Internet Engineering Task Force
INTERNET-DRAFT
File: draft-ietf-tcpm-rto-consider-13.txt
Intended Status: Best Current Practice
Expires: November 8, 2020

M. Allman
ICSI

May 8, 2020

Requirements for Time-Based Loss Detection

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. 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/1id-abstracts.html

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

This Internet-Draft will expire on November 8, 2020.

Copyright Notice

Abstract

Many protocols must detect packet loss for various reasons (e.g., to ensure reliability using retransmissions or to understand the level of congestion along a network path). While many mechanisms have been designed to detect loss, protocols ultimately can only count on the passage of time without delivery confirmation to declare a

packet "lost". Each implementation of a time-based loss detection mechanism represents a balance between correctness and timeliness and therefore no implementation suits all situations. This document

provides high-level requirements for time-based loss detectors 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, RFC 2119 [RFC2119].

1 Introduction

Loss detection is a crucial activity for many protocols and applications and is generally undertaken for two major reasons:

(1) Ensuring reliable data delivery.

This requires a data sender to develop an understanding of which transmitted packets have not arrived at the receiver. This knowledge allows the sender to retransmit missing data.

(2) Congestion control.

Packet loss is often taken as an indication that the sender is transmitting too fast and is overwhelming some portion of the network path. Data senders can therefore use loss to trigger transmission rate reductions.

Various mechanisms are used to detect losses in a packet stream. Often we use continuous or periodic acknowledgments from the recipient to inform the sender's notion of which pieces of data are missing. However, despite our best intentions and most robust mechanisms we cannot place ultimate faith in receiving such acknowledgments, but can only truly depend on the passage of time. Therefore, our ultimate backstop to ensuring that we detect all loss is a timeout. 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 concludes the data was lost in transit.

The specifics of time-based loss detection schemes represent a tradeoff between correctness and responsiveness. In other words we wish to simultaneously:

- wait long enough to ensure the detection of loss is correct, and

- minimize the amount of delay we impose on applications (before repairing loss) and the network (before we reduce the

congestion).

Serving both of these goals is difficult as they pull in opposite directions [AP99]. By not waiting long enough to accurately

determine a packet has been lost we risk sending unnecessary ("spurious") retransmissions and needlessly lowering the transmission rate. By waiting long enough that we are unambiguously certain a packet has been lost we cannot repair losses in a timely manner and we risk prolonging network congestion.

Many protocols and applications use their own time-based loss detection mechanisms (e.g., TCP [RFC6298], SCTP [RFC4960], SIP [RFC3261]). At this point, our experience leads to a recognition that often specific tweaks that deviate from standardized time-based loss detectors do not materially impact network safety. Therefore, in this document we outline a set of high-level protocol-agnostic requirements for time-based loss detection. The intent is to provide a safe foundation on which implementations have the flexibility to instantiate mechanisms that best realize their specific goals.

2 Context

This document is different from from the way we ideally like to engineer systems. Usually, we strive to understand high-level requirements as a starting point. We then methodically engineer specific protocols, algorithms and systems that meet these requirements. Within the standards process we have derived many time-based loss detection schemes without benefit from some over-arching requirements document---because we had no idea how to write such a document! Therefore, we made the best specific decisions we could in response to specific needs.

At this point, however, the community's experience has matured to the point where we can define a set of high-level requirements for time-based loss detection schemes. We now understand how to separate the strategies these mechanisms use that are crucial for network safety from those small details that do not materially impact network safety. However, adding a requirements umbrella to a body of existing specifications is inherently messy and we run the risk of creating inconsistencies with both past and future mechanisms. The correct way to view this document is as the default case. Specifically:

- This document does not update or obsolete any existing RFC. These previous specifications---while generally consistent with the requirements in this document---reflect community consensus and this document does not change that consensus.

- The requirements in this document are meant to provide for network safety and, as such, SHOULD be used by all time-based loss detection mechanisms.

- The requirements in this document may not be appropriate in all cases and, therefore, inconsistent deviations may be necessary (hence the "SHOULD" in the last bullet). However, inconsistencies MUST be (a) explained and (b) gather consensus.

3 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 4:

(S.1) The requirements in this document apply only to the primary or last resort time-based loss detection.

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

(S.2) The requirements for time-based loss detection mechanisms in this document are for the primary or "last resort" loss detection mechanism whether the mechanism is the sole loss repair strategy or works in concert with other mechanisms.

While a straightforward time-based loss detector is sufficient for simple protocols like DNS [RFC1034,RFC1035], more complex protocols often use more advanced loss detectors to aid performance. For instance, TCP and SCTP have methods to detect (and repair) loss based on explicit endpoint state sharing [RFC2018,RFC4960,RFC6675]. Such mechanisms often provide more timely and precise results than time-based loss detectors. However, these mechanisms do not obviate the need for a "retransmission timeout" or "RTO" because---as we discuss in Section 1---only the passage of time can ultimately be relied upon to detect loss. In cases such as these, the time-based loss detector functions as a "last resort".

Also, note, that some recent proposals have incorporated time as a component of advanced loss detection methods---either as an aggressive first loss detector or in conjunction with endpoint state sharing [DCCM13,CCDJ20,IS20]. Since these timers are not used as "last resort" the requirements in this document need not be directly used in these cases. However, we expect that many of the requirements are useful for these situations, as well.

