**Current Frame Format**

**ACK_FREQUENCY Frame** {
  Type (i) = 0xaf,
  Sequence Number (i),
  Ack-Eliciting Threshold (i),
  Request Max Ack Delay (i),
  Reserved (6),
  Ignore CE (1),
  Ignore Order (1)
}

**Sequence Number**: Allows receivers to ignore obsolete frames after reordering.

**Ack-Eliciting Threshold**: The maximum number of ack-eliciting packets the recipient of this frame can receive before sending an acknowledgment.

**Request Max Ack Delay**: The value to which the endpoint requests the peer update its max_ack_delay

**Ignore CE**: This field is set to true by an endpoint that does not wish to receive an immediate acknowledgement when the peer receives CE-marked packets.

**Ignore Order**: This field is set to true by an endpoint that does not wish to receive an immediate acknowledgement when the peer receives a packet out of order.
Latency to detect packet loss? (#96)

**Issue:** One ACK is sent immediately, like QUIC v1. But after that, the next ACK will not be sent until the Ack-Eliciting Threshold or Ack Delay are hit.

Loss detection delayed when Ack-Eliciting Threshold is larger than the Packet Threshold.

Importantly, loss detection latency is worse than QUIC v1.
Latency to detect packet loss? (#96)

Proposal (#100):

Communicate Reordering Threshold to receiver instead of Ignore Order

Receiver immediately ACKs when missing packets in:

\[
[\text{largest\_acknowledged\_sent} - \text{Reordering Threshold}, \text{largest\_acknowledged} - \text{Reordering Threshold}]
\]

Result: Receiver reduces ACKs when packets received out of order while improving loss detection latency over QUIC v1
What next

Several deployments have shown perf improvement

WGLC?