Internet Engineering Task Force   M. Allman
INTERNET-DRAFT   ICSI
File: draft-ietf-tcpm-rto-consider-05.txt   March 10, 2017
Intended Status: Best Current Practice
Expires: September 10, 2017

                  Retransmission Timeout Requirements

Abstract

    Ensuring reliable communication often manifests in a timeout and

retry mechanism. Each implementation of a retransmission timeout
mechanism represents a balance between correctness and timeliness
and therefore no implementation suits all situations. This document

   provides high-level requirements for retransmission timeout schemes
   appropriate for general use in the Internet. Within the
   requirements, implementations have latitude to define particulars
   that best address each situation.

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](BCP 14), [RFC 2119](RFC 2119)
   [[RFC2119](RFC2119)].

1   Introduction

   Reliable transmission is a key property for many network protocols
   and applications. Our protocols use various mechanisms to achieve
   reliable data transmission. Often we use continuous or periodic
   reports from the recipient to inform the sender's notion of which
   pieces of data are missing and need to be retransmitted to ensure
   reliability. Alternatively, information coding---e.g., FEC---can be
   used to achieve probabilistic reliability without retransmissions.
   However, despite our best intentions and most robust mechanisms, the
   only thing we can truly depend on is the passage of time and
   therefore our ultimate backstop to ensuring reliability is a timeout
   and re-try mechanism. That is, the sender sets some expectation for
   how long to wait for confirmation of delivery for a given piece of
   data. When this time period passes without delivery confirmation
   the sender assumes the data was lost in transit and therefore
   schedules a retransmission. This process of ensuring reliability
   via time-based loss detection and resending lost data is commonly
   referred to as a "retransmission timeout (RTO)" mechanism.

   Various protocols have defined their own RTO mechanisms (e.g., TCP
   [[RFC6298](RFC6298)], SCTP [[RFC4960](RFC4960)], SIP [[RFC3261](RFC3261)]). The specifics of
   retransmission timeouts often represent a particular tradeoff
   between correctness and responsiveness [[AP99](AP99)]. In other words we
   want to simultaneously:

     - wait long enough to ensure the detection of loss is correct and
       therefore a retransmission is in fact needed, and

     - bound the delay we impose on applications before repairing
       loss.

   Serving both of these goals is difficult as they pull in opposite
   directions. I.e., towards either (a) withholding needed
   retransmissions too long to ensure the original transmission is
   truly lost or (b) not waiting long enough---to help application
   responsiveness---and hence sending unnecessary (often denoted

"spurious") retransmissions.

We have found that even though the RTO procedure is standardized for some protocols (e.g., TCP [RFC6298]), implementations often add their own subtle imprint on the specifics of the process to tilt the

tradeoff between correctness and responsiveness in some particular way.

At this point we recognize that often these specific tweaks that deviate from standardized RTO mechanisms do not materially impact network safety. Therefore, in this document we outline a set of high-level protocol-agnostic requirements for RTO mechanisms. The intent is to provide a safe foundation on which implementations have the flexibility to instantiate mechanisms that best realize their specific goals.

2   **Scope**

The principles we outline in this document are protocol-agnostic and widely applicable. We make the following scope statements about the application of the requirements discussed in Section 3:

(S.1) The requirements in this document apply only to timer-based loss detection and retransmission.

While there are a bevy of uses for timers in protocols---from rate-based pacing to connection failure detection to making congestion control decisions and beyond---these are outside the scope of this document.

(S.2) The requirements in this document only apply to cases where loss detected via a timer is repaired by a retransmission of the original data.

Other cases are certainly possible---e.g., replacing the lost data with an updated version---but fall outside the scope of this document.

(S.3) The requirements in this document apply only to endpoint-to-endpoint unicast communication. Reliable multicast (e.g., [RFC5740]) protocols are explicitly outside the scope of this document.

Protocols such as SCTP [RFC4960] and MP-TCP [RFC6182] that communicate in a unicast fashion with multiple specific endpoints can leverage the requirements in this document provided they track state and follow the requirements for each endpoint independently. I.e., if host A communicates with hosts B and C, A must use independent RTOs for traffic sent to B and C.

(S.4) There are cases where state is shared across connections or flows (e.g., [RFC2140], [RFC3124]). The RTO is one piece state that is often discussed as sharable. These situations raise issues that the simple flow-oriented RTO mechanism

discussed in this document does not consider (e.g., how long
to preserve state between connections). Therefore, while the
general principles given in Section 3 are likely applicable,
sharing RTOs across flows is outside the scope of this

           document.

