Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: March 18, 2021

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

September 14, 2020

Changing the Default QUIC ACK Policy
draft-fairhurst-quic-ack-scaling-03

Abstract

Acknowledgement packets (ACKs) are used by transport protocols to confirm the delivery of packets, and their reception is used in a variety of other ways (to measure path round trip time, to gauge path congestion, etc). However, the transmission of ACKs also consumes resources at the receiver, forwarding resource in the network and processing resources at the sender.

On network paths with significant path asymmetry, transmission of ACKs can limit the available throughput or can reduce the efficient use of network capacity. This occurs when the return capacity is significantly more constrained than the forward capacity, and/or the cost of transmission per packet is a significant component of the total transmission cost. In these cases, reducing the ratio of ACK packets to data packets can improve link utilisation 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 March 18, 2021.

Table of Contents

1. Introduction

Acknowledgement packets (ACKs) are used by transport protocols to confirm the delivery of packets, and their reception is used in a variety of other ways (to measure path round trip time, to gauge path

congestion, etc) [Cust20]. However, the transmission of ACKs also consumes resources at the receiver, forwarding resource in the network and processing resources at the sender.

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, see Appendix A.

This document motivates 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 [RFC3449], the transmission of ACKs can adversely impact either transport performance and/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) [Cust20].

Where the ACK rate is limited by the capacity of the return path, this constrains the maximum throughput for the forward path. ACK traffic also competes for capacity and/or transmission opportunities with other traffic that shares a constrained return path. This motivates a 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.

These provides an incentive to reduce the rate of ACK traffic generated by a transport protocol.

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

We define the ACK Ratio as the ratio between the number of packets that are received by a transport endpoint (on the forward path) and the number of ACK packets that are returned (on the return path). A simple protocol may send an ACK for each data packet, resulting in an ACK Ratio of 1:1. One that acknowledges alternate data packets has a ratio of 1:2. Most IETF-defined protocols specify a default ACK Ratio. Note on the use of ACKs in TCP and the resulting ACK Ratio are provided in Appendix A.

Various methods have been proposed to modify the ACK Ratio used by transport protocols. Two examples follow, based on a receiver understanding whether the return path has become congested:

Various proposals and analyses methods to allow TCP to dynamically adjust the AR (e.g., [Lan08]). 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 signalling that allows this sender to adjust the receiver ACK ratio within certain parameters. This also was not widely used.

These methods are suggested to help when there is congestion on the return path. However, end-to-end transport methods do not have a way to detect and account for the cost of ACK transmission on a link along the return path, and are difficult to tune to adapt to delays introduced by links (e.g., the variations produced by radio resource management). Transport adaptation therefore can only provide a partial mitigation to the impacts when sending ACKs over an asymmetric paths.

2.4. Adapting the TCP 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 asymmetry to improve

performance of TCP. They can modify the ACK Ratio experienced by a network link from the default TCP ACK Ratio of 1:2 [RFC3449]. Performance Enhancing Proxies (PEPs) implement these functions when they detect/know an upstream link is (or is likely to be) filled with TCP ACKs, or on cross-layer information provided by the physical layer.

Removing redundant TCP ACKs (also known as "ACK Thinning") leads to stretch ACKs (where a single ACK acknowledges more than two TCP segments). Stretch ACKs have been observed on Internet paths [All98] and are are now common [Din15] for various reasons.

The introduction of TCP Appropriate Byte Counting (ABC) [RFC3465] mostly mitigates the impact of stretch ACKs, and also recommends burst mitigation techniques at a TCP sender.

2.5. QUIC ACKs

QUIC, like other transports, generates ACK information and sends this in QUIC packets on the return path.

ACK Thinning methods can only be used when ACKs can be observed by a network device on the path. In contrast, QUIC uses an encrypted feedback packet to communicate an ACK. This use of encryption intentionally prevents such in-network optimisations.

A typical QUIC ACK is around 25% larger than a corresponding TCP ACK, and can still be larger when there is loss/reordering. This means additional processing overhead and link usage for all Internet paths, with a significant impact on asymmetric links, where this can also limit throughput:

- The smallest IPv4 TCP ACK is 20 bytes, resulting in a 40 byte IPv4 TCP ACK packet, although this could be up to 40 bytes larger when TCP options are present.

- The TCP ACK size often increases following a reordering event due to the presence of SACK option blocks. The maximum size for IPv4 is max 60 bytes.

