                    Retransmission Timeout Considerations

Status of this Memo

Copyright Notice

Abstract

    This document provides for high-level guidance for retransmission
    timeout schemes appropriate for general use in the Internet.

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 BCP 14, RFC 2119

[RFC2119].

1   Introduction

    Despite our best intentions and most robust mechanisms, reliability
    in networking ultimately requires a timeout and re-try mechanism.
    Often there are more timely and precise mechanisms for repairing
    loss (e.g., TCP's fast retransmit [RFC5681], NewReno [RFC6582] or
    selective acknowledgment scheme [RFC2018,RFC3517]) which require
    information exchange between components in the system. Such
    communication cannot be guaranteed. To the contrary, we can always
    depend on the passage of time and therefore our ultimate backstop to
    ensuring reliability is a timeout. (Note: There is a case when we
    cannot count on the passage of time, but in this case we believe
    repairing loss will be a moot point and hence we do not further
    consider this case in this document.)

    Various protocols have defined their own timeout mechanisms (e.g.,
    TCP [RFC6298], SCTP [RFC4960]). The specifics of retransmission
    timeouts often represent a particular tradeoff between correctness
    and responsiveness. Therefore, we have found that even though the
    procedures are standardized, implementations also often add their
    own subtle imprint on the specifics of the process to tilt the
    tradeoff between correctness and responsiveness in some way. At
    this point we recognize that often the specifics are not crucial for
    network safety. Hence, in this document we outline the high-level
    principles that are crucial for any retransmission timeout scheme to
    leverage. The intent is to then allow implementations of protocols
    and applications to instantiate mechanisms that best realize their
    specific goals within this framework. These specific mechanisms
    could be standardized or ad-hoc, but as long as they adhere to the
    guidelines given in this document they would be considered
    consistent with the standards.

2   Guidelines

    We now list the four guidelines that apply when utilizing a
    retransmission timeout (RTO).

    (1) In the absence of any knowledge about the round-trip time (RTT)
        of a path the RTO MUST be conservatively set to no less than 1
        second, per TCP's current default RTO [RFC6298].

        This guideline ensures two important aspects of the RTO. First,
        when transmitting into an unknown network, retransmissions will
        not be sent before an ACK would reasonably be expected to arrive
        and hence possibly waste scarce network resources. Second, as
        noted below, sometimes retransmissions can lead to ambiguities
        in assessing the RTT of a network path. Therefore, it is
        especially important for the first RTT sample to be free of
        ambiguities such that there is a baseline for the remainder of
        the communication.

    (2) We specify three guidelines that pertain to the sampling of the

RTT.

(a) In steady state the RTO MUST be set based on recent
    observations of both the RTT and the variance of the RTT.

    In other words, the RTO should be based on a reasonable
    amount of time the sender should wait for an acknowledgment
    of the data before retransmitting the given data.

(b) RTT observations MUST be taken regularly.

    The exact definition of "regularly" is deliberately left
    vague. TCP takes an RTT sample once per RTT, or if using
    the timestamp option [RFC1323] on each acknowledgment
    arrival. [AP99] shows that both these approaches result in
    roughly equivalent performance for the RTO estimator.
    Additionally, [AP99] shows that taking only a single RTT
    sample per TCP connection is also suboptimal. Therefore,
    for the purpose of this guideline we state that RTT samples
    SHOULD be taken at least every RTT or as frequently as data
    is exchanged and ACKed if that happens less frequently than
    every RTT. However, we also recognize that it may not
    always be practical to take an RTT sample this often in all
    cases and hence this requirement is explicitly a "SHOULD"
    and not a "MUST".

(c) RTT samples used in the computation of the RTO MUST NOT be
    ambiguous.

    Assume two copies of some segment X are transmitted at times
    t0 and t1 and then segment X is acknowledged at time t2. It
    is not clear which copy of X triggered the ACK and hence the
    actual RTT is either t2-t1 or t2-t0, but which could be a
    mystery. Therefore, in this situation we use Karn's
    algorithm [KP87,RFC6298] to use neither version of the RTT
    sample and hence not update the RTO.

    There are cases where two copies of some data are
    transmitted in a way whereby the sender can tell which is
    being acknowledged by an incoming ACK. E.g., TCP's
    timestamp option [RFC1323] allows for segments to be
    uniquely identified and hence avoid the ambiguity. In such
    cases there is no ambiguity and the resulting samples can
    update the RTO.

(3) Each time the RTO fires and causes a retransmission the value of
    the RTO MUST be exponentially backed off such that the next
    firing requires a longer interval. The backoff may be removed
    after the successful transmission of non-retransmitted data.

