QUIC
Internet-Draft
Intended status: Standards Track
Expires: 28 April 2022

J. Iyengar
Fastly
I. Swett
Google

25 October 2021

QUIC Acknowledgement Frequency
draft-ietf-quic-ack-frequency-01

Abstract

This document describes a QUIC extension for an endpoint to control its peer's delaying of acknowledgements.

Note to Readers

Discussion of this draft takes place on the QUIC working group mailing list (quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=quic. Source code and issues list for this draft can be found at https://github.com/quicwg/ack-frequency.

Working Group information can be found at https://github.com/quicwg.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 April 2022.

Copyright Notice

Table of Contents

1. Introduction

This document describes a QUIC extension for an endpoint to control its peer's delaying of acknowledgements.

1.1. Terms and Definitions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

In the rest of this document, "sender" refers to a QUIC data sender (and acknowledgement receiver). Similarly, "receiver" refers to a QUIC data receiver (and acknowledgement sender).

An "acknowledgement packet" refers to a QUIC packet that contains only an ACK frame.

This document uses terms, definitions, and notational conventions described in Section 1.2 and Section 1.3 of [QUIC-TRANSPORT].

2. Motivation

A receiver acknowledges received packets, but it can delay sending these acknowledgements. The delaying of acknowledgements can impact connection throughput, loss detection and congestion controller performance at a data sender, and CPU utilization at both a data sender and a data receiver.

Reducing the frequency of acknowledgement packets can improve connection and endpoint performance in the following ways:

* Sending UDP packets can be noticeably CPU intensive on some platforms. Reducing the number of packets that only contain acknowledgements can therefore reduce the amount of CPU consumed at a data receiver. Experience shows that this cost reduction can be significant for high bandwidth connections.

* Similarly, receiving and processing UDP packets can also be CPU

intensive, and reducing acknowledgement frequency reduces this cost at a data sender.

* Severely asymmetric link technologies, such as DOCSIS, LTE, and satellite links, connection throughput in the data direction becomes constrained when the reverse bandwidth is filled by acknowledgment packets. When traversing such links, reducing the number of acknowledgments allows connection throughput to scale much further.

As discussed in Section 9 however, there can be undesirable consequences to congestion control and loss recovery if a receiver uniltaerally reduces the acknowledgment frequency. A sender's constraints on the acknowledgement frequency need to be taken into account to maximize congestion controller and loss recovery performance.

[QUIC-TRANSPORT] currently specifies a simple delayed acknowledgement mechanism that a receiver can use: send an acknowledgement for every other packet, and for every packet that is received out of order (Section 13.2.1 of [QUIC-TRANSPORT]). This simple mechanism does not allow a sender to signal its constraints. This extension provides a mechanism to solve this problem.

3. Negotiating Extension Use

Endpoints advertise their support of the extension described in this document by sending the following transport parameter (Section 7.2 of [QUIC-TRANSPORT]):

min_ack_delay (0xff03de1a): A variable-length integer representing the minimum amount of time in microseconds by which the endpoint can delay an acknowledgement. This limit could be based on the receiver's clock or timer granularity.

An endpoint's min_ack_delay MUST NOT be greater than its max_ack_delay. Endpoints that support this extension MUST treat receipt of a min_ack_delay that is greater than the received max_ack_delay as a connection error of type

TRANSPORT_PARAMETER_ERROR. Note that while the endpoint's max_ack_delay transport parameter is in milliseconds (Section 18.2 of [QUIC-TRANSPORT]), min_ack_delay is specified in microseconds.

The min_ack_delay transport parameter is a unilateral indication of support for receiving ACK_FREQUENCY frames. If an endpoint sends the transport parameter, the peer is allowed to send ACK_FREQUENCY frames independent of whether it also sends the min_ack_delay transport parameter or not.

Receiving a min_ack_delay transport parameter indicates that the peer might send ACK_FREQUENCY frames in the future. Until an ACK_FREQUENCY frame is received, receiving this transport parameter does not cause the endpoint to change its acknowledgement behavior.

