# Ack Frequency

draft-ietf-quic-ack-frequency

https://github.com/quicwg/ack-frequency

QUIC WG, !San Francisco, July 2021

# Issues editors believe are resolved

To be merged and closed soon:

- Rename `'Packet Tolerance'` to `'ACK-Eliciting Threshold'` #55
- Change `'Update Max Ack Delay'` to `'Request Max Ack Delay'` #56
- Set limits to `min_ack_delay` and `max_ack_delay` #57

# FYI: Tweaking ACK-Eliciting Threshold's meaning

## Issue #49 Off-by-one in packet tolerance (PR#58)

Adjusts the meaning of this field by 1 and changes 0 from an invalid value to meaning ACK every packet.

**NOTE: This changes interop, so the PR changes the `min_ack_delay` Transport Parameter codepoint**

# FYI: Cap ACK-Eliciting Threshold if it's too large

**Issue #45 Maximum Values for Frame Fields (PR#59)**

If a receiver receives an ACK-Eliciting Threshold larger than what it wants to store(ie: it uses a uint8), what should it do?

**Proposed Resolution:**

MUST cap the received value to the largest supported value.

# Where can ACK_FREQUENCY appear? (#22)

- Should min_ack_delay be remembered for 0-RTT? (#40)

Key Question: Can it appear in 0-RTT?

=> If so, must remember min_ack_delay

Otherwise you could accidentally send an invalid ACK_FREQUENCY frame.

# Where can ACK_FREQUENCY appear? (#22)

Pros of allowing ACK_FREQUENCY in 0-RTT:

- Why forbid it?
- Reduction of ACKs for large client to server 0-RTT flights

Pros of forbidding ACK_FREQUENCY in 0-RTT:

- `max_ack_delay` isn't remembered, so more consistent
- `min_ack_delay` may vary by server platform/OS,
  Causing 0-RTT rejections

**Recommendation:** Make ACK_FREQUENCY 1-RTT only

# Eliciting an immediate ACK (#34)

**Problem:** `'Ignore_order'= true`

=> skipping packet numbers no longer elicits an ACK

**Solutions**

1. Use an unused bit in the header
2. Use a 1 byte frame
3. Use a 1 byte frame and offer a STREAM frame codepoint that elicits an immediate ACK

# Eliciting an immediate ACK (#34)

1) **Use an unused bit in the header**

Pros:

- Re-packetization is easy, since no extra frames are necessary for retransmission of a packet's payload
- No byte overhead

Cons:

- Not many header bits left

# Eliciting an immediate ACK (#34)

## 2) Use a 1 byte frame

Pros:

- Simple to implement and understand
- Lots more 1 byte frame types than header bits

Cons:

- Retransmitting previously sent payloads may not fit into a single packet.  ie: PTO-ing a full packet of data

# Eliciting an immediate ACK (#34)

**3) Use a 1 byte frame AND offer a STREAM frame codepoint that elicits an immediate ACK**

Pros:

- *Fairly* simple to implement and understand
- Lots more 1 byte frame types than header bits

Cons:

- Retransmitting data fits into a single packet.
- More complex than a 1 byte frame

# Eliciting an immediate ACK (#34)

## Solutions

1. Use an unused bit in the header
2. Use a 1 byte frame
3. Use a 1 byte frame and offer a STREAM frame codepoint that elicits an immediate ACK

**Proposal:** Add a 1 byte frame now and evaluate whether we need the STREAM frame codepoint

# Replace Ignore Order with Packet Threshold (#35)

**Idea:** Send the local Packet Threshold used for loss detection to the peer, so it avoids sending immediate ACKs earlier than when packets can be declared lost. (ie: twiddles)

# Replace Ignore Order with Packet Threshold (#35)

Pros:

- Reduces unnecessary ACK-only packets in some cases.
- Causes immediate ACKs that enable loss detection earlier in some cases.

Cons:

- More complex than Ignore Order, which may be prone to implementation errors.

**Recommendation:** No changes until this has proven value

# Next Steps

Merge outstanding PRs and close most issues

=> Ship a -01 draft

Request: If you haven't already implemented 00, wait for 01, because the inflight changes are breaking.