    This ensures network safety.

(4) Retransmission timeouts MUST be taken as indications of
    congestion in the network and the sending rate adapted using a

standard mechanism (e.g., TCP collapses the congestion window to
one segment [RFC5681]).

This ensures network safety.

An exception is made to this rule if a standard mechanism is
used to determine that a particular loss is due to a
non-congestion event (e.g., bit errors). In such a case a
congestion control action is not required.

3   Discussion

We note that research has shown the tension between responsiveness
and correctness of TCP's RTO seems to be a fundamental tradeoff
[AP99]. That is, making TCP's RTO more aggressive (via the EWMA
gains, lowering the minimum RTO, etc.) can reduce the time spent
waiting on needed RTOs. However, at the same time such
aggressiveness leads to more needless RTOs, as well. Therefore,
being as aggressive as the guidelines sketched in the last section
allow in any particular situation may not be the best course of
action (e.g., because an RTO expiration carries a requirement to
slow down).

While the tradeoff between responsiveness and correctness seems
fundamental, the tradeoff can be made less relevant if the sender
can detect and recover from spurious RTOs. Several mechanisms have
been proposed for this purpose, such as Eifel [RFC3522], F-RTO
[RFC5682] and DSACK [RFC2883,RFC3708]. Using such mechanisms may
allow a data originator to tip towards being more responsive without
incurring (as much of) the attendant costs of needless retransmits.

Also, note, that in addition to the experiments discussed in [AP99],
the Linux TCP implementation has been using various non-standard RTO
mechanisms for many years seemingly without large scale problems
(e.g., using different EWMA gains). Also, a number of
implementations use minimum RTOs that are less than the 1 second
specified in [RFC6298]. While the precise implications of this may
show more spurious retransmits (per [AP99]) we are aware of no large
scale problems caused by this change to the minimum RTO.

Finally, we note that while allowing implementations to be more
aggressive may in fact increase the number of needless
retransmissions the above guidelines fail safe in that they insist
on exponential backoff of the RTO and a transmission rate reduction.
Therefore, allowing implementers latitude in their instantiations of
an RTO mechanism does not somehow open the flood gates to aggressive
behavior. Since there is a downside to being aggressive the
incentives for proper behavior are retained in the mechanism.

4   Security Considerations

This document does not alter the security properties of
retransmission timeout mechanisms. See [RFC6298] for a discussion
of these within the context of TCP.

    This document benefits from years of discussions with Sally Floyd,
    Shawn Ostermann, Vern Paxson and the members of the TCPM and
    TCP-IMPL working groups.

Normative References

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

Informative References

    [AP99] Allman, M., V. Paxson, "On Estimating End-to-End Network Path
        Properties", Proceedings of the ACM SIGCOMM Technical Symposium,
        September 1999.

    [KP87] Karn, P. and C. Partridge, "Improving Round-Trip Time
        Estimates in Reliable Transport Protocols", SIGCOMM 87.

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

    [RFC2883] Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, "An
        Extension to the Selective Acknowledgement (SACK) Option for
        TCP", RFC 2883, July 2000.

    [RFC3517] Blanton, E., Allman, M., Fall, K., and L. Wang, "A
        Conservative Selective Acknowledgment (SACK)-based Loss Recovery
        Algorithm for TCP", RFC 3517, April 2003.

    [RFC3522] Ludwig, R., M. Meyer, "The Eifel Detection Algorithm for
        TCP", RFC 3522, april 2003.

    [RFC3708] Blanton, E., M. Allman, "Using TCP Duplicate Selective
        Acknowledgement (DSACKs) and Stream Control Transmission
        Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs)
        to Detect Spurious Retransmissions", RFC 3708, February 2004.

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

    [RFC5682] Sarolahti, P., M. Kojo, K. Yamamoto, M. Hata, "Forward
        RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious
        Retransmission Timeouts with TCP", RFC 5682, September 2009.

    [RFC6298] Paxson, V., M. Allman, H.K. Chu, M. Sargent, "Computing
        TCP's Retransmission Timer", June 2011, RFC 6298.

    [RFC6582] Henderson, T.,, S. Floyd, A. Gurtov, Y. Nishida, "The
        NewReno Modification to TCP's Fast Recovery Algorithm", April
        2012, RFC 6582.

Authors' Addresses

    Mark Allman
    International Computer Science Institute
    1947 Center St. Suite 600
    Berkeley, CA 94704

    Phone: 440-235-1792
    EMail: mallman@icir.org
    http://www.icir.org/mallman