

QUIC
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2026

I. Swett, Ed.
Google LLC
J. Beshay, Ed.
Meta Platforms, Inc.
16 March 2026

QUIC Extended Acknowledgement for Reporting Packet Receive Timestamps
draft-ietf-quic-receive-ts-02

Abstract

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

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 17 September 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Motivation	2
3. Conventions and Definitions	3
4. New ACK_RECEIVE_TIMESTAMP Frame Wire Format	3
4.1. Timestamp Ranges	4
5. PATH_ACK_RECEIVE_TIMESTAMP Frame Wire Format	5
6. Extension Negotiation	6
6.1. Receive Timestamp Basis	7
7. Discussion	7
7.1. Best-Effort Behavior	7
7.2. Frame Size	8
8. Examples	8
9. Security Considerations	10
10. IANA Considerations	10
11. References	10
11.1. Normative References	10
11.2. Informative References	11
Acknowledgments	11
Authors' Addresses	12

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 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. New ACK_RECEIVE_TIMESTAMP Frame Wire Format

Once the receive timestamps extension is negotiated (see Section 6), an endpoint MAY use the ACK_RECEIVE_TIMESTAMP frame defined below to report receive timestamps to its peer. An endpoint MAY continue to use the existing ACK frames as specified in Section 19.3 of [RFC9000] if it does not have any receive timestamps or does not want to report them.

Endpoints send ACK_RECEIVE_TIMESTAMP frames in 1-RTT packets, with 0 or more receive timestamps following the Ack Ranges and optional ECN Counts. Similar to the ACK frame types (0x02..0x03), the ACK_RECEIVE_TIMESTAMP frame defines two frame types (0x03178307..0x03178308) to indicate whether the frame includes ECN counts. ACK frames are never sent in 0-RTT packets, so the same applies to ACK_RECEIVE_TIMESTAMP frames.

```
ACK_RECEIVE_TIMESTAMP Frame {
  Type (i) = 0x03178307..0x03178308,
  Largest Acknowledged (i),
  ACK Delay (i),
  ACK Range Count (i),
  First ACK Range (i),
  ACK Range (...) ...,
  [ECN Counts (...)],           // included iff Type == 0x03178308
  Receive Timestamps (...) // see {{ts-ranges}}
}
```

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.

Each packet in a range MUST be an acknowledged packet, i.e., the packet number MUST have been included in an ACK Range in the current or a previously sent ACK, ACK_RECEIVE_TIMESTAMP, PATH_ACK, or PATH_ACK_RECEIVE_TIMESTAMP frame.

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 6.

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 6.1), 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 multiplying the value in the field by 2 to the power of the `receive_timestamps_exponent` transport parameter received by the sender of the frame (see Section 6):

When the receiver receives packets out-of-order, it SHOULD report them with other packets in a single `ACK_RECEIVE_TIMESTAMP`s or `PATH_ACK_RECEIVE_TIMESTAMP`s frame, starting with the most recently received packet regardless of the packet number order. See Section 8 for examples of reporting timestamps of out-of-order packets.

5. `PATH_ACK_RECEIVE_TIMESTAMP`s Frame Wire Format

When both the receive timestamps extension and the multipath extension [MULTIPATH] are negotiated, an endpoint MAY use the `PATH_ACK_RECEIVE_TIMESTAMP`s frame defined below to report receive timestamps for packets received on a specific path. An endpoint MAY continue to use the existing `PATH_ACK` frames as specified in Section 4.1 of [MULTIPATH] if it does not have any receive timestamps or does not want to report them.

Endpoints send `PATH_ACK_RECEIVE_TIMESTAMP`s frames in 1-RTT packets, with 0 or more receive timestamps following the Ack Ranges and optional ECN Counts. Similar to the `PATH_ACK` frame types (0x3e..0x3f), the `PATH_ACK_RECEIVE_TIMESTAMP`s frame defines two frame types (0x03178309..0x0317830a) to indicate whether the frame includes ECN counts.

```
PATH_ACK_RECEIVE_TIMESTAMP Frame {
  Type (i) = 0x03178309..0x0317830a,
  Path Identifier (i),
  Largest Acknowledged (i),
  ACK Delay (i),
  ACK Range Count (i),
  First ACK Range (i),
  ACK Range (...) ...,
  [ECN Counts (...)],          // included iff Type == 0x0317830a
  Receive Timestamps (...)    // see {{ts-ranges}}
}
```

Figure 4: `PATH_ACK_RECEIVE_TIMESTAMP`s Frame Format

Compared to the `ACK_RECEIVE_TIMESTAMP`s frame defined in Section 4, the following field is added:

Path Identifier: The path ID associated with the packet number space of the 1-RTT packets which are acknowledged by this frame, as specified in Section 4.1 of [MULTIPATH].

