QUIC
Internet-Draft
Intended status: Informational
Expires: 28 April 2022

C. Smith
Magic Leap, Inc.
I. Swett
Google LLC

25 October 2021

QUIC Extension for Reporting Packet Receive Timestamps
draft-smith-quic-receive-ts-00

Abstract

This document defines an extension to the QUIC transport protocol which supports reporting multiple packet receive timestamps using a new ACK_RECEIVE_TIMESTAMPS frame type.

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

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 28 April 2022.

Copyright Notice

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 using a new ACK_RECEIVE_TIMESTAMPS frame type.

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.

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. Extension Negotiation

The use of the ACK_RECEIVE_TIMESTAMPS frame is negotiated using the following two transport parameters (Section 7.2 of [RFC9000]):

max_receive_timestamps_per_ack (TBD): A variable-length integer indicating that the sending endpoint would like to receive ACK_RECEIVE_TIMESTAMPS frames from the peer containing no more than the given maximum number of receive timestamps.

Upon receiving this parameter with a non-zero value, the peer SHOULD send ACK_RECEIVE_TIMESTAMPS frames instead of ACK frames if new receive timestamp information is available. The peer MAY still send regular ACK frames (e.g. if no timing information is

available), in which case the endpoint MUST still support processing regular ACK frames as defined by Section 19.3 of [RFC9000].

Each ACK_RECEIVE_TIMESTAMPS frame sent MUST NOT contain more than the specified maximum number of receive timestamps, but MAY contain fewer or none.

receive_timestamps_exponent (TBD): A variable-length integer

indicating the exponent to be used when encoding and decoding timestamp delta fields in ACK_RECEIVE_TIMESTAMPS frames sent by the peer (see Section 5.1). If this value is absent, a default value of 0 is assumed (indicating microsecond precision). Values above 20 are invalid.

4.1. Receive Timestamp Basis

Endpoints which send ACK_RECEIVE_TIMESTAMPS frames must determine a value, receive_timestamp_basis, relative to which all receive timestamps for the session will be reported (see Section 5.1).

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.

TODO: Discuss (here or below) why receive timestamps are reported relative to the basis, rather than in absolute time to avoid clock synchronization between endpoints.

5. ACK_RECEIVE_TIMESTAMPS Frame

Receivers send ACK_RECEIVE_TIMESTAMPS (type=TBD) frames in place of--and in the same manner as--regular ACK frames as described in Section 13.2 of [RFC9000]. However, ACK_RECEIVE_TIMESTAMPS frames contain additional fields to report packet receive timestamps.

ACK_RECEIVE_TIMESTAMPS frames are formatted as shown in Figure 1.

```
ACK_RECEIVE_TIMESTAMPS Frame {
  Type (i) = TBD
  // Fields of the existing ACK (type=0x02) frame:
  Largest Acknowledged (i),
  ACK Delay (i),
  ACK Range Count (i),
  First ACK Range (i),
  ACK Range (..) ...,
  // Additional fields for ACK_RECEIVE_TIMESTAMPS:
  Timestamp Range Count (i),
  Timestamp Ranges (..) ...,
}
```

Figure 1: ACK_RECEIVE_TIMESTAMPS Frame Format

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

ACK_RECEIVE_TIMESTAMPS frames contain the following additional 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 5.1.

5.1. 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 Gap 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).

Note that reporting receive timestamps for packets received out of order is not supported. Specifically: for any packet number P for which a receive timestamp T is reported, all reports for packet numbers less than P must have timestamps less than or equal to T; and

all reports for packet numbers greater than P must have timestamps greater than or equal to T.

Timestamp Ranges are structured as shown in Figure 2.

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

Figure 2: Timestamp Range Format

The fields that form each Timestamp Range are:

Gap: A variable-length integer indicating the largest packet number in the Timestamp Range as follows:

For the first Timestamp Range: Gap is the difference between (a) the Largest Acknowledged packet number in the frame and (b) the largest packet in the current (first) Timestamp Range.

For subsequent Timestamp Ranges: Gap is the difference between (a) the packet number two lower than the smallest packet number in the previous Timestamp Range and (b) the largest packet in the current Timestamp Range.

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 4.

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 4.1), 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 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_RECEIVE_TIMESTAMPS frame (see Section 4):

6. Discussion

6.1. Best-Effort Behavior

Receive timestamps are sent on a best-effort basis and endpoints MUST gracefully handle scenarios where receive timestamp information for sent packets is not received. Examples of such scenarios are:

* The packet containing the ACK_RECEIVE_TIMESTAMP 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 4); or (b) fit the ACK frame into a packet.

7. Security Considerations

TODO Security

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. 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>.

9.2. Informative References

[I-D.ietf-rmcat-gcc] Holmer, S., Lundin, H., Carlucci, G., Cicco, L. D., 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.a.L., "pathChirp: Efficient Available Bandwidth Estimation for Network Paths", 2003.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Connor Smith
Magic Leap, Inc.

Email: connorsmith.ietf@gmail.com

Ian Swett
Google LLC

Email: ianswett@google.com