History

gQUIC had receive timestamps in its ACK frame

Only used by GoogCC and other latency sensitive CC’s

More Recently....

draft-huitema-quic-ts specifies a separate frame with a 
Timestamp of when the packet is sent.
Use Cases

Existing algorithms use receive timestamps to improve transport performance.

Examples include:

- WebRTC congestion control [I-D.ietf-rmcat-gcc] uses inter-departure and inter-arrival times.
- pathChirp ([RRBNC]) technique estimates available bandwidth by measuring inter-arrival time of multiple packets.
Updated the gQUIC format to varints

ACK_RECEIVE_TIMESAMPS Frame {
    Type (i) = TBD
    // Fields of the ACK (type=0x02) frame.
    ...
    Timestamp Range Count (i),
    Timestamp Ranges (..) {
        Gap (i),
        Timestamp Delta Count (i),
        Timestamp Delta (i) ....,
    }
}
Transport Parameters

max_receive_timestamps_per_ack:

Indicates the sender would like to receive no more this many receive timestamps in a ACK_RECEIVE_TIMESTAMPS frame.

receive_timestamps_exponent:

The exponent to be used when encoding and decoding timestamp delta fields.
Next Steps

Is this worth pursuing?

If so, should it be separate?

It’s subtly different from draft-huitema-quic-ts

What is needed before adoption?