QUIC
Internet-Draft
Intended status: Informational
Expires: 22 January 2026

C. Smith
NVIDIA
I. Swett, Ed.
Google LLC
J. Beshay, Ed.
S. Jaiswal, Ed.
Meta Platforms, Inc.
21 July 2025

QUIC Extended Acknowledgement for Reporting Packet Receive Timestamps draft-smith-quic-receive-ts-03

Abstract

This document defines an extension to the QUIC transport protocol which supports reporting multiple packet receive timestamps for post-handshake packets.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the QUIC Working Group mailing list (quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/.

Source for this draft and an issue tracker can be found at https://github.com/wcsmith/draft-quic-receive-ts.

Status of This Memo

Table of Contents

1. Introduction

The QUIC Transport Protocol [RFC9000] provides a secure, multiplexed connection for transmitting reliable streams of application data.

This document defines an extension to the QUIC transport protocol which supports reporting multiple packet receive timestamps.

2. Motivation

QUIC congestion control ([RFC9002]) supports sampling round-trip time (RTT) by measuring the time from when a packet was sent to when it is acknowledged. However, more precise delay signals measured via packet receive timestamps have the potential to improve the accuracy of network bandwidth measurements and the effectiveness of congestion control, especially for latency-critical applications such as real-time video conferencing or game streaming.

Numerous existing algorithms and techniques leverage receive receive timestamps to improve transport performance. Examples include:

* The WebRTC congestion control algorithm described in [I-D.ietf-rmcat-gcc] uses the difference between packet inter-departure and packet inter-arrival times as the input to its delay-based controller.

* The pathChirp ([RRBNC]) technique estimates available bandwidth by measuring inter-arrival time of multiple packets.

Notably, these techniques require receive timestamps for more than one packet per round-trip in order to best measure the network.

Additionally, receive timestamps can provide valuable network telemetry, even if they are not used by the congestion controller.

3. Conventions and Definitions

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.

4. ACK Frame Wire Format

Endpoints send ACK frames in 1-RTT packets as they otherwise would, with 0 or more receive timestamps following the Ack Ranges and optional ECN Counts. Receive timestamps MUST NOT be sent in Initial or Handshake packets, because the peer would not know to use the extended wire format. ACK frames are never sent in 0-RTT packets, so there is no change to 0-RTT.

Once negotiated, the ACK format is identical to RFC9000, but with an additional section for receive timestamps at the end:

```
ACK Frame {
  Type (i) = 0x02..0x03,
  Largest Acknowledged (i),
  ACK Delay (i),
  ACK Range Count (i),
  First ACK Range (i),
  ACK Range (..) ...,
  [ECN Counts (..)],
  // Timestamp Extension, see {{ts-ranges}}
  Receive Timestamps (..)
}
```

Figure 1: ACK Frame Format

The fields Largest Acknowledged, ACK Delay, ACK Range Count, First ACK Range, ACK Range and ECN Counts are the same as for ACK (type=0x02..0x03) frames specified in Section 19.3 of [RFC9000].

The format of the Receive Timestamps field is shown in Figure 2.

```
Receive Timestamps {
  Timestamp Range Count (i),
  Timestamp Range (..) ...
}
```

Figure 2: Receive Timestamps Fields

Timestamp Range Count: A variable-length integer specifying the number of Timestamp Range fields in the frame.

Timestamp Ranges: Ranges of receive timestamps for contiguous packets in descending packet number order; see Section 4.1.

4.1. Timestamp Ranges

Each Timestamp Range describes a series of contiguous packet receive timestamps in descending sequential packet number (and descending timestamp) order. Timestamp Ranges consist of a Delta Largest Acknowledged indicating the largest packet number in the range, followed by a list of Timestamp Deltas describing the relative receive timestamps for each contiguous packet in the Timestamp Range (descending). Packets within a range are in descending packet number and timestamp order. Ranges are in descending timestamp order but do not have to be in descending packet number order.

Timestamp Ranges are structured as shown in Figure 3.

```
Timestamp Range {
  Delta Largest Acknowledged (i),
  Timestamp Delta Count (i),
  Timestamp Delta (i) ...,
}
```

Figure 3: Timestamp Range Format

The fields that form each Timestamp Range are:

Delta Largest Acknowledged: A variable-length integer indicating the largest packet number in the Timestamp Range as a delta to subtract from the Largest Acknowledged in the ACK frame. For example, 0 indicates the range starts with the Largest Acknowledged.

Timestamp Delta Count: A variable-length integer indicating the number of Timestamp Deltas in the current Timestamp Range.

The sum of Timestamp Delta Counts for all Timestamp Ranges in the frame MUST NOT exceed max_receive_timestamps_per_ack as specified in Section 5.

Timestamp Deltas: Variable-length integers encoding the receive timestamp for contiguous packets in the Timestamp Range in descending packet number order as follows:

For the first Timestamp Delta of the first Timestamp Range in the frame: the value is the difference between (a) the receive timestamp of the largest packet in the Timestamp Range (indicated by Gap) and (b) the session receive_timestamp_basis (see Section 5.2), decoded as described below.

For all other Timestamp Deltas: the value is the difference between (a) the receive timestamp specified by the previous Timestamp Delta and (b) the receive timestamp of the current packet in the Timestamp Range, decoded as described below.

All Timestamp Delta values are decoded by mulitplying the value in the field by 2 to the power of the receive_timestamps_exponent transport parameter received by the sender of the ACK frame (see Section 5):

When the receiver receives packets out-of-order, it SHOULD report them with other packets in a single ACK frame, starting with the most recently received packet regardless of the packet number order. See Section 7 for examples of reporting timestamps of out-of-order packets.

