QUIC Packet Receive Timestamps via Extended Ack

draft-smith-quic-receive-ts

QUIC @ IETF 122

Presented by **Joseph Beshay**, Meta
Connor Smith, NVIDIA, Ian Swett, Google, Sharad Jaiswal, Meta, Ilango Purushothaman, Meta, Brandon Schlinker, Meta

Previously presented: Why receive timestamps?

- Better bandwidth estimation, especially for real-time congestion control algorithms.
  - Originally motivated by SQP
  - More important when acknowledging more packets in an ACK (ie: Ack Frequency)
  - Used in Google Congestion Control-derived algorithm for streaming media.
- Last-mile congestion detection
  - Variations between transmission and delivery gaps can be used to infer physical medium congestion.

draft-smith-quic-receive-ts

- Initially presented at IETF 112 (recording)
- Brought back with minor modifications at IETF 118 (recording)
  - Proposed two new fields to allow reporting out-of-order timestamps (Discussed later)
- Bringing it back again at IETF 122 with an extensible ACK frame that can carry ECN and potentially other optional sets of fields.

Overview of Draft

Enable the peer to report packet receive timestamps for some or all ACKed packets. The timestamp is connection local and not intended to be absolute.

New fields for the ACK frame:

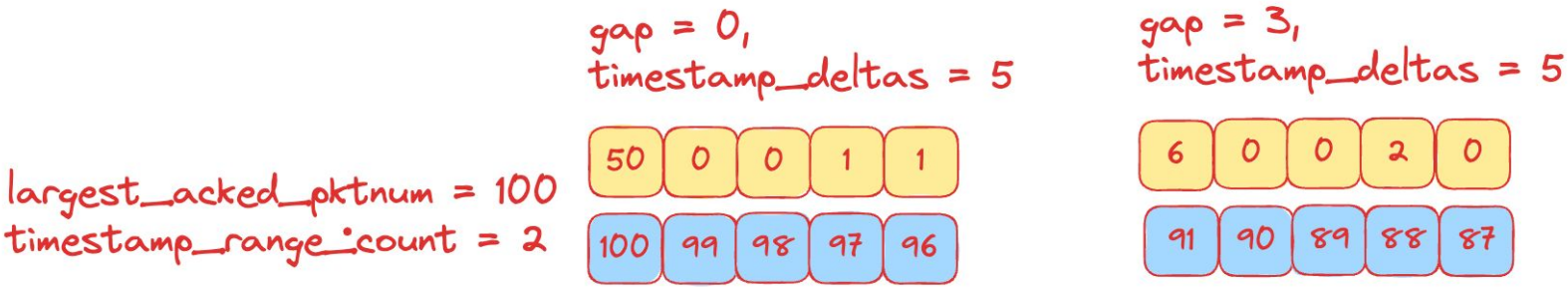
```
Timestamp Range Count (i)
Timestamp Ranges (..)

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

Timestamp Ranges Encoding

Each Timestamp Range describes a series of contiguous packet receive timestamps in descending sequential packet number (and descending timestamp) order.

Gap	A variable-length integer indicating the largest packet number in the Timestamp Range (as the difference between the largest acked packet, or the smallest packet of the previous range)
Timestamp Delta Count	A variable-length integer indicating the number of Timestamp Deltas in the current Timestamp Range.
Timestamp Deltas	Variable-length integers encoding the receive timestamp for contiguous packets in the Timestamp Range in descending packet number



Extending the ACK frame

The previous draft defined a single ACK_RECEIVE_TIMESTAMPS that carried ACK fields + new timestamp fields.

But receive timestamps are not the only set of optional fields. We also need ECN counts.

```
ACK EXTENDED Frame {
   Type (i) = 0xB1 (temporary)

   // Fields of the existing ACK (type=0x02) frame:
   Largest Acknowledged (i),
   ACK Delay (i),
   ACK Range Count (i),
   First ACK Range (i),
   ACK Range (..) ...,

   // New fields
   Extended Ack Features (i),
   // Optional ECN counts (if bit 0 is set in Features)
   [ECN Counts (..)],
   // Optional Receive Timestamps (if bit 1 is set in Features)
   [Receive Timestamps (..)]
 }
```

Why an extensible ACK frame?

Extensions to the ACK frame currently need to introduce two different variants one with and another without ECN.

If there are more extensions or extension versions to ACKs, the number of frames needed to support the different valid combinations grows exponentially.

An extensible ACK frame with multiple sets of optional fields allows combining the different extensions without adding more frames.

ACKs are one of the most frequently sent frames, so a 1 byte codepoint is key

Issue #2: Should timestamps have to be in order?

The proposal from IETF 118 suggests a modification to the frame format to support that.

Issue #7: Extended ACK frame with multiple optional fields vs. new ACK frame types vs Always send timestamps?

Would the ExtendedAckFeatures field require an IANA registry?

If yes, does avoiding the potential of increasing frame types justify the IANA overhead?

Do we expect to see more extensions that add ACK fields?

Is it OK to have two different ways to send ACKs with ECN? ACK_ECN and ACK_EXTENDED?

Potential options:

1. Maintain the ACK_EXTENDED frame approach that can support this and future extensions.
2. Use two new frame types ACK_RECEIVE_TIMESTAMPS and ACK_ECN_RECEIVE_TIMESTAMPS which correspond to the existing ACK and ACK_ECN frames with additional receive timestamp fields.
3. Use one new frame type ACK_RECEIVE_TIMESTAMPS frame that can optionally carry the ECN counts and/or receive timestamps fields, without being extensible to new features, to avoid the IANA overhead.
4. Require all ACK and ACK_ECN frames to have a timestamp section if it's negotiated. Consumes an extra byte if no timestamps are present, but doesn't use any new frame types. PR#8

Next steps

Does the WG think Extended ACK should be pursued?

Mvfst has an implementation of the draft. No congestion control but can log timestamps in qlog. Any interest in interop testing?

Is there enough interest to adopt this draft and resume the discussions?

Backup Slides

Previous discussion: What about composability?

This could be included in its own frame rather than the ACK frame to make it usable for other purposes, such as path challenge.

For current use-cases, timestamps are useful when processing ACKs. Keeping them in one frame significantly simplifies the implementation where the timestamp information can be checked at the same time as the ACK arrival.

- ECN was put into the same frame for this reason as well