Endpoints MUST NOT remember the value of the min_ack_delay transport parameter they received. Consequently, ACK_FREQUENCY frames cannot be sent in 0-RTT packets, as per Section 7.4.1 of [QUIC-TRANSPORT].

This Transport Parameter is encoded as per Section 18 of [QUIC-TRANSPORT].

4. ACK_FREQUENCY Frame

Delaying acknowledgements as much as possible reduces both work done by the endpoints and network load. An endpoint's loss detection and congestion control mechanisms however need to be tolerant of this delay at the peer. An endpoint signals the frequency it wants to receive ACK frames to its peer using an ACK_FREQUENCY frame, shown below:

```
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)
}
```

Following the common frame format described in Section 12.4 of [QUIC-TRANSPORT], ACK_FREQUENCY frames have a type of 0xaf, and contain the following fields:

Sequence Number: A variable-length integer representing the sequence number assigned to the ACK_FREQUENCY frame by the sender to allow receivers to ignore obsolete frames, see Section 5.

Ack-Eliciting Threshold: A variable-length integer representing the maximum number of ack-eliciting packets the recipient of this frame can receive without sending an acknowledgment. In other words, an acknowledgement is sent when more than this number of ack-eliciting packets have been received. Since this is a maximum value, a receiver can send an acknowledgement earlier. A value of 0 results in a receiver immediately acknowledging every ack-eliciting packet.

Request Max Ack Delay: A variable-length integer representing the value to which the endpoint requests the peer update its max_ack_delay (Section 18.2 of [QUIC-TRANSPORT]). The value of this field is in microseconds, unlike the 'max_ack_delay' transport parameter, which is in milliseconds. Sending a value smaller than the min_ack_delay advertised by the peer is invalid. Receipt of an invalid value MUST be treated as a connection error of type PROTOCOL_VIOLATION.

Reserved: This field has no meaning in this version of ACK_FREQUENCY. The value of this field MUST be 0x00. Receipt of any other value MUST be treated as a connection error of type FRAME_ENCODING_ERROR.

Ignore Order: A 1-bit field representing a boolean truth value. 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 (Section 7.1). 0 represents 'false' and 1 represents 'true'.

Ignore CE: A 1-bit field representing a boolean truth value. 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 (Section 7.1). 0 represents 'false' and 1 represents

'true'.

ACK_FREQUENCY frames are ack-eliciting. However, their loss does not require retransmission if an ACK_FREQUENCY frame with a larger Sequence Number value has been sent.

An endpoint MAY send ACK_FREQUENCY frames multiple times during a connection and with different values.

An endpoint will have committed a max_ack_delay value to the peer, which specifies the maximum amount of time by which the endpoint will delay sending acknowledgments. When the endpoint receives an ACK_FREQUENCY frame, it MUST update this maximum time to the value proposed by the peer in the Request Max Ack Delay field.

5. Multiple ACK_FREQUENCY Frames

An endpoint can send multiple ACK_FREQUENCY frames, and each one of them can have different values in all fields. An endpoint MUST use a sequence number of 0 for the first ACK_FREQUENCY frame it constructs and sends, and a strictly increasing value thereafter.

An endpoint MUST allow reordered ACK_FREQUENCY frames to be received and processed, see Section 13.3 of [QUIC-TRANSPORT].

On the first received ACK_FREQUENCY frame in a connection, an endpoint MUST immediately record all values from the frame. The sequence number of the frame is recorded as the largest seen sequence number. The new Ack-Eliciting Threshold and Request Max Ack Delay values MUST be immediately used for delaying acknowledgements; see Section 7.

On a subsequently received ACK_FREQUENCY frame, the endpoint MUST check if this frame is more recent than any previous ones, as follows:

* If the frame's sequence number is not greater than the largest one seen so far, the endpoint MUST ignore this frame.

* If the frame's sequence number is greater than the largest one

seen so far, the endpoint MUST immediately replace old recorded state with values received in this frame. The endpoint MUST start using the new values immediately for delaying acknowledgements; see Section 7. The endpoint MUST also replace the recorded sequence number.

