Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 6, 2020

G. Fairhurst
A. Custura
T. Jones
University of Aberdeen

March 05, 2020

Changing the Default QUIC ACK Policy
draft-fairhurst-quic-ack-scaling-01

Abstract

ACKs are used by transport protocols to confirm delivery of packets. The transmission of ACKs consumes resources at the receiver, in the network and at the sender. On network paths where there is significant path asymmetry, acknowledgments of data receipt can reduce the efficient use of network capacity. This effect occurs when the return capacity is significantly more constrained than the forward capacity, or the cost of transmission per packet is a significant component of the total transmission cost. In these cases, reducing the ratio of acknowledgements to data can improve link utilization and reduce link transmission costs. It can also reduce processing overhead at the sender and receiver.

This document proposes a change to the default acknowledgement policy of the QUIC transport protocol to improve performance over paths with appreciable asymmetry.

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 September 6, 2020.

Table of Contents

1. Introduction

The current design of QUIC [I-D.ietf-quic-transport] currently

proposes a default acknowledgement (ACK) ratio of 1:2 (at least one ACK for every 2 ack-eliciting packets) inspired by current recommendations for TCP, Appendix A, see. This document proposes an increase in the ratio of ACK packets to data packets from 1:2 to 1:10 for QUIC flows.

2. Motivation

When the characteristics of the forward and return paths are not symmetric, the transmission of ACKs can adversely impact either transport performance or the cost of sending data across a link.

2.1. ACK Ratio Impact on Asymmetric Paths

TCP Performance Implications of Network Path Asymmetry [RFC3449] describes a series of problems and mitigations when transports use an asymmetric path. Performance problems arise in several access networks, including bandwidth-asymmetric networks (such as broadband satellite access, DOCSIS cable networks, cellular mobile, WiFi, etc).

Where the ACK rate is limited by the capacity of the return path, this constrains the maxium throughput for the forward path. The ACK traffic also competes for capacity and/or transmission opportunities with other traffic that shares a constrained return path. This motivates the need to reduce the volume of ACK traffic (increase the number of segments/packets that are acknowledged by each ACK).

Capacity is not the only asymmetric path constraint. Sending ACKs can consume significant transmission resources and the cost of transmitting ACKs can become a significant part of the cost of transmission when using a network segment. In many wireless technologies, there is appreciable overhead for the transmission of each packet burst (data and ACK). There can also be associated costs (e.g. in radio resource management and transmission scheduling) that are often different for the forward and return paths because they use different technologies or configurations. This provides an incentive to reduce the rate of ACK traffic.

2.2. Terminology

The key words "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.

2.3. Adapting the ACK Ratio in Current Transport Specifications

Various methods have been proposed to modify the ACK Ratio used by transport protocols. Two examples follow:

ACK-CC [RFC5690] proposed a method to control the rate of ACKs to avoid the return path from becoming congested, but this did not achieve wide-scale deployment.

The Datagram Congestion Control Protocol (DCCP) [RFC4340] has methods to control the ACK ratio at the receiver. DCCP specifies a TCP-friendly congestion control [RFC4341], which includes the ability to use signaling to allow this sender to adjust the receiver ACK ratio within certain parameters. This also was not widely used.

End-to-end transport methods do not have a way to detect and account for the cost of ACK transmission, and are difficult to tune to adapt to delays introduced by the path (especially when fair-queueing and other link scheduling methods hide the effects of increased ACK traffic). They therefore only provide a partial mitigation to the impacts when sending ACKs over asymmetric paths.

2.4. Adapting the ACK Ratio in the Network

An alternative approach has been deployed for TCP that uses a middlebox in the network to thin the rate of ACKs. This method is used with paths that exhibit significant ACK symmetry to improve performance of TCP when it uses an ACK Ratio of 1:2 [RFC3449]. Performance Enhancing Proxies (PEPs) can implement these functions when they detect/know an upstream link is filled with TCP ACKs.

Removing redundant ACKs (also known as "ACK Thinning") leads to TCP stretch ACKs (where a single ACK acknowledges more than two TCP segments). The introduction of TCP Appropriate Byte Counting (ABC) i[RFC3465] partly mitigates the impact of stretch ACKs, and also recommends burst mitigation techniques at a TCP sender.

