Network Working Group   M. Allman
Request for Comments: 3042   NASA GRC/BBN
Category: Standards Track   H. Balakrishnan
                                                              MIT
                                                         S. Floyd
                                                            ACIRI
                                                     January 2001

Enhancing TCP's Loss Recovery Using Limited Transmit

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 proposes a new Transmission Control Protocol (TCP)
mechanism that can be used to more effectively recover lost segments
when a connection's congestion window is small, or when a large
number of segments are lost in a single transmission window. The
"Limited Transmit" algorithm calls for sending a new data segment in
response to each of the first two duplicate acknowledgments that
arrive at the sender. Transmitting these segments increases the
probability that TCP can recover from a single lost segment using the
fast retransmit algorithm, rather than using a costly retransmission
timeout. Limited Transmit can be used both in conjunction with, and
in the absence of, the TCP selective acknowledgment (SACK) mechanism.

1   Introduction

A number of researchers have observed that TCP's loss recovery
strategies do not work well when the congestion window at a TCP
sender is small. This can happen, for instance, because there is
only a limited amount of data to send, or because of the limit
imposed by the receiver-advertised window, or because of the
constraints imposed by end-to-end congestion control over a
connection with a small bandwidth-delay product
[Riz96,Mor97,BPS+98,Bal98,LK98]. When a TCP detects a missing
segment, it 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 timeout occurs and the
segment is resent [RFC793,PA00]. 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
TCP sender waits for three duplicate ACKs in an attempt to
disambiguate segment loss from packet reordering. Once in a loss
recovery phase, a number of techniques can be used to retransmit lost
segments, including slow start-based recovery or Fast Recovery
[RFC2581], NewReno [RFC2582], and loss recovery based on selective
acknowledgments (SACKs) [RFC2018,FF96].

TCP's retransmission timeout (RTO) is based on measured round-trip
times (RTT) between the sender and receiver, as specified in [PA00].
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
when 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
or if a large number of segments in a window are lost. 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. Since three duplicate ACKs are required
to trigger Fast Retransmit, a timeout will be required to resend the
dropped packet.

[BPS+97] found that roughly 56% of retransmissions sent by a busy web
server were sent after the RTO expires, while only 44% were handled
by Fast Retransmit. In addition, only 4% of the RTO-based
retransmissions could have been avoided with SACK, which of course
has to continue to disambiguate reordering from genuine loss. In
contrast, using the technique outlined in this document and in
[Bal98], 25% of the RTO-based retransmissions in that dataset would
have likely been avoided.

The next section of this document outlines small changes to TCP
senders that will decrease the reliance on the retransmission timer,
and thereby improve TCP performance when Fast Retransmit is not
triggered. These changes do not adversely affect the performance of
TCP nor interact adversely with other connections, in other
circumstances.

1.1 Terminology

In this document, he key words "MUST", "MUST NOT", "REQUIRED",
"SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY",
AND "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and
indicate requirement levels for protocols.

2   The Limited Transmit Algorithm

When a TCP sender has previously unsent data queued for transmission
it SHOULD use the Limited Transmit algorithm, which calls for a TCP
sender to transmit new data upon the arrival of the first two
consecutive duplicate ACKs when the following conditions are
satisfied:

   * The receiver's advertised window allows the transmission of the
     segment.

   * The amount of outstanding data would remain less than or equal
     to the congestion window plus 2 segments. In other words, the
     sender can only send two segments beyond the congestion window
     (cwnd).

The congestion window (cwnd) MUST NOT be changed when these new
segments are transmitted. Assuming that these new segments and the
corresponding ACKs are not dropped, this procedure allows the sender
to infer loss using the standard Fast Retransmit threshold of three
duplicate ACKs [RFC2581]. This is more robust to reordered packets
than if an old packet were retransmitted on the first or second
duplicate ACK.

Note: If the connection is using selective acknowledgments [RFC2018],
the data sender MUST NOT send new segments in response to duplicate
ACKs that contain no new SACK information, as a misbehaving receiver
can generate such ACKs to trigger inappropriate transmission of data
segments. See [SCWA99] for a discussion of attacks by misbehaving
receivers.