6. IMMEDIATE_ACK Frame

A sender can use an ACK_FREQUENCY frame to reduce the number of acknowledgements sent by a receiver, but doing so increases the chances that time-sensitive feedback is delayed as well. For example, as described in Section 9.1, delaying acknowledgements can increase the time it takes for a sender to detect packet loss. The IMMEDIATE_ACK frame helps mitigate this problem.

An IMMEDIATE_ACK frame can be useful in other situations as well. For example, it can be used with a PING frame (Section 19.2 of [QUIC-TRANSPORT]) if a sender wants an immediate RTT measurement or if a sender wants to establish receiver liveness as quickly as possible.

An endpoint SHOULD send a packet containing an ACK frame immediately upon receiving an IMMEDIATE_ACK frame. An endpoint MAY delay sending an ACK frame despite receiving an IMMEDIATE_ACK frame. For example, an endpoint might do this if a large number of received packets contain an IMMEDIATE_ACK or if the endpoint is under heavy load.

```
IMMEDIATE_ACK Frame {
  Type (i) = 0xac,
}
```

7. Sending Acknowledgments

Prior to receiving an ACK_FREQUENCY frame, endpoints send acknowledgements as specified in Section 13.2.1 of [QUIC-TRANSPORT].

On receiving an ACK_FREQUENCY frame and updating its recorded max_ack_delay and Ack-Eliciting Threshold values (Section 5), the endpoint MUST send an acknowledgement when one of the following conditions are met:

* Since the last acknowledgement was sent, the number of received

ack-eliciting packets is greater than or equal to the recorded Ack-Eliciting Threshold.

* Since the last acknowledgement was sent, max_ack_delay amount of time has passed.

Section 7.1, Section 7.2, and Section 7.3 describe exceptions to this strategy.

An endpoint is expected to bundle acknowledgements when possible. Every time an acknowledgement is sent, bundled or otherwise, all counters and timers related to delaying of acknowledgments are reset.

The receiver of an ACK_FREQUENCY frame can continue to process multiple available packets before determining whether to send an ACK frame in response, as stated in Section 13.2.2 of [QUIC-TRANSPORT].

7.1. Response to Out-of-Order Packets

As specified in Section 13.2.1 of [QUIC-TRANSPORT], endpoints are expected to send an acknowledgement immediately on receiving a reordered ack-eliciting packet. This extension modifies this behavior.

If the endpoint has not yet received an ACK_FREQUENCY frame, or if the most recent frame received from the peer has an Ignore Order value of false (0x00), the endpoint MUST immediately acknowledge any subsequent packets that are received out of order.

If the most recent ACK_FREQUENCY frame received from the peer has an Ignore Order value of true (0x01), the endpoint does not make this exception. That is, the endpoint MUST NOT send an immediate acknowledgement in response to packets received out of order, and instead continues to use the peer's Ack-Eliciting Threshold and max_ack_delay thresholds for sending acknowledgements.

7.2. Expediting Congestion Signals

An endpoint SHOULD send an immediate acknowledgement when a packet marked with the ECN Congestion Experienced (CE) codepoint in the IP header is received and the previously received packet was not marked CE.

Doing this maintains the peer's response time to congestion events, while also reducing the ACK rate compared to Section 13.2.1 of [QUIC-TRANSPORT] during extreme congestion or when peers are using DCTCP [RFC8257] or other congestion controllers that mark more frequently than classic ECN [RFC3168].

7.3. Batch Processing of Packets

For performance reasons, an endpoint can receive incoming packets from the underlying platform in a batch of multiple packets. This batch can contain enough packets to cause multiple acknowledgements to be sent.

To avoid sending multiple acknowledgements in rapid succession, an endpoint MAY process all packets in a batch before determining whether a threshold has been met and an acknowledgement is to be sent in response.

8. Computation of Probe Timeout Period

On sending an update to the peer's max_ack_delay, an endpoint can use this new value in later computations of its Probe Timeout (PTO) period; see Section 5.2.1 of [QUIC-RECOVERY]. The endpoint MUST however wait until the ACK_FREQUENCY frame that carries this new value is acknowledged by the peer.

