Changing Ack Frequency

draft-iyengar-quic-delayed-ack
https://janaiyengar.github.io/ack-frequency

QUIC Interim, Zurich, February 2020

Background

RFC 5681 recommends ACK every 2 packets for TCP

In practice, ACK collapsing (thinning) is widespread for TCP
- at endhosts
- by middleboxes

These optimizations are critical for
- high bandwidth links
- highly asymmetric links (satellite)

Background

QUIC transport currently recommends the same as TCP

Sending QUIC ACKs is noticeably expensive (CPU)

- preference to send fewer ACKs in general
  (see Issue 3304, Issue 1978 for discussion)

Background

QUIC transport currently recommends the same as TCP

Sending QUIC ACKs is noticeably expensive (CPU)
- preference to send fewer ACKs in general
- (see Issue 3304, Issue 1978 for discussion)

ACK rates however have implications for congestion control
- window-based CC (Reno, Cubic) is driven by ACK events
- delaying ACKs decreases throughput of these controllers
- sender capable of fewer ACKs needs to indicate tolerance

Background

QUIC transport currently recommends the same as TCP

Sending QUIC ACKs is noticeably expensive (CPU)
- preference to send fewer ACKs in general
- (see Issue 3304, Issue 1978 for discussion)

ACK rates however have implications for congestion control
- window-based CC (Reno, Cubic) is driven by ACK events
- delaying ACKs decreases throughput of these controllers
- sender capable of fewer ACKs needs to indicate tolerance

Sender may want to *increase* ACKs for future things
- new startup schemes (eg, paced chirping)

Proposal

Sender: Sender of ack-eliciting packets

Receiver: Sender of ACK-only frames in response

Assumption:

- Receiver is naturally incented to ACK minimally
- Sender is naturally incented to process fewer ACKs
- Sender knows its controller's tolerance / desire

Proposal

Sender: Sender of ack-eliciting packets

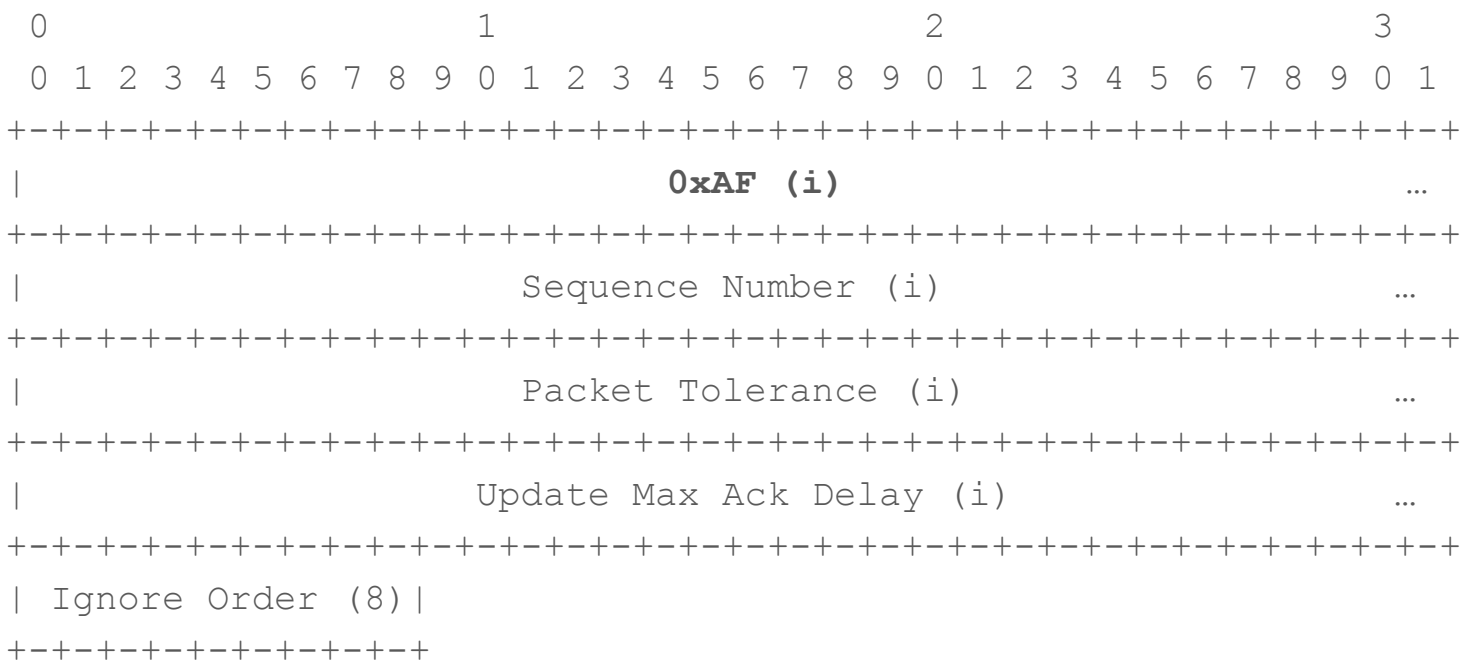
Receiver: Sender of ACK-only frames in response