These methods only work when ACKs can be observed by a device in the

network. QUIC uses an encrypted feedback packet to communicate an ACK. The use of encryption intentionally prevents such in-network optimisations. Compared to TCP, performance of QUIC is therefore disadvantaged when QUIC uses an ACK Ratio of 1:2.

QUIC ACKs are also significantly larger in size than TCP ACKS (e.g., 1.5-2 times), which means additional processing overhead and link usage for all Internet paths, with a significant impact on asymmetric links, where this can also limit throughput.

3. Updating the Default ACK Ratio for QUIC

Any ACK policy that changes the ACK ratio from 1:2 needs to compensate for three issues:

- A reduced frequency of feedback can increase the time to detect congestion, impacting the congestion control algorithm. QUIC mitigates this by using PTO-based retransmission.

- A reduced frequency of feedback can increase the time to detect loss, impacting the loss recovery algorithm, and potentially leading to cases of spurious retransmission;

- A reduced ACK rate can lead to bursts of acknowledged packets, and introduces a need for burst mitigation at the sender;

3.1. Considerations Updating the Default QUIC ACK Policy

The QUIC transport protocol currently specifies a maximum ACK Delay, which is communicated by the sender to indicate the maximum time an endpoint will delay sending acknowledgments. A default of 25 milliseconds is recommended and QUIC currently recommends a default ACK Ratio of 1:2 [I-D.ietf-quic-transport].

This document proposes changing the default QUIC behaviour to send an ACK for at least every 10 received packets. Further background to the proposed method is detailed in the Annexe (Appendix B).

The ACK Delay timer ensures ACKs are not unduly delayed. The effect of a large delay could be significant when a stretched ACK acknowledges more packets, and therefore the proposed method also

ensures feedback at least four times per RTT (less important when the RTT is greater than 100ms and the ACK Delay forms a small part of the total RTT).

Loss detection can be impacted by delayed acknowledgments. Although timer-based methods (e.g., using the QUIC Probe Timeout (PTO), see section 5.1 of [I-D.ietf-quic-recovery]) can reduce the reliance on ACKs to detect loss, prompt communication of ACK ranges after loss is still important to efficient loss recovery.

Since the introduction of the specification to allow a larger TCP Initial Window (IW) [RFC6928], there has been deployment experience using TCP with an IW of 10 segments at startup. QUIC continues this practice, which in part motivates the need for a QUIC transport protocol to operate when a burst of acknowledgements for up to ten packets is received. QUIC therefore transport recommends the use of pacing to mitigate packet bursts being generated by a sender (see section 6.8 of [I-D.ietf-quic-recovery]).

The proposed method seeks to allow QUIC to effectively operate over asymmetric paths.

3.2. During Slow Start

Some congestion controllers can benefit from frequent feedback during an initial slow start period, where the sender is probing for available path capacity. This update therefore does not change the recommended ACK Ratio during the initial part of slow start. This ensures stretch ACKs do not impact the initial rate of growth for the congestion window.

A suitable method might send an ACK frame for every received ack-eliciting packets for the first 100 received packets if max_ack_delay time has passed since the oldest unacknowledged data was received.

A receiver typically has no understanding of the senders congestion control state. The number 100 reflects a trade-off, corresponding to

an appreciable opening of the sender's congestion window.

A congestion controller typically re-enters slow start after congestion is detected. The need to probe to re-establish a working congestion window is helped by the ACK policy after loss.

3.3. After Slow Start

A receiver sends an ACK if a period more than MIN(max_ack_delay,min_rtt/4) has passed since receiving the oldest unacknowledged data OR it has accumulated 10 unacknowledged packets.

3.4. When Encountering Loss/Congestion

Following detected loss or congestion, a receiver sends ACKs according to section 13.2.1 of QUIC transport [I-D.ietf-quic-transport].

4. Recommended ACK Policy

The max_ack_delay needs to be set so that at least several samples can be generated per RTT to estimate the path RTT.

A QUIC receiver can generate one ACK frame for every received ack-eliciting packet. TCP recommends that a receiver generates an ACK corresponding to every second MSS of received data, Section 4.2 of RFC 5681 [RFC5681], however RFC 3449 [RFC3449] also notes the need for, and deployment of, methods to further reduce the number of TCP ACKs in networks with asymmetric paths.

An ACK frame SHOULD be generated for at least every tenth ack-eliciting packet. The maximum of receiving not more than 10 ack-

eliciting packets is derived from the recommended TCP Initial Window [RFC6928].