Until the frame is acknowledged, the endpoint MUST use the greater of the current max_ack_delay and the value that is in flight when computing the PTO period. Doing so avoids spurious PTOs that can be caused by an update that increases the peer's max_ack_delay.

While it is expected that endpoints will have only one ACK_FREQUENCY frame in flight at any given time, this extension does not prohibit having more than one in flight. Generally, when using max_ack_delay for PTO computations, endpoints MUST use the maximum of the current value and all those in flight.

When the number of in-flight ack-eliciting packets is larger than the ACK-Eliciting Threshold, an endpoint can expect that the peer will not need to wait for its max_ack_delay period before sending an acknowledgement. In such cases, the endpoint MAY therefore exclude the peer's 'max_ack_delay' from its PTO calculation. Note that this optimization requires some care in implementation, since it can cause premature PTOs under packet loss when ignore_order is enabled.

9. Implementation Considerations

There are tradeoffs inherent in a sender sending an ACK_FREQUENCY frame to the receiver. As such it is recommended that implementers experiment with different strategies and find those which best suit their applications and congestion controllers. There are, however, noteworthy considerations when devising strategies for sending ACK_FREQUENCY frames.

9.1. Loss Detection

A sender relies on receipt of acknowledgements to determine the amount of data in flight and to detect losses, e.g. when packets experience reordering, see [QUIC-RECOVERY]. Consequently, how often a receiver sends acknowledgments determines how long it takes for losses to be detected at the sender.

9.2. New Connections

Many congestion control algorithms have a startup mechanism during the beginning phases of a connection. It is typical that in this period the congestion controller will quickly increase the amount of data in the network until it is signalled to stop. While the mechanism used to achieve this increase varies, acknowledgments by the peer are generally critical during this phase to drive the congestion controller's machinery. A sender can send ACK_FREQUENCY frames while its congestion controller is in this state, ensuring that the receiver will send acknowledgments at a rate which is optimal for the the sender's congestion controller.

9.3. Window-based Congestion Controllers

Congestion controllers that are purely window-based and strictly adherent to packet conservation, such as the one defined in [QUIC-RECOVERY], rely on receipt of acknowledgments to move the congestion window forward and send additional data into the network. Such controllers will suffer degraded performance if acknowledgments are delayed excessively. Similarly, if these controllers rely on the timing of peer acknowledgments (an "ACK clock"), delaying

acknowledgments will cause undesirable bursts of data into the network.

9.4. Connection Migration

To avoid additional delays to connection migration confirmation when using this extension, a client can bundle an IMMEDIATE_ACK frame with the first non-probing frame (Section 9.2 of [QUIC-TRANSPORT]) it sends or it can simply send an IMMEDIATE_ACK frame, which is a non-probing frame.

An endpoint's congestion controller and RTT estimator are reset upon confirmation of migration (Section 9.4 of [QUIC-TRANSPORT]), which can impact the number of acknowledgements received after migration. An endpoint that has sent an ACK_FREQUENCY frame earlier in the connection SHOULD update and send a new ACK_FREQUENCY frame immediately upon confirmation of connection migration.

9.5. Path MTU Discovery

A sender might use timers to detect loss of PMTUD probe packets. A sender SHOULD bundle an IMMEDIATE_ACK frame with any PTMUD probes to avoid triggering such timers.

10. Security Considerations

TBD.

11. IANA Considerations

TBD.

12. References

12.1. Normative References

[QUIC-TRANSPORT]
Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[QUIC-RECOVERY]
Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/rfc/rfc9002>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

12.2. Informative References

[RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <https://www.rfc-editor.org/rfc/rfc8257>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/rfc/rfc3168>.

Appendix A. Change Log

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

Acknowledgments

The following people directly contributed key ideas that shaped this draft: Bob Briscoe, Kazuho Oku, Marten Seemann.

Authors' Addresses

Jana Iyengar
Fastly

Email: jri.ietf@gmail.com

Ian Swett
Google

Email: ian.swett@google.com