Stream: Internet Engineering Task Force (IETF)
RFC: 8961
BCP: 233
Category: Best Current Practice
Published: November 2020
ISSN: 2070-1721
Author: M. Allman
ICSI

RFC 8961
Requirements for Time-Based Loss Detection

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, ultimately, protocols 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; therefore, no implementation suits all situations. This document provides high-level requirements for time-based loss detectors appropriate for general use in unicast communication across the Internet. Within the requirements, implementations have latitude to define particulars that best address each situation.

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8961.

Copyright Notice

Table of Contents

1. Introduction
   1.1. Terminology
2. Context
3. Scope
4. Requirements
5. Discussion
6. Security Considerations
7. IANA Considerations
8. References
   8.1. Normative References
   8.2. Informative References

Acknowledgments

Author's Address

1. Introduction

As a network of networks, the Internet consists of a large variety of links and systems that support a wide variety of tasks and workloads. The service provided by the network varies from best-effort delivery among loosely connected components to highly predictable delivery within controlled environments (e.g., between physically connected nodes, within a tightly controlled data center). Each path through the network has a set of path properties, e.g., available capacity, delay, and packet loss. Given the range of networks that make up the Internet, these properties range from largely static to highly dynamic.

This document provides guidelines for developing an understanding of one path property: packet loss. In particular, we offer guidelines for developing and implementing time-based loss detectors that have been gradually learned over the last several decades. We focus on the general case where the loss properties of a path are (a) unknown a priori and (b) dynamically varying over time. Further, while there are numerous root causes of packet loss, we leverage the conservative notion that loss is an implicit indication of congestion [RFC5681]. While this stance is not always correct, as a general assumption it has historically served us well [Jac88]. As we

discuss further in Section 2, the guidelines in this document should be viewed as a general default for unicast communication across best-effort networks and not as optimal -- or even applicable -- for all situations.

Given that packet loss is routine in best-effort networks, 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

As we mention above, packet loss is often taken as an implicit 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 may provide a needed retransmission in a timely manner but risk both 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 -- such as TCP [RFC6298], SCTP [RFC4960], and SIP [RFC3261] -- use their own time-based loss detection mechanisms. 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 with respect to congestion control [AP99]. 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.

1.1. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Context

This document is different 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 IETF standards process, we have derived many time-based loss detection schemes without the benefit of 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 general, 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. The requirements in this document may not be appropriate in all cases. In particular, the guidelines in Section 4 are concerned with the general case, but specific situations may allow for more flexibility in terms of loss detection because specific facets of the environment are known (e.g., when operating over a single physical link or within a tightly controlled data center). Therefore, variants, deviations, or wholly different time-based loss detectors may be necessary or useful in some cases. The correct way to view this document is as the default case and not as one-size-fits-all guidance that is optimal in all cases.

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. Therefore, we make the following statements about the relationship of this document to past and future specifications:

- 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 future time-based loss detection mechanisms.
- The requirements in this document may not be appropriate in all cases; therefore, deviations and variants may be necessary in the future (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) While there are a bevy of uses for timers in protocols -- from rate-based pacing to connection failure detection and beyond -- this document is focused only on loss detection.

(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 loss detection 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 other words, we ultimately cannot count on acknowledgments to arrive at the data sender to indicate which packets never arrived at the receiver. In cases such as these, we need a time-based loss detector to function 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 in certain situations or in conjunction with endpoint state sharing [DCCM13] [CCDJ20] [IS20]. While these mechanisms can aid timely loss recovery, the protocol ultimately leans on another more conservative timer to ensure reliability when these mechanisms break down. The requirements in this document are only directly applicable to last-resort loss detection. However, we expect that many of the requirements can serve as useful guidelines for more aggressive non-last-resort timers 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 Multipath TCP (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. That is, 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] and [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 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 take 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 round-trip times (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 framed as assessing the RTT of the network path. The RTT 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 misleading, and the

expected latency is better framed as the "feedback time" (FT). In other words, the expectation is not always simply a network property; it 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 [RFC1035]. When the request can be served from the resolver's cache, the feedback time (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 that the packet has been lost, regardless of whether the packet will be retransmitted.

(a) The RTO **SHOULD** be set based on multiple observations of the FT when available.

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; therefore, it is crucial to incorporate multiple recent 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 exponentially weighted moving average (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.

While multiple FT samples are crucial for capturing the delay dynamics of a path, we explicitly do not tightly specify the process -- including the number of FT samples to use and how/when to age samples out of the RTO calculation -- as the particulars could depend on the situation and/or goals of each specific loss detector.

