Internet Congestion Control
Internet-Draft
Intended status: Informational
Expires: 25 July 2026

T. Li
Renmin University of China
K. Xu
Tsinghua University
B. Wu
Tencent
Y. Zhao
Tsinghua University

21 January 2026

Minimum RTT Estimation Under Low ACK Frequency
draft-li-iccrg-minimum-rtt-estimation-00

Abstract

In traditional acknowledgment mechanisms, the sender frequently "pulls" ACK packets, resulting in significant protocol control overhead. This leads to wasted CPU and I/O resources, contention for packet spectrum on half-duplex links (e.g., WLAN), and reverse-path congestion in asymmetric links (e.g., satellite network). Reducing the number of ACKs is essential in scenarios where ACK overhead is non-negligible. However, a lower ACK frequency can introduce biases in delay estimation, such as overestimating the minimum round-trip time (minRTT). This document proposes how to calibrate the estimation of the minRTT under low ACK frequency conditions.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 25 July 2026.

Copyright Notice

Table of Contents

1. Requirements Language

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 [RFC2119].

2. Overview of Standards on ACK Mechanism

[RFC9000] specifies a simple delayed ACK mechanism that a receiver can send an ACK for every other packet, and for every packet when reordering is observed, or when the max_ack_delay timer expires. However, this ACK mechanism may not match the number of ACKs to the transport's required intensity under different network conditions. For example, when the data throughput of a WLAN transport is extremely high, QUIC will generate a large number of ACKs. In this case, minimizing the ACK intensity of QUIC is not only a win for data throughput improvement but also a win for energy and CPU efficiency.

[RFC1122] and [RFC5681] were two core functionality standards that introduced delayed ACK, which was the default acknowledgment mechanism in most Linux distributions. [RFC4341] and [RFC5690] described an acknowledgment congestion control mechanism in which the minimum ACK frequency allowed is twice per send window. [RFC3449] discussed the imperfection and variability of TCP's acknowledgment mechanism because of asymmetric effects and recommended scaling ACK frequency as a mitigation to these effects. These RFCs reveal that the dependence on frequent ACKs is an artifact of current transport protocol designs rather than a fundamental requirement. Based on this insight, some work-in-progress IETF drafts have paid great attention to ACK scaling technologies in both TCP and QUIC working groups.

First of all, [ACK-PULL] proposed the TCP ACK pull mechanism, which allows a sender to request the ACK for a data segment to be sent without additional delay by the receiver. This helps in some cases when the delayed ACKs degrade transport performance.

Instead of pulling more ACKs, [QUIC-SCALING] recommended reducing the ACK frequency by sending an ACK for at least every 10 received packets and [QUIC-SATCOM] recommended an ACK frequency of four ACKs every round-trip time (RTT), aiming to reduce link transmission costs for asymmetric paths.

Different from using an empirical value of ACK frequency, instead, Li tried to improve the scalability by proposing a novel ACK mechanism named Tame ACK (TACK), whose frequency is a function of the bandwidth-delay product of network connections. The detailed TCP-based implementation (i.e., TCP-TACK) details and evaluation results in industrial applications have been shown in [LI-TACK].

The recent work [IYENGAR-ACK] has been adopted in QUIC working group, it specifies an extension to QUIC that enables an endpoint to request its peer change its behavior when acknowledging at a low ACK frequency.

Based on the observations, low ACK frequency not only addresses an industry-wide need but also demonstrates broad applicability across diverse operational environments.

3. Problem Statement

In this document, the minRTT is the minimum RTT samples observed at the sender for a given network path during a period of time. The minRTT is a fundamental state variable in transmission control, and its accuracy is critical for achieving both high throughput and low latency. For example, some congestion controllers depend on minRTT to estimate the congestion window [CARDWELL-BBR]. minRTT is also used by QUIC loss detection to reject implausibly small RTT samples [RFC9002]. minRTT is further used to update the ACK frequency in [LI-TACK].