All other fields are the same as for the `ACK_RECEIVE_TIMESTAMP`s frame (Section 4). The Receive Timestamps field follows the same format described in Section 4.1.

When truncation is necessary to fit within the `max_receive_timestamps_per_ack` limit or reduce the size of the frame, the receiver SHOULD retain timestamps for the most recently received packets and omit timestamps for older packets.

6. Extension Negotiation

`max_receive_timestamps_per_ack` (0x4ac07 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_RECEIVE_TIMESTAMP`s or `PATH_ACK_RECEIVE_TIMESTAMP`s frame.

Each `ACK_RECEIVE_TIMESTAMP`s or `PATH_ACK_RECEIVE_TIMESTAMP`s frame sent MUST NOT contain more than the peer's maximum number of receive timestamps.

`receive_timestamps_exponent` (0x4ac26 temporary value for draft use): A variable-length integer indicating the exponent to be used when encoding and decoding timestamp delta fields in `ACK_RECEIVE_TIMESTAMP`s and `PATH_ACK_RECEIVE_TIMESTAMP`s 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. If the `receive_timestamps_exponent` transport parameter is present and `max_receive_timestamps_per_ack` is not, receive timestamps are not supported and `receive_timestamps_exponent` MUST be ignored.

6.1. 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.

7. Discussion

7.1. 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_RECEIVE_TIMESTAMP`s or `PATH_ACK_RECEIVE_TIMESTAMP`s frame is lost.
- * The receiver truncates the number of timestamps sent in order to (a) avoid sending more than `max_receive_timestamps_per_ack` (Section 6); or (b) fit the `ACK_RECEIVE_TIMESTAMP`s or `PATH_ACK_RECEIVE_TIMESTAMP`s frame into a packet.
- * The receiver is unable to measure the arrival timestamp of a packet with sufficient accuracy, for example due to a scheduling delay in a userspace implementation, and omits the packet from the Timestamp Ranges while still acknowledging it in the ACK Ranges.

7.2. Frame Size

The addition of receive timestamps increases the size of ACK frames. Receivers SHOULD use receive timestamps to fill available space in packets that would already be sent, rather than sending additional packets solely to report timestamps. In such cases, the receiver would send fewer timestamps than the maximum allowed by `max_receive_timestamps_per_ack`.

8. 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.

9. Security Considerations

TODO Security

10. IANA Considerations

This document uses temporary values for the transport parameters `max_receive_timestamps_per_ack` (0x4ac07) and `receive_timestamps_exponent` (0x4ac26), and for the frame types `ACK_RECEIVE_TIMESTAMP` (0x03178307..0x03178308) and `PATH_ACK_RECEIVE_TIMESTAMP` (0x03178309..0x0317830a). Prior to publication, these will be replaced with permanent registrations in the "QUIC Transport Parameters" (Section 22.3 of [RFC9000]) and "QUIC Frame Types" (Section 22.4 of [RFC9000]) registries.

11. References

11.1. Normative References

[MULTIPATH]

Liu, Y., Ma, Y., De Coninck, Q., Bonaventure, O., Huitema, C., and M. K端hlewind, "Managing multiple paths for a QUIC connection", Work in Progress, Internet-Draft, draft-ietf-quic-multipath-20, 20 February 2026, <<https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-20>>.

[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>>.

11.2. 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>>.

[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 Connor Smith for writing the initial draft. The editors would also like to thank Sharad Jaiswal, Ilango Purushothaman, and Brandon Schlinker for their contributions to the design of this QUIC extension.

Authors' Addresses

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

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