Using an appropriate value for max_ack_delay, or ensuring a minimum number of ACKs per RTT (e.g 8) would mitigate the effect of ACK loss on RTT estimation and aids performance for low-rate interactive applications.

4.1. Further Tuning of the ACK Policy

In situations where the default method is not sufficient, the ACK Ratio might be further tuned by server, as described in [I-D.iyengar-quic-delayed-ack]. This could also permit the ACK method to be adapted to match the behaviour of new congestion control algorithms. Reducing the rate of ACKs can also lower the computational effort required to process ACKs at the sender and receiver. For instance, this could reduce the workload for high speed network interfaces by reducing the rate of cache ejection for Generic Receiver Offload (GRO).

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

The security considerations for the QUIC transport protocol are described in [I-D.ietf-quic-transport].

7. References

7.1. Normative References

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-26 (work in progress), February 2020.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-27 (work in progress), February 2020.

[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/info/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/info/rfc8174>.

7.2. Informative References

[I-D.iyengar-quic-delayed-ack]
Iyengar, J. and I. Swett, "Sender Control of Acknowledgement Delays in QUIC", draft-iyengar-quic-delayed-ack-00 (work in progress), January 2020.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC2525] Paxson, V., Allman, M., Dawson, S., Fenner, W., Griner, J., Heavens, I., Lahey, K., Semke, J., and B. Volz, "Known TCP Implementation Problems", RFC 2525, DOI 10.17487/RFC2525, March 1999, <https://www.rfc-editor.org/info/rfc2525>.

[RFC3449] Balakrishnan, H., Padmanabhan, V., Fairhurst, G., and M. Sooriyabandara, "TCP Performance Implications of Network Path Asymmetry", BCP 69, RFC 3449, DOI 10.17487/RFC3449, December 2002, <https://www.rfc-editor.org/info/rfc3449>.

[RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, DOI 10.17487/RFC3465, February 2003, <https://www.rfc-editor.org/info/rfc3465>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, DOI 10.17487/RFC4341, March 2006, <https://www.rfc-editor.org/info/rfc4341>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC5690] Floyd, S., Arcia, A., Ros, D., and J. Iyengar, "Adding Acknowledgement Congestion Control to TCP", RFC 5690, DOI 10.17487/RFC5690, February 2010, <https://www.rfc-editor.org/info/rfc5690>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <https://www.rfc-editor.org/info/rfc6928>.

Appendix A. Summary of Recommended ACK Policy for TCP

RFC 5681 [RFC5681] clarifies the TCP ACK frequency described in RFC 1122 [RFC1122] to recommend the ACK policy for a TCP receiver: "an ACK SHOULD be generated for at least every second full-sized segment, and MUST be generated within 500 ms of the arrival of the first unacknowledged packet".

The TCP sender regards reception of an ACK as a positive indication that data has been received across the path. The congestion control algorithm uses this to increase the size of the congestion window, cwnd RFC 5681 [RFC5681].

To reduce the ACK Rate, a receiver can delay sending an ACK for a period of time called the ACK Delay. This can increase network efficiency. When the receiver delays ACKs, this reduces the rate of growth of the cwnd. TCP implementations often use heuristics such as DAAS (Delayed ACK after Slow Start) to mitigate this. This allows the receiver to estimate when the sender could be in the slow start phase of cwnd growth, and for a period of time sends an ACK for each received segment/packet (i.e., an ACK Ratio of 1:1).

ACKs can be lost on the return path (either through packet loss, or by intentional thinning of the ACK stream). If a sender does not receive an ACK for every second segment, a stretch ACK has occurred. RFC2525 [RFC2525] describes the significance of stretch ACK violations:

"this behavior will cause TCP senders to generate burstier traffic, which can degrade performance in congested environments. In addition, generating fewer ACKs increases the amount of time needed by the slow start algorithm to open the congestion window to an appropriate point, which diminishes performance in environments with large bandwidth-delay products. Finally, generating fewer ACKs may cause needless retransmission timeouts in lossy environments, as it increases the possibility that an entire window of ACKs is lost,

forcing a retransmission timeout."

RFC 3465 [RFC3465] further discusses the issue of bursts that may be caused by the interaction between ACK processing and congestion control. This motivates a need to deal with bursts within TCP. TCP senders can mitigate bursts by using Appropriate Byte Counting (ABC), which increases the congestion window in proportion to the amount of data sent into the network, rather than upon the arrival of each ACK. (This also defends against ACK Splitting, where multiple ACKs are received for parts of the same segment/packet).