     (S.5) The requirements in this document apply to reliable
           transmission, but do not assume that all data transmitted
           within a connection or flow is reliably sent.

           E.g., a protocol like DCCP [RFC4340] could leverage the
           requirements in this document for the initial reliable
           handshake even though the protocol reverts to unreliable
           transmission after the handshake.

           E.g., a protocol like SCTP [RFC4960] could leverage the
           requirements for data that is sent only "partially reliably".
           In this case, the protocol uses two phases for each message.
           In the first phase, the protocol attempts to ensure
           reliability and can leverage the requirements in this
           document. At some point the value of the data is gone and the
           protocol transitions to the second phase where the data is
           treated as unreliably transmitted and therefore the protocol
           will no longer attempt to repair the loss---and hence there
           are no more retransmissions and the requirements in this
           document are moot.

     (S.6) The requirements for RTO mechanisms in this document can be
           applied regardless of whether the RTO mechanism is the sole
           loss repair strategy or works in concert with other
           mechanisms.

           E.g., for a simple protocol like UDP-based DNS [] a timeout
           and re-try mechanism is likely to act alone to ensure
           reliability.

           E.g., within a complex protocol like TCP or SCTP we have
           designed methods to detect and repair loss based on explicit
           endpoint state sharing [RFC2018,RFC4960,RFC6675]. These
           mechanisms are preferred over the RTO as they are often more
           timely and precise than the coarse-grained RTO. In these
           cases, the RTO becomes a last resort when the more advanced
           mechanisms fail.

3   Requirements

     We now list the requirements that apply when designing
     retransmission timeout (RTO) mechanisms.

     (1) In the absence of any knowledge about the latency of a path, the
         RTO MUST be conservatively set to no less than 1 second.

         This requirement 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 latency of a network

path. Therefore, it is especially important for the first
latency sample to be free of ambiguities such that there is a
baseline for the remainder of the communication.

The specific constant (1 second) comes from the analysis of
Internet RTTs found in Appendix A of [RFC6298].

(2) As we note above, loss detection happens when a sender does not
    receive delivery confirmation within an some expected period of
    time. We now specify four requirements that pertain to setting
    the length of this expectation.

    Often measuring the time required for delivery confirmation is
    is framed as involving the "round-trip time (RTT)" of the
    network path as this is the minimum amount of time required to
    receive delivery confirmation and also often follows protocol
    behavior whereby acknowledgments are generated quickly after
    data arrives. For instance, this is the case for the RTO used
    by TCP [RFC6298] and SCTP [RFC4960]. However, this is somewhat
    mis-leading as the expected latency is better framed as the
    "feedback time" (FT). In other words, the expectation is not
    always simply a network property, but includes additional time
    before a sender should reasonably expect a response to a query.

    For instance, consider a UDP-based DNS request from a client to
    a recursive resolver. When the request can be served from the
    resolver's cache the FT likely well approximates the network RTT
    between the client and resolver. However, on a cache miss the
    resolver will request the needed information from one or more
    authoritative DNS servers, which will non-trivially increase the
    FT compared to the RTT between the client and resolver.

    Therefore, we express the following requirements in terms of FT:

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

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

    (b) FT observations SHOULD be taken regularly.

        Internet measurements show that taking only a single FT
        sample per TCP connection results in a relatively poorly
        performing RTO mechanism [AP99], hence this requirement that
        the FT be sampled continuously throughout the lifetime of
        communication.

        The notion of "regularly" SHOULD be defined as at least once

per RTT or as frequently as data is exchanged in cases where
that happens less frequently than once per RTT. However, we
also recognize that it may not always be practical to take
an FT sample this often in all cases. Hence, this

once-per-RTT definition of "regularly" is explicitly a
"SHOULD" and not a "MUST".

TCP takes an FT sample roughly once per RTT, or if using the
timestamp option [RFC7323] on each acknowledgment arrival.
[AP99] shows that both these approaches result in roughly
equivalent performance for the RTO estimator.

(c) FT observations MAY be taken from non-data exchanges.

Some protocols use keepalives, heartbeats or other messages
to exchange control information. To the extent that the
latency of these transactions mirrors data exchange, they
can be leveraged to take FT samples within the RTO
mechanism. Such samples can help protocols keep their RTO
accurate during lulls in data transmission. However, given
that these messages may not be subject to the same delays as
data transmission, we do not take a general view on whether
this is useful or not.

(d) An RTO mechanism MUST NOT use ambiguous FT samples.