An RTT estimation system contains a sender and a receiver. Ideally, when sending an ACK for every data packet, the minimum RTT sample can be computed by monitoring the per-packet RTT samples. However, the sender can hardly generate per-packet RTT samples in the case of sending fewer ACKs, which is the root cause of the minimum RTT estimation biases. When multiple packets carrying departure timestamps are transported between endpoints via the same path, an RTT of this path can be sampled at the sender upon receiving an ACK frame. However, when sending fewer ACK frames, more data packets might be received during the ACK interval, generating only one RTT sample among multiple packets is likely to result in biases. For example, a larger minimum RTT estimate. In general, the higher the throughput, the larger the biases. Experiments in [LI-TACK] show that the default way of RTT sampling suffers 8% 釿シ 18% larger minRTT estimates.

One alternative way to reduce biases can be that, each ACK frame carries multiple timestamps for the sender to generate more RTT samples. For example, the recent draft [SWETT-TS] has been adopted in QUIC working group, this document defines an extension to the QUIC transport protocol which supports reporting multiple packet receive timestamps in a best-effort way. However, (1) the overhead might be too high to be acceptable for high-bandwidth transport. Also, (2) the number of data packets might be far more than the maximum number of timestamps that an ACK frame is capable of carrying. This issue is unremarkable for the basic RTT estimation (e.g., smoothed RTT) when throughput is low. However, the minRTT might be overestimated because that the sender can hardly generate per-packet RTT samples when the throughput is hign.

4. MinRTT Estimation Under Low ACK Frequency

Since the receiver is capable of monitoring per-packet state, the one-way delay (OWD) of each packet can be easily computed according to the departure timestamps (carried in the TIMESTAMP frame) and the arrival timestamps of each packet. In this case, QUIC SHOULD adopt the OWD-based minRTT estimation.

The rationale is that the variation of OWD reflects the variation of RTT over near-symmetric links. The OWD-based minRTT estimation requires the sender to record the departure timestamp in each ack-eliciting packet. Meanwhile, at the receiver, the per-packet OWD samples SHOULD be computed upon packet arrivals and a function of computing the minimum OWD (minOWD) SHOULD be newly added. In this document, minOWD is the minimum OWD samples observed on the same network path during a period of time. The receiver then generates an ACK frame for the sender, in which the ACK delay and departure timestamp for the packet that achieves the minimum OWD is reported. The ACK delay is defined as the delay incurred between when the packet is received and when the ACK frame is sent. Based on the information reported by the incoming ACK frames and the ACK arrival timestamps, the sender can generate RTT samples and then compute minRTT accordingly.

4.1. Sender-Side Operation

Generally, the minRTT is calculated at the sender.

Before estimating the minRTT, the RTT samples should be computed based on the ACK frames collected during a period. Assume that a packet is sent by the sender at time t_1 and arrives at time t_3, and the ACK frame is sent at time t_4. The ACK delay can be computed at the receiver. For example, the receiver computes the ACK delay delta_t = t_4 - t_3, and syncs the ACK delay to the sender via an ACK frame. The ACK delay can also be computed at the sender. For example, the receiver directly syncs an ACK frame carrying t_4 and t_3 to the sender, the sender then computes the ACK delay delta_t = t_4 - t_3.

The sender therefore computes an RTT sample according to delta_t, t_1, and the arrival time (t_2) of the ACK frame, i.e., RTT_sample = t_2 - t_1 - delta_t. Measuring delta_t at the receiver assures an explicit correction for a more accurate RTT estimate. RTT samples SHOULD be smoothed using exponentially weighted moving average (EWMA) as specified in [RFC6298]. The sender then computes the minRTT according to these RTT samples during a period.

The sender SHOULD insert a TIMESTAMP frame into each packet for measuring OWD at the receiver.

4.2. Receiver-side Operation

As specified in [RFC9000], by default the QUIC receiver reports ACK delays for only the largest acknowledged packet in an ACK frame, hence an RTT sample is generated using only the largest acknowledged packet in the received ACK frame. For a more accurate minRTT estimate when sending fewer ACK frames, this document requires the QUIC receiver to filter the departure timestamp for the packet who achieves the minOWD during the interval between two ACK frames and report the ACK delay of this packet.

Hence, to achieve the OWD-based minRTT estimation, the QUIC receiver is RECOMMENDED to send an MINOWD-ACK frame periodically to report ACK delay and the departure timestamp of the packet who achieves the minOWD.