Finally, FT samples come from packet exchanges between peers. We encourage protocol designers -- especially for new protocols -- to strive to ensure the feedback is not easily spoofable by on- or off-path attackers such that they can perturb a host's notion of the FT. Ideally, all messages would be cryptographically secure, but given that this is not always possible -- especially in legacy protocols -- using a healthy amount of randomness in the packets is encouraged.

(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 non-data exchanges for various reasons, e.g., keepalives, heartbeats, and control messages. To the extent that the latency of these exchanges 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 packet X are transmitted at times t0 and t1. 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; hence, the actual FT is either t2-t1 or t2-t0, but which is a mystery. Therefore, in this situation, an implementation **MUST NOT** use either version of the FT sample and hence not update the RTO (as discussed in [KP87] and [RFC6298]).

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. For example, TCP's timestamp option [RFC7323] allows for packets 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) 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 packet [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]).

(4) 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.

As with guideline (3), an exception to this rule exists if an IETF standardized mechanism determines that a particular loss is not due to congestion.

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 that 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], Forward RTO-Recovery (F-RTO) [RFC5682], and Duplicate Selective Acknowledgement (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 TCP implementations use a steady-state minimum RTO that is 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. This informs the guidelines in the last section (e.g., there is no minimum RTO specified).

Finally, we note that while allowing implementations to be more aggressive could in fact increase the number of needless retransmissions, the above requirements fail safely 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.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

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

[CCDJ20] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "The RACK-TLP loss detection algorithm for TCP", Work in Progress, Internet-Draft, draft-ietf-tcpm-rack-13, 2 November 2020, <https://tools.ietf.org/html/draft-ietf-tcpm-rack-13>.

[DCCM13] Dukkipati, N., Cardwell, N., Cheng, Y., and M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses", Work in Progress, Internet-Draft, draft-dukkipati-tcpm-tcp-loss-probe-01, 25 February 2013, <https://tools.ietf.org/html/draft-dukkipati-tcpm-tcp-loss-probe-01>.

[IS20] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", Work in Progress, Internet-Draft, draft-ietf-quic-recovery-32, 20 October 2020, <https://tools.ietf.org/html/draft-ietf-quic-recovery-32>.

[Jac88] Jacobson, V., "Congestion avoidance and control", ACM SIGCOMM, DOI 10.1145/52325.52356, August 1988, <https://doi.org/10.1145/52325.52356>.

[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", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, DOI 10.17487/RFC2018, October 1996, <https://www.rfc-editor.org/info/rfc2018>.

[RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140, DOI 10.17487/RFC2140, April 1997, <https://www.rfc-editor.org/info/rfc2140>.

[RFC2883] Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", RFC 2883, DOI 10.17487/RFC2883, July 2000, <https://www.rfc-editor.org/info/rfc2883>.

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, DOI 10.17487/RFC3124, June 2001, <https://www.rfc-editor.org/info/rfc3124>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3522] Ludwig, R. and M. Meyer, "The Eifel Detection Algorithm for TCP", RFC 3522, DOI 10.17487/RFC3522, April 2003, <https://www.rfc-editor.org/info/rfc3522>.

[RFC3708] Blanton, E. and 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, DOI 10.17487/RFC3708, February 2004, <https://www.rfc-editor.org/info/rfc3708>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC5682] Sarolahti, P., Kojo, M., Yamamoto, K., and M. Hata, "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP", RFC 5682, DOI 10.17487/RFC5682, September 2009, <https://www.rfc-editor.org/info/rfc5682>.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <https://www.rfc-editor.org/info/rfc5740>.

[RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, DOI 10.17487/RFC6182, March 2011, <https://www.rfc-editor.org/info/rfc6182>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/info/rfc6298>.

[RFC6675] Blanton, E., Allman, M., Wang, L., Jarvinen, I., Kojo, M., and Y. Nishida, "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP", RFC 6675, DOI 10.17487/RFC6675, August 2012, <https://www.rfc-editor.org/info/rfc6675>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

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 TCPIMPL Working Groups. Ran Atkinson, Yuchung Cheng, David Black, Stewart Bryant, Martin Duke, Wesley Eddy, Gorry Fairhurst, Rahul Arvind Jadhav, Benjamin Kaduk, Mirja Kühlewind, Nicolas Kuhn, Jonathan Looney, and Michael Scharf provided useful comments on previous draft versions of this document.

Author's Address

Mark Allman
International Computer Science Institute
2150 Shattuck Ave., Suite 1100
Berkeley, CA 94704
United States of America
Email: mallman@icir.org
URI: https://www.icir.org/mallman