- The smallest IPv4 QUIC ACK packet is 51 bytes. The ACK frame is only 4 bytes, but is sent using a QUIC packet. For IPv4 this is: 20 (IP) + 8 (UDP) + 3 (1+1+1 QUIC header ... Assuming a 1 B CID) + 4 (minimum ACK frame size) + 16 (crypto overhead). Other types of frames may also be sent in the same packet, increasing this size. The QUIC ACK packet size also becomes larger for very long connections due to the varint encoding.

- The packet size for an IPv4 QUIC ACK increases after a reordering event due to the inclusion of ACK range information. The maximum size of this information is limited to the size of a QUIC packet. Implementation may further limit the size (often observed as 100's of bytes depending on the pattern of loss/reordering).

Compared to TCP, the performance of QUIC is therefore disadvantaged when QUIC uses an ACK Ratio of 1:2.

3. Updating the Default ACK Ratio for QUIC

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

- 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. QUIC mitigates this by using timer-based retransmission (using the PTO).

- A reduced frequency of feedback can increase the time to detect and react to congestion, impacting the congestion control algorithm. The speed of loss detection can be impacted by delayed acknowledgements. 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.

- 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

A QUIC receiver can generate one ACK frame for every received ack-eliciting packet, but normally aggregates the ACK information into a cumulative ACK. The QUIC transport protocol currently recommends a default ACK Ratio of 1:2 [I-D.ietf-quic-transport]. Additionally, QUIC relies on pacing, rather than ACK-Clocking as a burst mitigation. Hence reception of larger cumulative ACKs does not normally have a significant impact on the sender's traffic.

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 transport recommends the use of pacing to mitigate packet bursts generated by a sender (see section 6.8 of [I-D.ietf-quic-recovery])

This document proposes changing the 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 C).

The QUIC transport protocol also currently specifies a maximum ACK Delay, which is communicated by the sender at connection setup to indicate the maximum time an endpoint will delay sending acknowledgements. A default of 25 milliseconds is recommended [I-D.ietf-quic-transport]. The ACK Delay timer ensures ACKs are not unduly delayed (e.g., when data packets are spaced in time, or at the end of a packet burst). The effect of a large delay could be significant when a stretch ACK acknowledges more QUIC packets.

When the ACK Ratio is increased, an appropriate choice of the maximum ACK Delay becomes more important - because it could otherwise withhold ACK information for a time long enough to impact protocol performance or operation. The proposed therefore method also triggers ACK feedback at least four times per RTT (this additional rule is less important when the RTT is greater than 100ms and the ACK Delay forms a small part of the total RTT).

3.2. Mitigating the Impact of ACKs during Slow Start

A sender during slow start is often cwnd-limited, so any additional delay in returning an ACK has the effect of increasing the duration of the slow-start phase (see Appendix B). The requested change is:

> The receiver can provide a higher rate of acknowledge for the first 100 ACK-eliciting packets, where it acknowledges at least every second received ACK-eliciting packet. A suitable method might send an ACK frame for every two received ack-eliciting packets for the first 100 received packets if max_ack_delay time has passed since the oldest unacknowledged data was received.

NOTE: The original design of TCP used each ACK as a trigger to increase the congestion window, motivating an initial ACK Ratio of 1:1 (as in DAASS Appendix A).

NOTE: A receiver typically has no understanding of the sender's congestion control state, so the number 100 reflects a trade-off, corresponding to an appreciable opening of the sender's congestion window.

NOTE: A congestion controller typically re-enters slow start after congestion is detected, if the congestion window is collapsed to a small value, this could motivate again using this method. However, the receiver is typically unaware of this event, and we do not propose this is considered.

3.3. ACKs after Slow Start

We recommend an ACK frame should be generated for at least every tenth ack-eliciting packet. The requested change is:

> The receiver SHOULD send an ACK frame if a period more than MIN(max_ack_delay,min_rtt/4) has passed since receiving the oldest unacknowledged data or when the received has accumulated at most 10 unacknowledged ack-eliciting packets.

NOTE: The maximum of receiving not more than 10 ack-eliciting packets is derived from the recommended TCP Initial Window [RFC6928].

NOTE: This utilises the receiver's estimated min_rtt, when this is known.

3.4. ACKs after re-ordering is Detected

Prompt and reliable communication of ACK ranges after loss is important for efficient loss recovery. This suggests that additional ACKs may be needed after a reordering event to protect the sender from potential ACK loss in the return direction. However, such ACKs also consume return path capacity and the number of additional packets needs to be considered.