Assume two copies of some segment X are transmitted at times
t0 and t1 and then at time t2 the sender receives
confirmation that X in fact arrived. In some cases, it is
not clear which copy of X triggered the confirmation and
hence the actual FT is either t2-t1 or t2-t0, but which is a
mystery. Therefore, in this situation an implementation
MUST use Karn's algorithm [KP87,RFC6298] and use neither
version of the FT 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 [RFC7323] 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 is used to detect a loss and a retransmission
is scheduled, the value of the RTO MUST be exponentially backed
off such that the next firing requires a longer interval. The
backoff SHOULD be removed after the successful repair of the
lost data and subsequent transmission of non-retransmitted data.

A maximum value MAY be placed on the RTO. The maximum RTO MUST
NOT be less than 60 seconds (a la [RFC6298]).

This ensures network safety.

(4) Retransmissions triggered by the RTO mechanism MUST be taken as
    indications of network congestion and the sending rate adapted
    using a standard mechanism (e.g., TCP collapses the congestion
    window to one segment [RFC5681]).

This ensures network safety.

Exception could be made to this rule if an IETF standardized
mechanism is used to determine that a particular loss is due to
a non-congestion event (e.g., packet corruption). In such a
case a congestion control action is not required. Additionally,
RTO-triggered congestion control actions may be reversed when a
standard mechanism determines that the cause of the loss was not
congestion after all (e.g., [RFC5682]).

4 Discussion

We note that research has shown the tension between the
responsiveness and correctness of retransmission timeouts seems to
be a fundamental tradeoff in the context of TCP [AP99]. That is,
making the RTO more aggressive (e.g., via changing TCP's EWMA gains,
lowering the minimum RTO, etc.) can reduce the time spent waiting on
needed retransmissions. However, at the same time, such
aggressiveness leads to more needless retransmissions. Therefore,
being as aggressive as the requirements given in the previous
section allow in any particular situation may not be the best course
of action because an RTO expiration carries a requirement to invoke
a congestion response and hence slow transmission 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 than specified in [RFC6298]).
Further, a number of implementations use minimum RTOs that are less
than the 1 second specified in [RFC6298]. While the implication of
these deviations from the standard may be more spurious retransmits
(per [AP99]), we are aware of no large scale network safety issues
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 requirements fail safe in that they insist
on exponential backoff of the RTO and a transmission rate reduction.
Therefore, providing implementers more latitude than they have
traditionally been given in IETF specifications of RTO mechanisms

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.

5 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.

Acknowledgments

   This document benefits from years of discussions with Ethan Blanton,
   Sally Floyd, Jana Iyengar, Shawn Ostermann, Vern Paxson, and the
   members of the TCPM and TCP-IMPL working groups. Ran Atkinson,
   Yuchung Cheng, David Black, Gorry Fairhurst, Mirja Kuhlewind,
   Jonathan Looney and Michael Scharf provided useful comments on a
   previous version of this draft.

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.

   [RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140,
        April 1997.

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

   [RFC3124] Balakrishnan, H., S. Seshan, "The Congestion Manager", RFC
        2134, June 2001.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
        A., Peterson, J., Sparks, R., Handley, M., and E. Schooler,
        "SIP: Session Initiation Protocol", RFC 3261, June 2002.

   [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.

[RFC3940] Adamson, B., C. Bormann, M. Handley, J. Macker,

            "Negative-acknowledgment (NACK)-Oriented Reliable Multicast
            (NORM) Protocol", November 2004, RFC 3940.

   [RFC4340] Kohler, E., M. Handley, S. Floyd, "Datagram Congestion
            Control Protocol (DCCP)", March 2006, RFC 4340.

   [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.

   [RFC5740] Adamson, B., C. Bormann, M. Handley, J. Macker,
            "NACK-Oriented Reliable Multicast (NORM) Transport Protocol",
            November 2009, RFC 5740.

   [RFC6182] Ford, A., C. Raiciu, M. Handley, S. Barre, J. Iyengar,
            "Architectural Guidelines for Multipath TCP Development", March
            2011, RFC 6182.

   [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.

   [RFC6675] Blanton, E., M. Allman, L. Wang, I. Jarvinen, M. Kojo,
            Y. Nishida, "A Conservative Loss Recovery Algorithm Based on
            Selective Acknowledgment (SACK) for TCP", August 2012, RFC 6675.

   [RFC7323] Borman D., B. Braden, V. Jacobson, R. Scheffenegger, "TCP
            Extensions for High Performance", September 2014, RFC 7323.

Authors' Addresses

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

   EMail: mallman@icir.org
   http://www.icir.org/mallman