Assumption:

- Receiver is naturally incented to ACK minimally
- Sender is naturally incented to process fewer ACKs
- Sender knows its controller's tolerance / desire

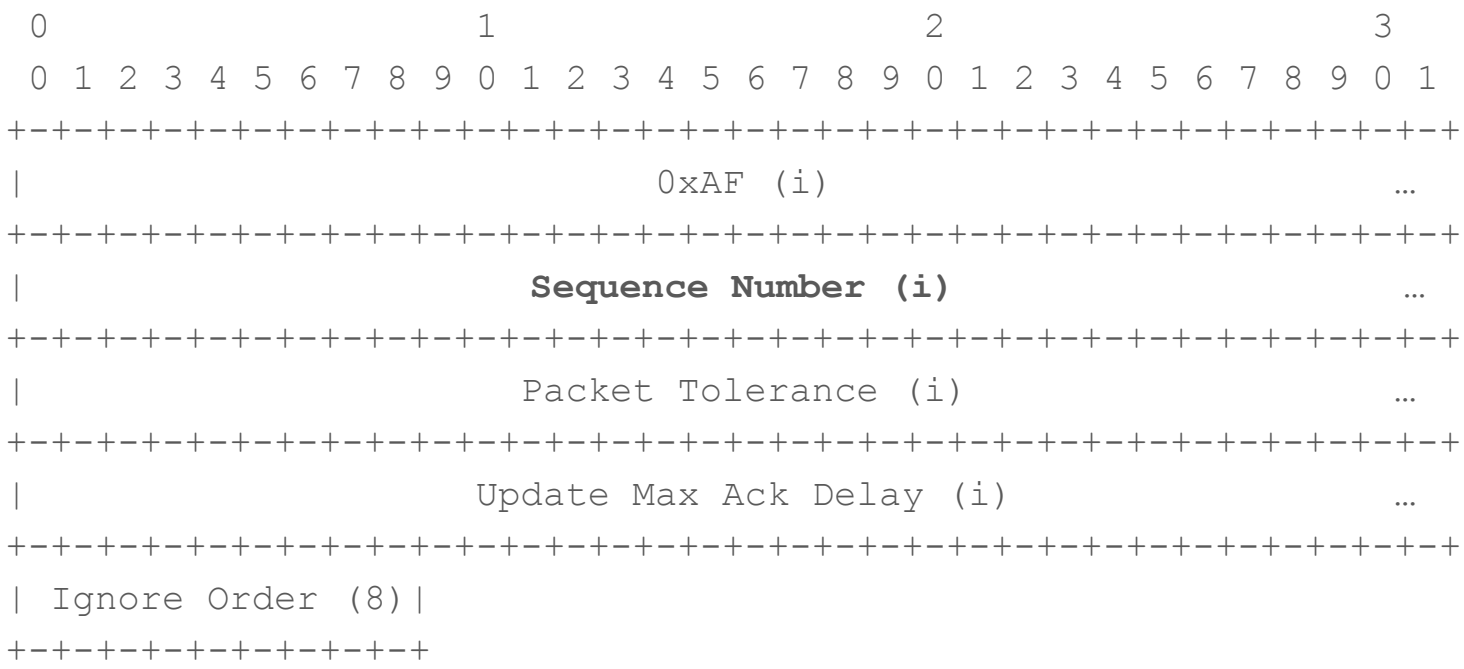
Design: Frame sent from **Sender** to **Receiver** to change receiver's ACK behavior

