Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 1, 2020

C. Huitema
Private Octopus Inc.
February 29, 2020

Quic Timestamps For Measuring One-Way Delays
draft-huitema-quic-ts-02

Abstract

The TIME_STAMP frame can be added to Quic packets when one way delay measurements is useful. The timestamp is set to the number of microseconds from the beginning of the connection to the time at which the packet is sent. The draft defines the "enable_time_stamp" transport parameter for negotiating the use of this extension frame, and a new frame types for the time_stamped frame.

Table of Contents

1. Measuring One-Way Delays

The QUIC Transport Protocol [I-D.ietf-quic-transport] provides a secure, multiplexed connection for transmitting reliable streams of application data. The algorithms for QUIC Loss Detection and Congestion Control [I-D.ietf-quic-recovery] use measurement of Round Trip Time (RTT) to determine when packets should be retransmitted. RTT measurements are useful, but there are however many cases in which more precise One-Way Delay (1WD) measurements enable more efficient Loss Detection and Congestion Control.

An example would be the Low Extra Delay Background Transport (LEDBAT) [RFC6817] which uses variations in transmission delay to detect competition for transmission resource. Experience shows that while LEDBAT may be implemented using RTT measurements, it is somewhat inefficient because it will cause unnecessary slowdowns in case of queues or delayed ACKs on the return path. Using 1WD solves these issues. Similar argument can be made for most delay-based algorithms.

We propose to enable one way delay measurements in QUIC by defining a TIME_STAMP frame carrying the time at which a packet is sent. The use of this extension frame is negotiated with a transport parameter, "enable_time_stamp". When the extension is negotiated by both

parties, this frame can be used in conjunction with other such as ACK to measure one way delays.

1.1. Terms and Definitions

The keywords "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. Specification

The enable_time_stamp transport parameter used for negotiating the use of the extension frame is defined in Section 2.1. The time_stamp frame format is defined in Section 2.3.

2.1. Negotiation

The use of the time_stamp frame extension is negotiated using a transport parameter:

- enable_time_stamp (TBD)

The enable time stamp transport parameter is included if the endpoint accepts and sends time_stamp frames for this connection. This parameter has a zero-length value. Negotiation is successful if both peers support include this parameter in their transport parameter message. Peers that receive a time_stamp frame in the absence of successful negotiation MAY terminate the connection with a PROTOCOL VIOLATION error.

2.2. Sending TIME_STAMP frames

If negotiation is successful the peers SHOULD add a time_stamp frame to 1RTT packets carrying an ACK frame. This specification does not impose a placement of TIME_STAMP frames in the packet. They MAY be sent either before or after the ACK frame.

Implementations SHOULD NOT send more than one TIME_STAMP frame per packet, but they MAY send more than one in rare circumstances. When multiple TIME_STAMP frames are present in a packet, the receiver retains the frame indicating the largest timestamp.

Implementations MUST NOT send the TIME_STAMP frame in Initial, 0-RTT or Handshake packets, because there is a risk that the peer will receive such packets before the negotiation completes. This restriction may appear excessive because some Handshake packets are

typically sent after the negotiation completes, but restricting TIME_STAMP frames to 1RTT packets is simpler and less error prone than allowing the TIME_STAMP frame in just a fraction of Handshake packets.

2.3. TIME_STAMP frame format

TIME_STAMP frames are identified by the frame type:

- o TIME_STAMP (TBD)

TIME_STAMP frames carry a single parameter, the time stamp.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Time Stamp (i)                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: TIME_STAMP Frame Format with Time Stamp

The time stamp encodes the number of microseconds since the beginning of the connection, as measured by the peer at the time at which the packet is sent. It is encoded using the exponent selected by the peer in the ack_delay_exponent. The exponent reduced time stamp is encoded as a variable length integer.

The beginning of the connection is defined as follow:

o for the client, the time when the first Initial packet is sent;

o for the server, the time when the first Initial packet is received.

TIME_STAMP frames are not ack-eliciting. Their loss does not require retransmission.

2.4. RTT Measurements

RTT measurements are performed as specified in Section 4 of [I-D.ietf-quic-recovery], without reference to the Timestamp parameter of the Timestamped ACK frames.

2.5. One-Way Delay Measurements

An endpoint generates a One Way Delay Sample on receiving a packet containing both a TIME_STAMP frame and an ACK frame that meets the following two conditions:

o the largest acknowledged packet number is newly acknowledged, and

o at least one of the newly acknowledged packets was ack-eliciting.

The One Way Delay sample, latest_1wd, is generated as the time elapsed since the largest acknowledged packet was sent, corrected for the difference between local time at the sending peer and connection time at the receiving peer, phase_shift.

```
latest_1wd = time_stamp - send_time_of_largest_acked - phase_shift
```

By convention, the phase_shift is estimated upon reception of the first RTT sample, first_rtt. It is set to:

```
phase_shift = time_stamp - send_time_of_largest_acked - latest_rtt/2
```

In that formula, we assume that the local time are measured in microseconds since the beginning of the connection.

