins, Gabriel Montenegro, Markku Kojo, Vincent Magret. End-to-end Performance Implications of Slow Links. RFC 3150, July 2001.

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

[RFC3517] Ethan Blanton, Mark Allman, Kevin Fall, Lili Wang. A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP. RFC 3517, April 2003.

Author's Addresses:

Mark Allman
International Computer Science Institute
1947 Center Street, Suite 600

Berkeley, CA 94704-1198
Phone: 440-235-1792
mallman@icir.org
http://www.icir.org/mallman/

Konstantin Avrachenkov
INRIA
2004 route des Lucioles, B.P.93
06902, Sophia Antipolis
France
Phone: 00 33 492 38 7751
k.avrachenkov@sophia.inria.fr
http://www.inria.fr/mistral/personnel/K.Avrachenkov/moi.html

Urtzi Ayesta
LAAS-CNRS
7 Avenue Colonel Roche
31077 Toulouse
France
urtzi@laas.fr
http://www.laas.fr/~urtzi

Josh Blanton
Ohio University
301 Stocker Center
Athens, OH 45701
jblanton@irg.cs.ohiou.edu

Per Hurtig
Karlstad University
Department of Computer Science
Universitetsgatan 2 651 88
Karlstad Sweden
per.hurtig@kau.se

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 3. This appendix outlines several schemes that have been suggested to mitigate the problems caused by Early Retransmit in the face of packet reordering. These methods need further research before they are suggested for general use (and, current consensus is that the cases that make Early Retransmit unnecessarily retransmit a large amount of data are pathological and therefore these mitigations are not generally required).

MITIGATION A.1: Allow a connection to use Early Retransmit as long as the algorithm is not injecting "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. For example, a connection could stop using Early Retransmit after the first spurious retransmit is detected.

MITIGATION A.2: 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.3: Allow a connection to trigger Early Retransmit using the criteria given in section 2, 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 Retransmit is invoked. 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.

Intellectual Property Statement

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE

Copyright Statement

Acknowledgment

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