ACK_FREQUENCY frame



0xAF : Frame Type

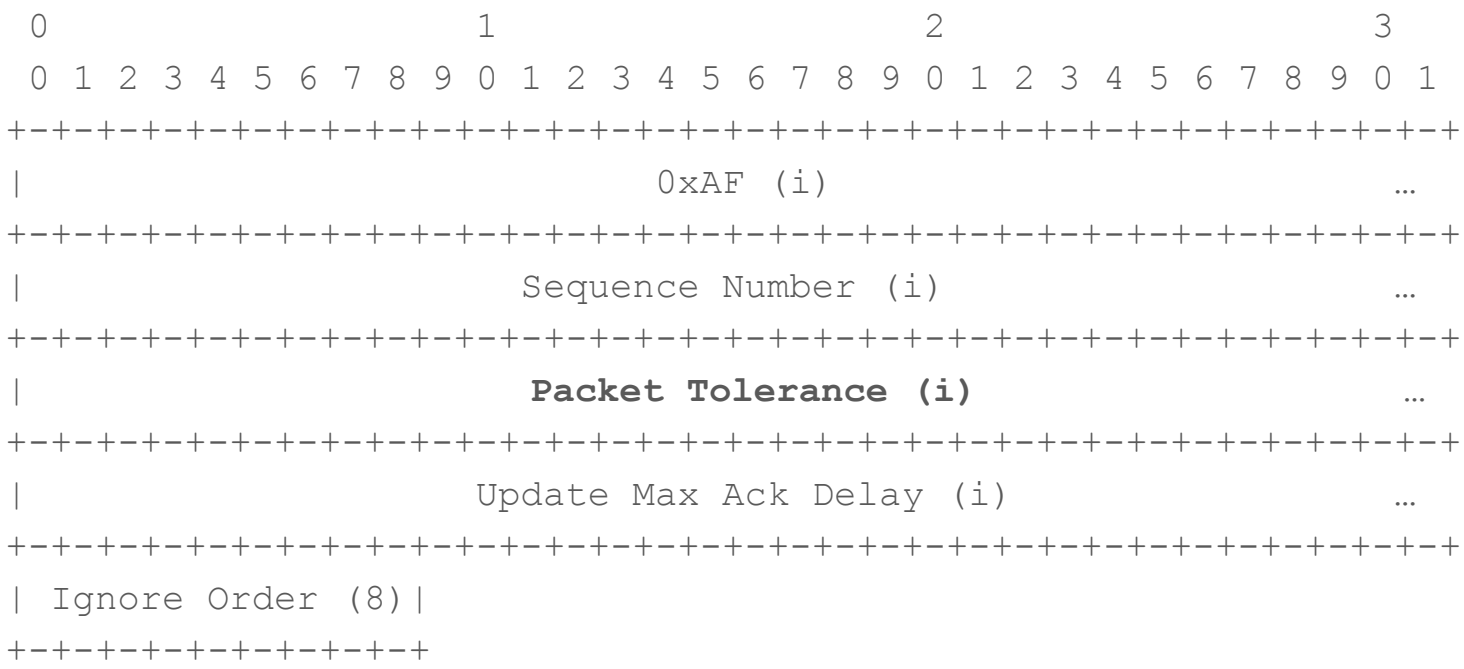
ACK_FREQUENCY frame



Sequence Number:

Ensures consistent processing order

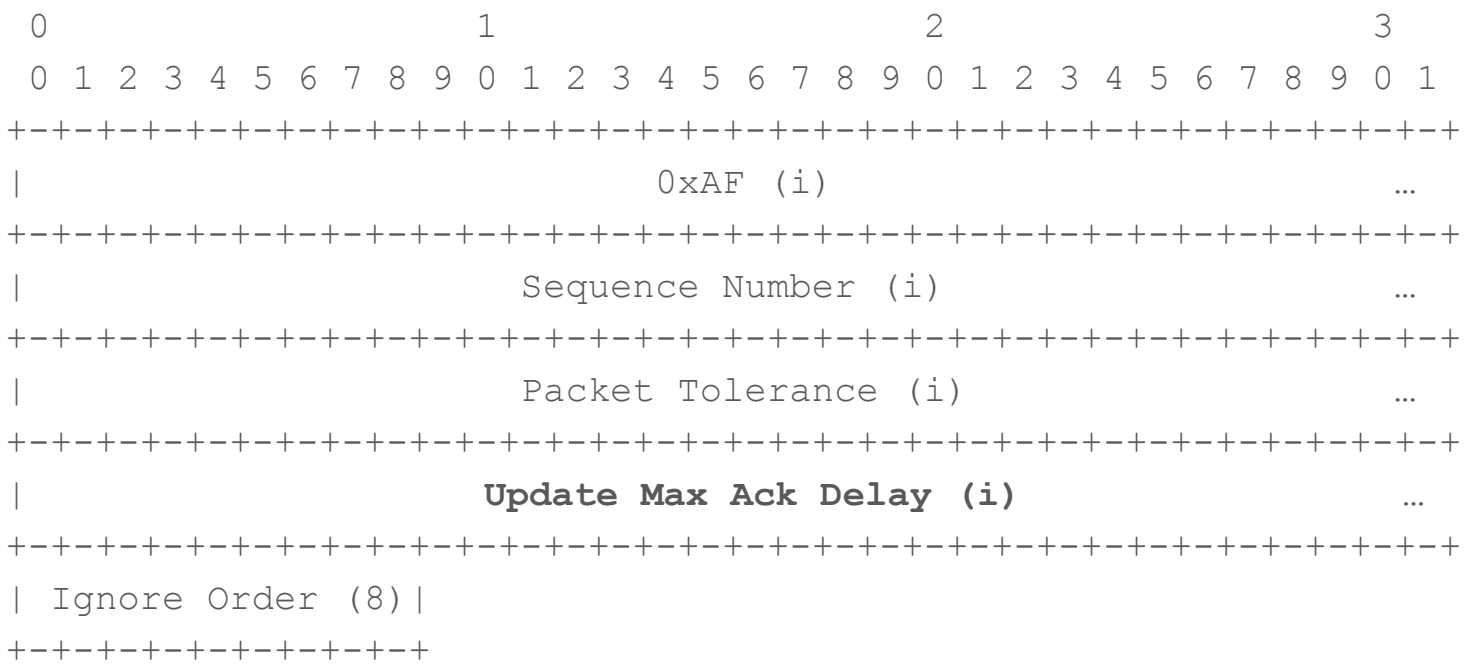
ACK_FREQUENCY frame



Packet Tolerance:

- Number of ack-eliciting packets before an immediate ACK
- Changes default of 2 to be a peer-controlled variable

ACK_FREQUENCY frame



Update Max Ack Delay:

- Updates receiver's max_ack_delay in microseconds
- Changes max_ack_delay to be a peer-controlled variable

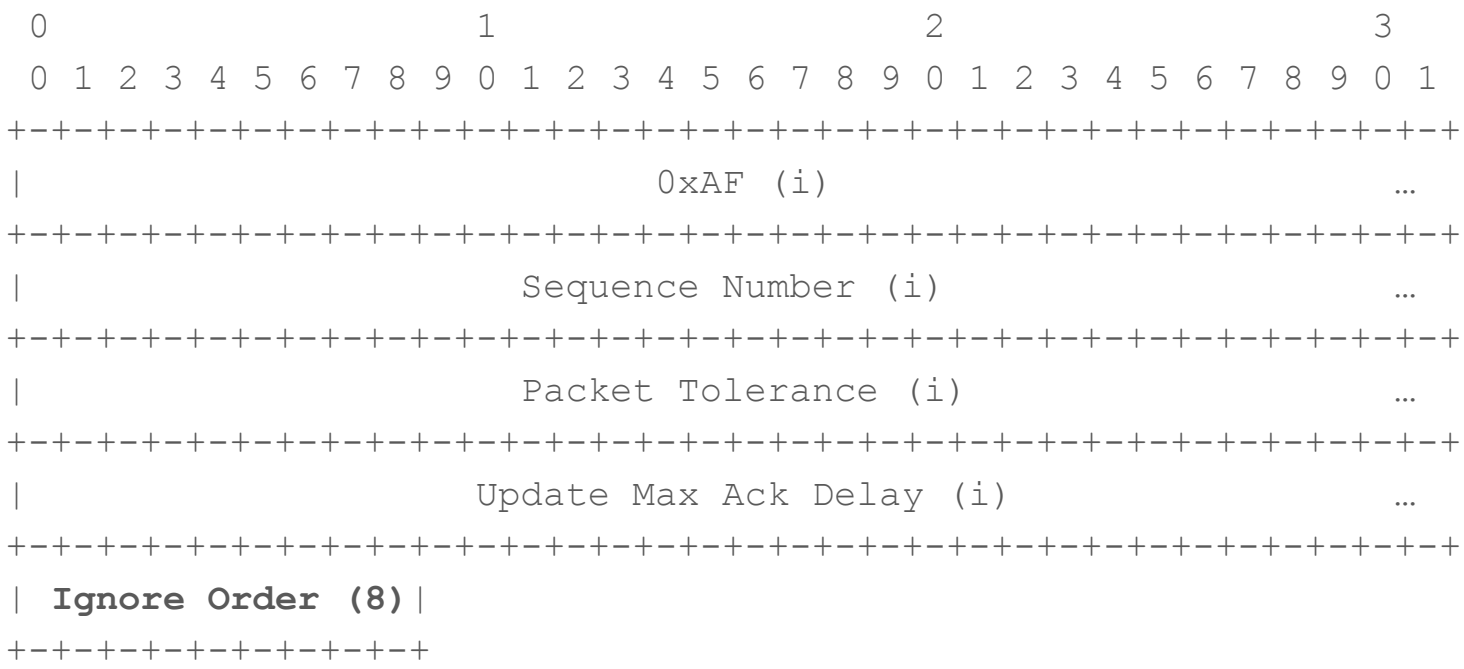
How low can “Update Max Ack Delay” be?

Transport Parameter: `min_ack_delay (0xde1a)`

the minimum amount of time (in microseconds)
by which the endpoint can delay an acknowledgement

Used for negotiating use of this extension

ACK_FREQUENCY frame



Ignore Order :

- 0x01 means *always* delay
- (i.e., DONT send immediate ACK on reordering)
- Used by senders that expect or observe reordering