Upon packet arrivals, the receiver is capable of generating per-packet OWD samples according to the difference between the packet departure timestamp and packet arrival timestamp. The receiver then computes the minOWD by comparing the per-packet OWD samples. The OWD estimation does not require clock synchronization here because the relative values are adopted.

Afterwards, based on the ACK delay and the departure timestamp corresponding to the packet that achieves the minOWD, the sender calculates the RTT of this packet as a minimum RTT sample. Ultimately, the minRTT is computed according to these minimum RTT samples.

The MINOWD-ACK frame is RECOMMENDED to send at most once an RTT to limit the acknowledgment frequency.

5. Modification to QUIC Protocol

5.1. Transport Parameter: timestamp-support

A new field named timestamp-support should be added for negotiation between both parties on whether to sync packet departure timestamps in QUIC connection. The endpoints send this parameter during handshakes. Only when both parties agree, packet departure timestamp synchronization can be adopted.

timestamp-support (0x XX): This parameter has two values (0 or 1) specifying whether the sending endpoint is willing to sync packet departure timestamps. When the value is set as 1, it means that the

sending endpoint wants to sync packet departure timestamps during connection. When the value is set as 0, it means that the sending endpoint does not support this function.

5.2. TIMESTAMP Frame

Instead of the invasive way of adding a new field in the QUIC public packet header, it is RECOMMENDED that a new frame be added for exchanging the departure timestamp of each packet.

It is worth noting that [HUITEMA-TS] has proposed an extension of TIMESTAMP frame for QUIC, which can be reused for minRTT estimation in this document.

Based on [HUITEMA-TS], a TIMESTAMP frame is shown in Figure 1.

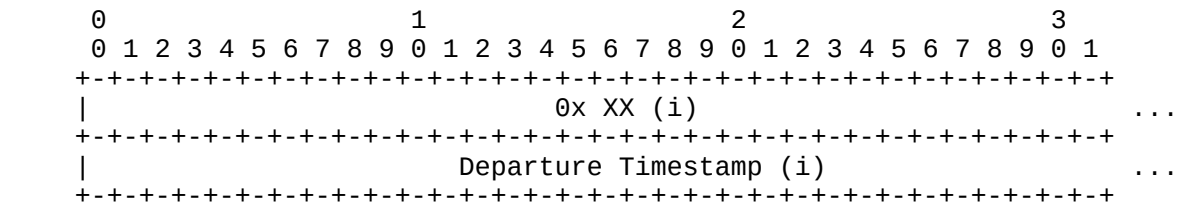

Figure 1: TIMESTAMP Frame

A TIMESTAMP frame contains the following fields:

Departure Timestamp: An integer indicating the departure time of a packet.

QUIC SHOULD carry the TIMESTAMP Frame in each packet.

5.3. MINOWD-ACK Frame

Instead of the invasive way of redefining the ACK Delay field in the QUIC ACK frame, it is RECOMMENDED that a new MINOWD-ACK frame be added for minRTT estimation.

A MINOWD-ACK frame is shown in Figure 2.

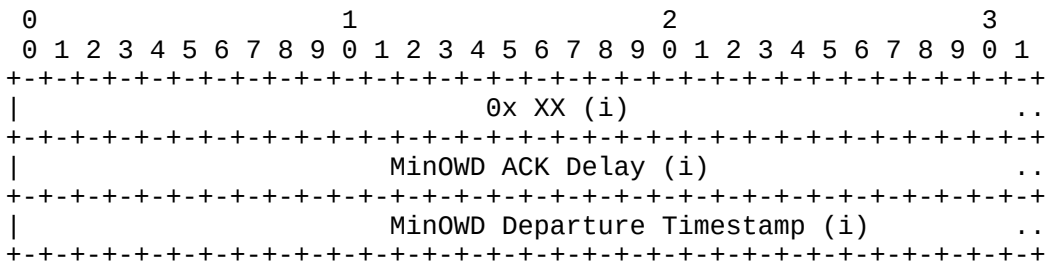

Figure 2: MINOWD-ACK Frame

A MINOWD-ACK frame contains the following fields:

MinOWD ACK Delay: An integer indicating the MINOWD-ACK Delay. When generating the MINOWD-ACK frame, QUIC SHOULD change the way of computing MINOWD-ACK Delay according to the arrival timestamp of the packet with minimum OWD instead of the arrival timestamp of the largest acknowledged packet.