5. Extension Negotiation

max_receive_timestamps_per_ack (0xff0a002 temporary value for draft use): A variable-length integer indicating that the maximum number of receive timestamps the sending endpoint would like to receive in an ACK frame.

Each ACK frame sent MUST NOT contain more than the peer's maximum number of receive timestamps.

receive_timestamps_exponent (0xff0a003 temporary value for draft use): A variable-length integer indicating the exponent to be used when encoding and decoding timestamp delta fields in ACK frames sent by the peer (see Section 4.1). If this value is absent, a default value of 0 is assumed (indicating microsecond precision). Values above 20 are invalid.

5.1. Multiple Extensions to the ACK Frame

Multiple extensions can alter the ACK Frame or define new codepoints for variations on the ACK frame, such as [MP-QUIC]. Each extension defines how it co-exists with past extensions. If multiple extensions add more information to the ACK Frame, as this receive timestamp extension does, the additional extensions are appended at the end of the ACK Frame in the order of their RFC number, unless otherwise specified.

5.2. Receive Timestamp Basis

Endpoints which negotiate the extension need to determine a value, receive_timestamp_basis, relative to which all receive timestamps for the session will be reported (see Section 4.1).

The value of receive_timestamp_basis MUST be less than the smallest receive timestamp reported, and MUST remain constant for the entire duration of the session. The receive_timestamp_basis is a local value that is not communicated to the peer.

Receive timestamps are reported relative to the basis, rather than in absolute time to avoid requiring clock synchronization between endpoints and to make the frame more compact.

6. Discussion

6.1. Best-Effort Behavior

Receive timestamps are sent on a best-effort basis. Endpoints MUST gracefully handle scenarios where the receiver does not communicate receive timestamps for acknowledged packets. Examples of such scenarios are:

* A packet containing an ACK frame is lost.

* The sender truncates the number of timestamps sent in order to (a) avoid sending more than max_receive_timestamps_per_ack (Section 5); or (b) fit the ACK frame into a packet.

7. Examples

To illustrate the usage of the Receive Timestamps fields, consider a peer that sent 14 packets with numbers 87 to 100.

Assume the receiver receives packets 87 to 91 and 96 to 100 at the following timestamps relative to the basis:

Packet Number	Relative Timestamp
87	300
88	305
89	310
90	320
91	330
96	350
97	355
98	360
99	370
100	380

Table 1

When it’s time to acknowledge these packets, the receiver will send an ACK frame with two ranges, as follows:

```
Largest Acknowledged: 100
...
Timestamp Ranges Count: 2

Timestamp Range 1:
  Delta Largest Acknowledged: 0 // Starting at packet 100
  Timestamp Delta Count: 5
  Timestamps Deltas: 380, 10, 10, 5, 5

Timestamp Range 2:
  Delta Largest Acknowledged: 9 // Starting at packet 91
  Timestamp Delta Count: 5
  Timestamp Deltas: 20, 10, 10, 5, 5
```

After that assume that the receiver receives packets 92 to 95 out-of-order at the following timestamps relative to the basis:

Packet Number	Relative Timestamp
92	390
93	392
94	394
95	395

Table 2

The receiver can send a new ACK frame with all of the timestamps, as follows:

```
Largest Acknowledged: 100
...
Timestamp Ranges Count: 3

Timestamp Range 1:
  Delta Largest Acknowledged: 5 // Starting at packet 95
  Timestamp Delta Count: 4
  Timestamps Deltas: 395, 1, 2, 2

Timestamp Range 2:
  Delta Largest Acknowledged: 0 // Starting at packet 100
  Timestamp Delta Count: 5
  Timestamps Deltas: 10, 10, 10, 5, 5

Timestamp Range 3:
  Delta Largest Acknowledged: 9 // Starting at packet 91
  Timestamp Delta Count: 5
  Timestamp Deltas: 20, 10, 10, 5, 5
```

In this particular scenario, the receiver can also choose to report the first timestamp range only since the timestamps for the other two ranges have already been reported.

8. Security Considerations

TODO Security

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. 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/rfc/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/rfc/rfc8174>.

[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/rfc/rfc9000>.

10.2. Informative References

[I-D.ietf-rmcat-gcc]
Holmer, S., Lundin, H., Carlucci, G., De Cicco, L., and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication", Work in Progress, Internet-Draft, draft-ietf-rmcat-gcc-02, 8 July 2016, <https://datatracker.ietf.org/doc/html/draft-ietf-rmcat-gcc-02>.

[MP-QUIC] Liu, Y., Ma, Y., De Coninck, Q., Bonaventure, O., Huitema, C., and M. K端hlewind, "Multipath Extension for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-multipath-15, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-15>.

[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/rfc/rfc9002>.

[RRBNC] Cottrel, R. V. R. R. B. R. N. J. and L., "pathChirp: Efficient Available Bandwidth Estimation for Network Paths", 2003.

Acknowledgments

The editors would like to thank Ilango Purushothaman and Brandon Schlinker for their contributions to the design of this QUIC extension.

Authors' Addresses

Connor Smith
NVIDIA
Email: connorsmith.ietf@gmail.com

Ian Swett (editor)
Google LLC
Email: ianswett@google.com

Joseph Beshay (editor)
Meta Platforms, Inc.
Email: jbeshay@meta.com

Sharad Jaiswal (editor)
Meta Platforms, Inc.
Email: sj77@meta.com