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

NOTE: A future recommendation could recommend reducing the ACK frequency after loss/re-ordering.

3.5. 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 permit the ACK method to be adapted to match the behaviour of a new congestion control algorithm. Reducing the rate of ACKs can also lower the computational effort required to process ACKs at the sender and receiver, important for some high speed connections. For instance, this could reduce the workload for high speed network interfaces by reducing the rate of cache ejection for Generic Receiver Offload (GRO).

The change to the default motivated in this document is not related to such further tuning of the ACK policy.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

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

6. References

6.1. Normative References

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-30 (work in progress), September 2020.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-30 (work in progress), September 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>.

6.2. Informative References

[All98] Allman, M., "On the Generation and use of TCP Acknowledgments; ACM SIGCOMM CCR, 28 (5) p4-21.", October 1998.

[Cust20] Custura, A., Jones, T., and G. Fairhurst, "Rethinking ACKs at the Transport Layer; FIT Workshop, IEEE IFIP Networking", June 2020.

[Din15] Ding, H., "TCP Stretch Acknowledgements and Timestamps, Findings and Implications for Passive RTT Measurement; ACM SIGCOMM CCR, 45 (3) p20-27.", July 2015.

[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.

[Lan08] Landstroem, S., "TCP/IP Technology for Modern Network Environments; PhD Thesis", 2008.

[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

A TCP sender regards reception of a TCP 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.

RFC 5681 [RFC5681] clarifies the TCP ACK frequency described in RFC 1122 [RFC1122]. This recommends that a receiver generates an ACK corresponding to every second maximum segment size, MSS, of received data, Section 4.2 of RFC 5681 [RFC5681], specifies 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". 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.

When a receiver decides to delay ACKs, this can reduce the rate of growth of the cwnd.

Some congestion controllers can benefit from frequent feedback during an initial slow start period, where the sender is probing for available path capacity. When a TCP sender uses each ACK to increase the cwnd, this directly impacts the cwnd growth. 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). In contrast, a TCP congestion controller can increase its congestion window based on the number of bytes acknowledged, not the number of ACK packets. (QUIC chooses this approach).

A delayed ACK can also increase the time to complete slow start, by introducing an additional delay when returning ACKs. This effect is different to a direct change to the cwnd, but never-the-less can impact the performance, especially over paths where the ACK Delay period is comparable to the RTT.

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).

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, similar to using a larger ACK Ratio. 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."

There are other scenarios where a change to the TCP ACK policy could 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. Understanding ACKs in Slow Start

This section provides diagrams to help explain the way ACKs are using during slow start. This assumes the sender uses volume-based methods to increase cwnd, as in QUIC.

Some congestion controllers can benefit from frequent feedback during an initial slow start period, where the sender is probing for available path capacity (see Appendix A). Since a QUIC CC is expected to work in terms of the volume of data acknowledged, rather than the number of ACKs received, this is not expected to be an issue.

Consider a burst of 10 packets sent over a forward path. If the path RTT is shorter than the ACK Delay value, then all packets are cumulatively acknowledged by a single ACK. This is therefore the ACK pattern with an ACK Ratio of 1:10, where 10 packets are sent as a burst: 1 2 ... 10.

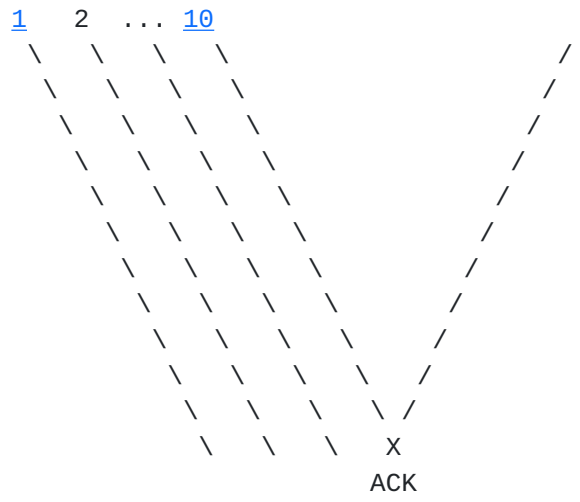

Figure B1: Burst; ACK Delay 25ms; RTT 10ms; ACK Ratio 1:10.

The single ACK both cumulatively acknowledges all the data, and increases the cwnd corresponding to reception of the 10 packets. An ACK policy that generates more frequent ACKs (e.g., a lower ACK Delay or ACK Ratio 1:1 or 1:2, would send multiple ACKs when receiving the 10 packets, the duration of the ACK burst is the same as the duration of the transmission burst: 1 2 ... 10.