(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 addresses B and C, A needs to use independent time-based loss detector instances for traffic sent to B and C.

(S.4) There are cases where state is shared across connections

or flows (e.g., [RFC2140], [RFC3124]). State pertaining to time-based loss detection is often discussed as sharable. These situations raise issues that the simple flow-oriented time-based loss detection 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 4 are likely applicable, sharing time-based loss detection information across flows is outside the scope of this document.

4 Requirements

We now list the requirements that apply when designing primary or last resort time-based loss detection mechanisms. For historical reasons and ease of exposition, we refer to the time between sending a packet and determining the packet has been lost due to lack of delivery confirmation as the "retransmission timeout" or "RTO". After the RTO passes without delivery confirmation, the sender may safely assume the packet is lost. However, as discussed above, the detected loss need not be repaired (i.e., the loss could be detected only for congestion control and not reliability purposes).

(1) As we note above, loss detection happens when a sender does not receive delivery confirmation within an some expected period of time. In the absence of any knowledge about the latency of a path, the initial RTO MUST be conservatively set to no less than 1 second.

Correctness is of the utmost importance when transmitting into a network with unknown properties because:

- Premature loss detection can trigger spurious retransmits that could cause issues when a network is already congested.

- Premature loss detection can needlessly cause congestion control to dramatically lower the sender's allowed transmission rate---especially since the rate is already likely low at this stage of the communication. Recovering from such a rate change can taken a relatively long time.

- Finally, as discussed below, sometimes using time-based loss detection and retransmissions can cause 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) We now specify four requirements that pertain to setting an expected time interval for delivery confirmation.

Often measuring the time required for delivery confirmation is

is framed as assessing 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 and the expected latency is better framed as the "feedback time" (FT). In other words, the expectation is not always simply a network property, but can include additional time before a sender should reasonably expect a response.

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 network RTT between the client and resolver.

Therefore, we express the requirements in terms of FT. Again, for ease of exposition we use "RTO" to indicate the interval between a packet transmission and the decision the packet has been lost---regardless of whether the packet will be retransmitted.

(a) If/when available, the RTO SHOULD be set based on multiple observations of the FT.

    In other words, the RTO should represent an empirically-derived reasonable amount of time that the sender should wait for delivery confirmation before deciding the given data is lost. Network paths are inherently dynamic and therefore it is crucial to incorporate multiple FT samples in the RTO to take into account the delay variation across time.

    For example, TCP's RTO [RFC6298] would satisfy this requirement due to its use of an EWMA to combine multiple FT samples into a "smoothed RTT". In the name of conservativeness, TCP goes further to also include an explicit variance term when computing the RTO.

(b) FT observations SHOULD be taken and incorporated into the RTO at least once per RTT or as frequently as data is exchanged in cases where that happens less frequently than once per RTT.

    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.

As an example, 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, 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 either (a) the subsequent successful transmission of non-retransmitted data, or (b) an RTO passes without detecting additional losses. The former will generally be quicker. The latter covers cases where loss is detected, but not repaired.

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

This ensures network safety.

(4) Loss detected by the RTO mechanism MUST be taken as an

indication 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.

An exception to this rule is if an IETF standardized mechanism determines 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, congestion control actions taken based on time-based loss detection could be reversed when a standard mechanism post-facto determines that the cause of the loss was not congestion (e.g., [RFC5682]).

5 Discussion

We note that research has shown the tension between the responsiveness and correctness of time-based loss detection 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 exponentially weighted moving average (EWMA) gains, lowering the minimum RTO, etc.) can reduce the time required to detect actual loss. However, at the same time, such aggressiveness leads to more cases of mistakenly declaring packets lost that ultimately arrived at the receiver. 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 detecting loss---even if falsely---carries a requirement to invoke a congestion response which will ultimately reduce the transmission rate.

While the tradeoff between responsiveness and correctness seems fundamental, the tradeoff can be made less relevant if the sender can detect and recover from mistaken loss detection. 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 mistakenly declaring packets to be lost.

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 a steady-state minimum RTO that are less than the 1 second specified in [RFC6298] (which is different from the initial RTO we specify in Section 4, Requirement 1). 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 could in fact increase the number of needless

retransmissions the above requirements fail safe in that they insist on exponential backoff 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.

6 Security Considerations

This document does not alter the security properties of time-based loss detection 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, Martin Duke, Gorry Fairhurst, Rahul Arvind Jadhav, Mirja Kuhlewind, Nicolas Kuhn, Jonathan Looney and Michael Scharf provided useful comments on previous versions of this document.

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.

[CCDJ20] Cheng, Y., N. Cardwell, N. Dukkipati, P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", Internet-Draft draft-ietf-tcpm-rack-08.txt (work in progress), March 2020.

[DCCM13] Dukkipati, N., N. Cardwell, Y. Cheng, M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses", Internet-Draft draft-dukkipati-tcpm-tcp-loss-probe-01.txt (work in progress), February 2013.

[IS20] Iyengar, I., I. Swett, "QUIC Loss Detection and Congestion Control", Internet-Draft draft-ietf-quic-recovery-27.txt (work in progress), March 2020.

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

[RFC1034] Mockapetris, P. "Domain Names - Concepts and Facilities", RFC 1034, November 1987.

[RFC1035] Mockapetris, P. "Domain Names - Implementation and Specification", RFC 1035, November 1987.

[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