Limited Transmit follows the "conservation of packets" congestion
control principle [Jac88]. Each of the first two duplicate ACKs
indicate that a segment has left the network. Furthermore, the
sender has not yet decided that a segment has been dropped and
therefore has no reason to assume that the current congestion control
state is inaccurate. Therefore, transmitting segments does not
deviate from the spirit of TCP's congestion control principles.

[BPS99] shows that packet reordering is not a rare network event.
[RFC2581] does not provide for sending of data on the first two
duplicate ACKs that arrive at the sender. This causes a burst of
segments to be sent when an ACK for new data does arrive following
packet reordering. Using Limited Transmit, data packets will be
clocked out by incoming ACKs and therefore transmission will not be
as bursty.

Note: Limited Transmit is implemented in the ns simulator [NS].
Researchers wishing to investigate this mechanism further can do so
by enabling "singledup_" for the given TCP connection.

3   Related Work

Deployment of Explicit Congestion Notification (ECN) [Flo94,RFC2481]
may benefit connections with small congestion window sizes [SA00].
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 congestion window
sizes because the sender can avoid many of the Fast Retransmits and
Retransmit Timeouts that would otherwise have been needed to detect
dropped segments [SA00].

When ECN-enabled TCP traffic competes with non-ECN-enabled TCP
traffic, ECN-enabled traffic can receive up to 30% higher goodput.
For bulk transfers, the relative performance benefit of ECN is
greatest when on average each flow has 3-4 outstanding packets during
each round-trip time [ZQ00]. This should be a good estimate for the
performance impact of a flow using Limited Transmit, since both ECN
and Limited Transmit reduce the reliance on the retransmission timer
for signaling congestion.

The Rate-Halving congestion control algorithm [MSML99] uses a form of
limited transmit, as it calls for transmitting a data segment on
every second duplicate ACK that arrives at the sender. The algorithm
decouples the decision of what to send from the decision of when to
send. However, similar to Limited Transmit the algorithm will always
send a new data segment on the second duplicate ACK that arrives at
the sender.

4   Security Considerations

The additional security implications of the changes proposed in this
document, compared to TCP's current vulnerabilities, are minimal.
The potential security issues come from the subversion of end-to-end
congestion control from "false" duplicate ACKs, where a "false"
duplicate ACK is a duplicate ACK that does not actually acknowledge
new data received at the TCP receiver. False duplicate ACKs could

result from duplicate ACKs that are themselves duplicated in the
network, or from misbehaving TCP receivers that send false duplicate
ACKs to subvert end-to-end congestion control [SCWA99,RFC2581].

When the TCP data receiver has agreed to use the SACK option, the TCP
data sender has fairly strong protection against false duplicate
ACKs. In particular, with SACK, a duplicate ACK that acknowledges
new data arriving at the receiver reports the sequence numbers of
that new data. Thus, with SACK, the TCP sender can verify that an
arriving duplicate ACK acknowledges data that the TCP sender has
actually sent, and for which no previous acknowledgment has been
received, before sending new data as a result of that acknowledgment.
For further protection, the TCP sender could keep a record of packet
boundaries for transmitted data packets, and recognize at most one
valid acknowledgment for each packet (e.g., the first acknowledgment
acknowledging the receipt of all of the sequence numbers in that
packet).

One could imagine some limited protection against false duplicate
ACKs for a non-SACK TCP connection, where the TCP sender keeps a
record of the number of packets transmitted, and recognizes at most
one acknowledgment per packet to be used for triggering the sending
of new data. However, this accounting of packets transmitted and
acknowledged would require additional state and extra complexity at
the TCP sender, and does not seem necessary.

The most important protection against false duplicate ACKs comes from
the limited potential of duplicate ACKs in subverting end-to-end
congestion control. There are two separate cases to consider: when
the TCP sender receives less than a threshold number of duplicate
ACKs, and when the TCP sender receives at least a threshold number of
duplicate ACKs. In the latter case a TCP with Limited Transmit will
behave essentially the same as a TCP without Limited Transmit in that
the congestion window will be halved and a loss recovery period will
be initiated.