MinOWD Departure Timestamp: An integer indicating the departure time of the packet who achieves the mimOWD.

6. Security Considerations

TBD

7. IANA Considerations

The values for the timestamp-support transport parameter, TIMESTAMP frame, and MINOWD-ACK frame should be allocated.

8. Acknowledgements

Thanks to Kai Zheng, Kun Tan, Rahul Arvind Jadhav, Jiao Kang, Keith Winstein, Marco Munizaga, Marten Seemann, and Christian Huitema for their reviews and suggestions.

9. References

9.1. Normative References

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[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>.

[RFC3449] Balakrishnan, H., Padmanabhan, V., Fairhurst, G., and M. Sooriyabandara, "TCP Performance Implications of Network Path Asymmetry", BCP 69, RFC 3449, DOI 10.17487/RFC3449, December 2002, <https://www.rfc-editor.org/info/rfc3449>.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, DOI 10.17487/RFC4341, March 2006, <https://www.rfc-editor.org/info/rfc4341>.

[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>.

[RFC5690] Floyd, S., Arcia, A., Ros, D., and J. Iyengar, "Adding Acknowledgement Congestion Control to TCP", RFC 5690, DOI 10.17487/RFC5690, February 2010, <https://www.rfc-editor.org/info/rfc5690>.

[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>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

9.2. Informative References

[ACK-PULL] Gomez, C., Ed. and J. Crowcroft, Ed., "TCP ACK Pull", Work in Progress, Internet-Draft, draft-gomez-tcpm-ack-pull-01, 4 November 2019, <https://datatracker.ietf.org/doc/html/draft-gomez-tcpm-ack-pull-01>.

[CARDWELL-BBR]
Cardwell, N., Cheng, Y., Gunn, C S., Yeganeh, S. H., and V. Jacobson, "BBR: Congestion-based congestion control", ACM QUEUE 14(5):20-53, 2016.

[HUITEMA-TS]
Huitema, C., Ed., "Quic Timestamps For Measuring One-Way Delays", Work in Progress, Internet-Draft, draft-huitema-quic-ts-08, 1 March 2023, <https://datatracker.ietf.org/doc/html/draft-huitema-quic-ts-08>.

[IYENGAR-ACK]
Iyengar, J., Ed., Swett, I., Ed., and M. Khlewind, Ed., "QUIC Acknowledgment Frequency", Work in Progress, Internet-Draft, draft-ietf-quic-ack-frequency-13, 6 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-13>.

[LI-TACK] Li, T., Zheng, K., Xu, K., Jadhav, R. A., Xiong, T., Winstein, K., and K. Tan, "TACK: Improving Wireless Transport Performance by Taming Acknowledgments", ACM SIGCOMM 2020:15-30, 2020.

[QUIC-SATCOM]
Kuhn, N., Ed., Fairhurst, G., Ed., Border, J., Ed., and E. Stephan, Ed., "QUIC for SATCOM", Work in Progress, Internet-Draft, draft-kuhn-quic-4-sat-06, 30 October 2020, <https://datatracker.ietf.org/doc/html/draft-kuhn-quic-4-sat-06>.

[QUIC-SCALING]
Fairhurst, G., Ed., Custura, A., Ed., and T. Jones, Ed., "Changing the Default QUIC ACK Policy", Work in Progress, Internet-Draft, draft-fairhurst-quic-ack-scaling-03, 14 September 2020, <https://datatracker.ietf.org/doc/html/draft-fairhurst-quic-ack-scaling-03>.

[SWETT-TS] Swett, I., Ed. and J. Beshay, Ed., "QUIC Extended Acknowledgement for Reporting Packet Receive Timestamps", Work in Progress, Internet-Draft, draft-ietf-quic-receive-ts-00, 3 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-receive-ts-00>.

Authors’ Addresses

Tong Li
Renmin University of China
Email: tong.li@ruc.edu.cn

Ke Xu
Tsinghua University
Email: xuke@tsinghua.edu.cn

Bo Wu
Tencent
Email: brynwu@tencent.com

Youjian Zhao
Tsinghua University
Email: zhaoyoujian@tsinghua.edu.cn