There are other scenarios where a change to the TCP ACK policy would have improved performance. However, the design of TCP, and ossification of the protocol has made it hard for new mechanisms to be deployed. QUIC does not suffer from these design constraints.

Appendix B. Experiments Exploring an ACK Ratio of 1:10

The performance of QUIC is at a disadvantage compared to other transport protocols if designs use a conservative ACK Ratio, because QUIC can not modified by in-network middleboxes (such as used for TCP ACK Thinning). We argue that a default ratio of 1:2 is too conservative.

We used an experimental approach to examine a change to QUIC's ACK Policy <http://erg.abdn.ac.uk/~downloads/ackscaling.pdf>. This updated the default ACK Ratio from 1:2 to 1:10. Our tests show that this did not negatively impact the protocol. This reduced the amount of IP, UDP and ACK overhead by a factor of approximately 5. The implemented congestion control, was also not negatively impacted. These experiments were performed in January 2020 based on available implementations at that time.

Unlike TCP, QUIC sends other types of data frames in addition to ACK frames, increasing the total overhead on the return path. On asymmetrical paths an ACK Ratio of 1:10 may still reduce the ACK traffic, helping to avoid return path capacity limits impacting the ability to use the forward path capacity.

Figure 1 presents a table with a set of asymmetric scenarios. The columns present the rate of ACK traffic required (in kbps) to fill

each of the forward paths. The table shows the results for TCP (without a PEP), both for lossless communication. It considers the period after loss, when ACKs communicate the loss information. It also shows the impact of using an ACK Ratio of 1:10 with QUIC.

An ACK Ratio of 1:10 reduces the utilisation of the return path. Scenarios where the ACK traffic exceeds the return link capacity (i.e. where this limits the forward path capacity that can be used) are marked with a star. Note that the QUIC figure does not include

the encryption overhead, which would be dependent on the ciphers chosen. This would add several additional bytes for every QUIC packet.

	10/2 Mbps	50/10 Mbps	250/3 Mbps
TCP no loss	133 - 346	650 -1,730	3250 - 8,650*
TCP loss	346 - 560	1,730 - 2,800	8,650 - 14,000*
QUIC 1:2 no loss	144 - 438	720 - 2,190	3,600 - 10,950*
QUIC 1:2 loss	290 - *	1450 - *	7,250 - *
QUIC 1:10 no loss	28.8 - 87.6	144 - 438	720 - 2,190

Figure 1: ACK traffic required to fill the forward path in different loss and asymmetry scenarios. The QUIC figures do not include encryption overhead.

Figure 2 presents a table with the numbers of packets sent by two QUIC implementations using a 1:2 and a 1:10 ACK Ratio. This shows between a four and five time reduction in the number of packets sent.

	Chromium Draft 23	Quicly Draft 23
Packets sent	77,419	83,238
Packets on return path (1:2)	39,089 (50.4%)	41,108 (49.3%)
Packets on return path (1:10)	10,409 (13.4%)	9650 (11.5%)

Figure 2: Number of packets sent and received during a 100MB QUIC

transfer using different ACK ratios, for two implementations

Figure 3 presents a table with the number of bytes sent by one of the QUIC implementations using a 1:2 and a 1:10 ACK Ratio. This shows a reduction in the number of bytes on the return path from 2.7 to 0.7% of the total bytes sent. In these scenarios, this is sufficient to take full advantage of the forward path capacity.

	Chromium Draft 23
Bytes sent	110M
Bytes on return path (1:2 AR)	3,056KB (2.7%)
Bytes on return path (1:10 AR)	810KB (0.7%)

Figure 3: Number of bytes sent and received during a 100MB Chromium QUIC transfer using different ACK ratios

The number of bytes and packets reduces as expected when using an ACK Ratio of 1:10, without any increase in loss, on a high-latency path with an asymmetry of 5:1. This offers a clear benefit for paths that are capacity-constrained, as well as paths which would benefit from a reduction in the ACK Rate.

Authors' Addresses

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen AB24 3UE

UK

Email: gorry@erg.abdn.ac.uk

Ana Custura
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen AB24 3UE
UK

Email: ana@erg.abdn.ac.uk

Tom Jones
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen AB24 3UE
UK

Email: tom@erg.abdn.ac.uk