We understand that clocks may drift over time, and that simply estimating a phase shift at the beginning of a connection may be too simplistic for long duration connections. Implementations MAY adopt different strategies to reestimate the phase shift at appropriate intervals. Specifying these strategies is beyond the scope of this document.

3. Discussion

This document replaces an earlier proposal to modify the format of the ACK frame by including a time stamp inside the modified frame. The revised proposal encodes the time stamp independently of the ACK frame, which requires slightly more overhead to encode the type of the TIME_STAMP frame.

Defining an independent frame allows for more flexibility. This draft defines the combination of TIME_STAMP with ACK frames, but they could be combined with other frames as well. For example, adding a TIME_STAMP to packets carrying a Path Response could allow measuring one way delays before deciding to migrate to a new path.

3.1. Management of Time

There are two known issues with deducing one way delays from RTT measurements: clock drift and undefined phase difference.

The phase difference problem is easy to understand. We start from a list of measurements associating the send time of packet number x (s[x]), the receive time of the acknowledgement of packet (a[x]), and

the time stamp indicating when packet x was received by the peer (p[x]). The peer's time stamp are expressed in the peer's clock.

Suppose that we model the peer's clock as local time plus phase difference f, and that we model the rtt as the sum of two one way delays, up (u[x]) and down (d[x]). We get:

$$u[x] = p[x] + f - s[x]$$

$$d[x] = a[x] - p[x] - f$$

Just looking at the equation shows that the value of f cannot be determined from the a series of measurement (s[x], a[x], p[x]). You can just add constraints that all u[x] and d[x] are positive numbers, which gives a range of plausible values for f: $\max(s[x] - p[x]) < f <$

min(a[x]-p[x]). In case you wonder, you get similar formulations in a multipath scenario. The plausible range may narrow to the min rtt of the shortest path, but no further.

The phase difference uncertainty is not a big issue in practice, because control algorithms are much more interested in the variations of the delays than by their absolute values. Suppose we want to compare one way delays at measurement (x) and (y). We get:

$$u[x] = p[x] + f - s[x]$$

$$u[y] = p[y] + f - s[y]$$

$$u[x] - u[y] = p[x] - p[y] - s[x] + s[y]$$

The phase difference does not affect the measurement of variations in the one way delay.

The clock drift is another matter. All the equations above assume that the local clock and the remote clock have the same frequency. This is an approximation. Clocks drift over time. Instead of just considering a stable phase difference, one should consider the sum of a phase difference and a time-varying drift component. Estimating drift is a complex problem. This was studied in detail in the development of the Network Time Protocol (NTP) [RFC5905]. In theory, implementations of Quic could copy the algorithms of NTP to build a model of the clocks used by the local node and the peer. That would be very complex.

Fortunately, implementations of Quic no not need to implement something as complex as NTP. Most time based algorithms are only interested in variations of delays over a short horizon. Clock drift happens at a slow pace, maybe 1 millisecond per minute. Time base

congestion control algorithms already have to cope with the potential drift of the minimum RTT due to changing network conditions. They do that by periodically restarting themeasurement of the minimum RTT after some delay, typically less than a minute. A simple implementation of one way delay measurements could follow the same approach, for example resetting the phase difference every 30 seconds or so.

4. Security Considerations

The Timestamp value in the TIME_STAMP frame is asserted by the sender of the packet. Adversarial peers could chose values of the time stamp designed to exercise side effects in congestion control algorithms or other algorithms relying on the one-way delays. This can be mitigated by running plausibility checks on the received values. For example, each peer can maintain statistics not just on the One Way Delays, but also on the differences between One Way Delays and RTT, and detect outlier values. Peers can also compare the differences between timestamps in packets carrying acknowledgements and the differences between the sending times of corresponding packets, and detect anomalies if the delays between acknowledging packets appears shorter than the delays when sending them.

5. IANA Considerations

This document registers a new value in the QUIC Transport Parameter Registry:

Value: TBD (using value 0x7157 in early deployments)

Parameter Name: enable_time_stamp

Specification: Indicates that the connection should use TimeStamped ACK frames

This document also registers a new value in the QUIC Frame Type registry:

Value: TBD (using value 757 in early deployments)

Frame Name: TIME_STAMP

Specification: Time stamp set at the time packet was sent

6. Acknowledgements

Thanks to Dmitri Tikhonov, Tal Misrahi and Watson Ladd for their reviews and suggestions.

7. References

7.1. Normative References

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-26 (work in progress), February 2020.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-27 (work in progress), February 2020.

[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>.

7.2. Informative References

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <https://www.rfc-editor.org/info/rfc6817>.

Author's Address

Christian Huitema
Private Octopus Inc.
427 Golfcourse Rd
Friday Harbor WA 98250
U.S.A

Email: huitema@huitema.net