When a TCP sender receives less than a threshold number of duplicate
ACKs a misbehaving receiver could send two duplicate ACKs after each
regular ACK. One might imagine that the TCP sender would send at
three times its allowed sending rate. However, using Limited
Transmit as outlined in section 2 the sender is only allowed to
exceed the congestion window by less than the duplicate ACK threshold
(of three segments), and thus would not send a new packet for each
duplicate ACK received.

References

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

   [BPS+97]   Hari Balakrishnan, Venkata Padmanabhan, Srinivasan Seshan,
              Mark Stemm, and Randy Katz. TCP Behavior of a Busy Web
              Server: Analysis and Improvements. Technical Report
              UCB/CSD-97-966, August 1997. Available from
              http://nms.lcs.mit.edu/~hari/papers/csd-97-966.ps. (Also
              in Proc. IEEE INFOCOM Conf., San Francisco, CA, March
              1998.)

   [BPS99]   Jon Bennett, Craig Partridge, Nicholas Shectman. Packet
              Reordering is Not Pathological Network Behavior. IEEE/ACM
              Transactions on Networking, December 1999.

   [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.

   [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, March
              1998.

   [MSML99]   Matt Mathis, Jeff Semke, Jamshid Mahdavi, Kevin Lahey. The
              Rate Halving Algorithm, 1999. URL:
              http://www.psc.edu/networking/rate_halving.html.

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

   [NS]   Ns network simulator. URL: http://www.isi.edu/nsnam/.

[PA00]   Paxson, V. and M. Allman, "Computing TCP's Retransmission
            Timer", RFC 2988, November 2000.

[Riz96]   Luigi Rizzo. Issues in the Implementation of Selective
            Acknowledgments for TCP. January, 1996. URL:
            http://www.iet.unipi.it/~luigi/selack.ps

[SA00]   Hadi Salim, J. and U. Ahmed, "Performance Evaluation of
            Explicit Congestion Notification (ECN) in IP Networks", RFC
            2884, July 2000.

[SCWA99]   Stefan Savage, Neal Cardwell, David Wetherall, Tom
            Anderson. TCP Congestion Control with a Misbehaving
            Receiver. ACM Computer Communications Review, October
            1999.

[RFC793]   Postel, J., "Transmission Control Protocol", STD 7, RFC
            793, September 1981.

[RFC2018]   Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP
            Selective Acknowledgement Options", RFC 2018, October 1996.

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

[RFC2481]   Ramakrishnan, K. and S. Floyd, "A Proposal to Add Explicit
            Congestion Notification (ECN) to IP", RFC 2481, January
            1999.

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

[RFC2582]   Floyd, S. and T. Henderson, "The NewReno Modification to
            TCP's Fast Recovery Algorithm", RFC 2582, April 1999.

[ZQ00]   Yin Zhang and Lili Qiu, Understanding the End-to-End
            Performance Impact of RED in a Heterogeneous Environment,
            Cornell CS Technical Report 2000-1802, July 2000. URL
            http://www.cs.cornell.edu/yzhang/papers.htm.

Authors' Addresses

   Mark Allman
   NASA Glenn Research Center/BBN Technologies
   Lewis Field
   21000 Brookpark Rd. MS 54-5
   Cleveland, OH 44135

   Phone: +1-216-433-6586
   Fax: +1-216-433-8705
   EMail: mallman@grc.nasa.gov
   http://roland.grc.nasa.gov/~mallman

   Hari Balakrishnan
   Laboratory for Computer Science
   545 Technology Square
   Massachusetts Institute of Technology
   Cambridge, MA 02139

   EMail: hari@lcs.mit.edu
   http://nms.lcs.mit.edu/~hari/

   Sally Floyd
   AT&T Center for Internet Research at ICSI (ACIRI)
   1947 Center St, Suite 600
   Berkeley, CA 94704

   Phone: +1-510-666-2989
   EMail: floyd@aciri.org
   http://www.aciri.org/floyd/

Full Copyright Statement

Acknowledgement