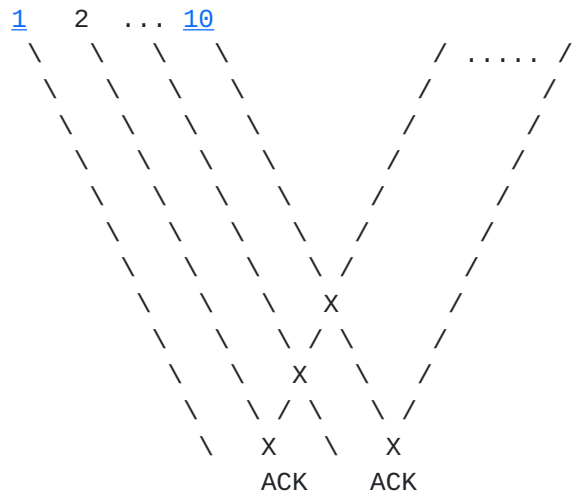

Figure : Burst; ACK Delay 25ms; RTT 10ms; ACK Ratio 1:2.

The result of using this ACK policy is that it reduces the time to the first ACK by about the burst size. This is not usually a significant benefit, because often the RTT is greater than the burst size. A similar result occurs when ACKs are generated by either the ACK Ratio or the ACK Delay timer. This also has an advantage that it generates more than one ACK, preventing progress being a hostage to fortune on a single ACK. QUIC recommends pacing. If the sender were

to pace the 10 packets over the RTT, and the receiver ACK Ratio was 1:1, then this is the ACK pattern for the 10 packets paced over the RTT:

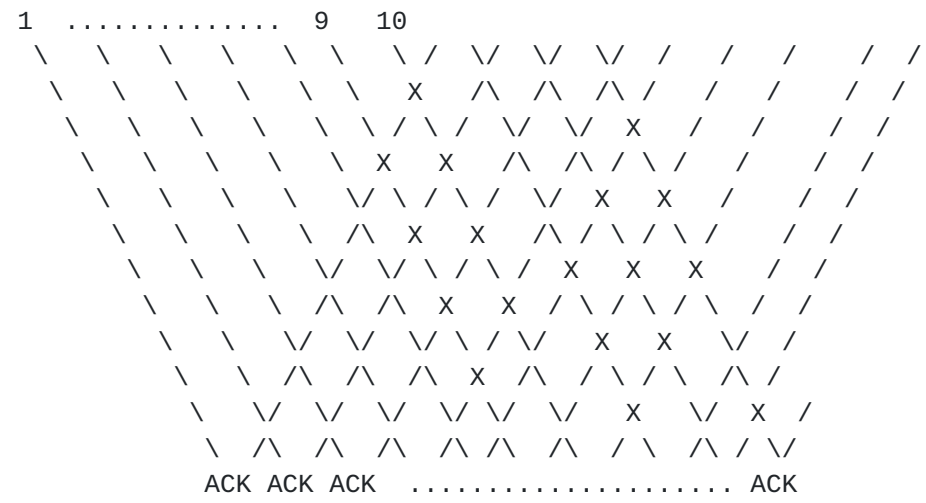

Figure B2: Pacing; ACK Delay not relevant; RTT 10ms; ACK Ratio 1:1.

The time to receive the first ACK is similar to that without pacing, and all later ACKs arrive paced. For the same path (RTT shorter than the ACK delay), but with the ACK Ratio set to 1:10, then the pattern for 10 packets paced over an RTT is:

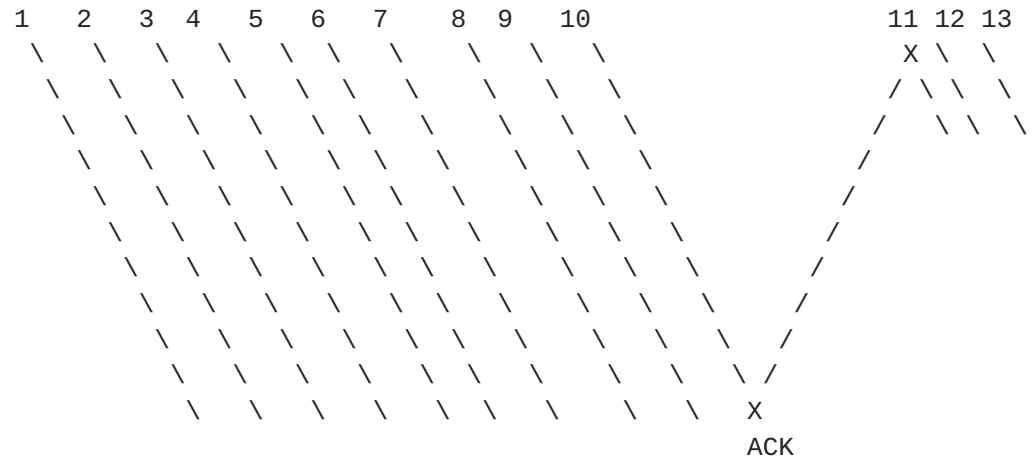

Figure B3: Pacing; ACK Delay 25ms; RTT 25ms; ACK Ratio 1;10.

The effect is similar to a very large ACK Delay. The time to receive the first ACK is increased by an RTT (because that was the pacing target). This causes the sender to wait an additional RTT (during which the sender becomes idle) before it receives an ACK for the series of 10 packets, and resumes pacing new data (at the new higher rate, following the ACK). Because this increases the time for

receiver to send the first ACK, it therefore slows the window growth. If the path is much longer than the ACK delay, this pathology does not occur. The patterns of ACKs is primarily the result of the ACK Delay setting. The first ACK is returned after waiting for the ACK Delay, and arrives shortly after one RTT:

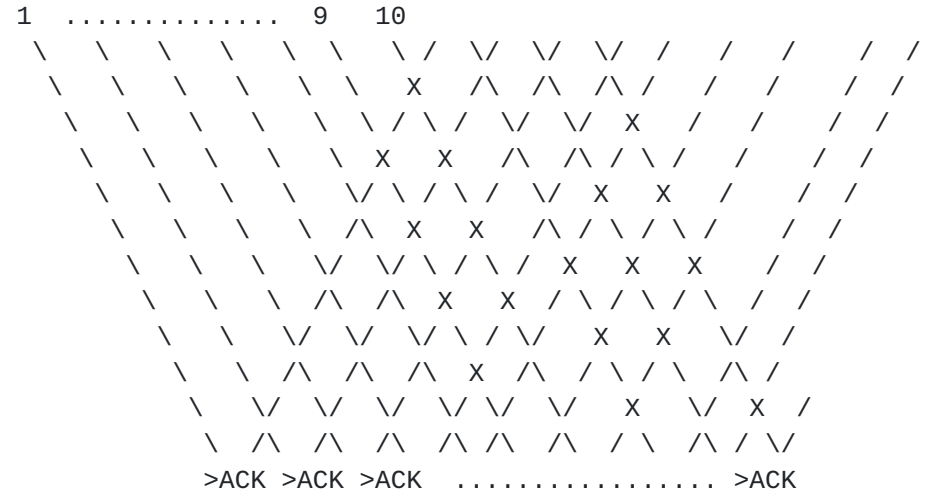

Figure B4: Pacing; ACK Delay 25ms; RTT 650ms; ACK Ratio - not relevant.

The window growth is slowed only by the additional time of the ACK delay period. In this case the role of the ACK Ratio becomes apparent only as the rate increases so that multiple ACKs are received with the ACK Delay interval.

Although QUIC uses cumulative ACKs to inflate the cwnd, and does not rely upon ACK-clocking to time the transmission of new packets, it is still influenced by the rate of ACKs in the time it is building the congestion window. This could be mitigated by choosing an appropriate ACK Delay, relative to the RTT, but the RTT is often unknown at the time when ACK Delay is negotiated. Another way to mitigate this is to ensure sufficient ACKs are sent for the first "n" packets.

For example, setting the ACK Ratio as 1:2 for the first 100 received packets. A value of 1:2 is expected to be a reasonable compromise between reducing delay and conserving return path capacity, since QUIC uses volume-based accounting to increase cwnd, it does not need to ACK each received packet as in DAAS.

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

We used an experimental approach to examine a change to QUIC's ACK Policy <http://erg.abdn.ac.uk/~downloads/ackscaling.pdf>. This section provides a few of the many results. These experiments were performed in January 2020 based on available implementations at that time.

Our tests show that the proposed change to the ACK Ratio did not negatively impact the protocol. It reduced the amount of IP, UDP and ACK overhead by a factor of approximately 5. The implemented congestion control, was also not negatively impacted. 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.

Additional data and analysis is provided in [Cust20].

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