Changing QUIC default to ACK 1:10

(Updating QUIC ACKs to avoid penalising asymmetric paths)

Gorry Fairhurst
Ana Custura
(University of Aberdeen)

Summary

The current QUIC transport specifies a default ACK Ratio of 1:2.

The ***baseline performance*** needs to be at least as good as for TCP

- We propose and implemented a change to the default ACK Ratio to be based on an ACK Ratio of 1:10.
- This change means that QUIC will work significantly better over many current Internet paths that have asymmetry.

This ***does not*** preclude implementations allowing a sender to use a higher or lower ACK Ratio for a connection, or varying this to meet the needs of a congestion-controller or capacity-probing technique.

Testbed

Endpoints:

- Linux TCP
- Quicly, draft revision 27
- Chromium, draft revision 26,
- PicoQUIC, draft revision 26.

FreeBSD router to emulate path delay of 600ms

When required, traffic shaping emulates a 1% packet loss for forward path

Experiments transferred 10MB of data on forward path, client to server

Network traces and logs collected and stored for analysis.

Why 1:10 for QUIC?

QUIC isn’t the same as TCP

- QUIC does not rely on ACKs for ACK-Clocking
- QUIC doesn’t block connection on packet loss
- QUIC retransmissions can use the PTO (mostly)

However, still needs an RTT estimate … at least 1/4 RTT

See: issue #3529

Implementation Testing

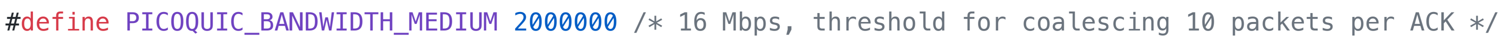
ACK Decimation is already implemented and on by default in Chromium, since February 2020.

The ACK Ratio starts as 1:2 and becomes 1:10 after 100 packets, bound by RTT.

quickly defined this as a constant (NUM_PACKETS_BEFORE_ACK=2), which we changed to 10 for the experiments. Since the 10th of April, Quicly can update the ACK frequency ratio using a transport frame - still only one line of code to change in the code.

PicoQUIC already implements an algorithm for calculating ACK frequency ratio locally.

Currently sets to 1:10 if the data rate is greater than 16Mbps and the RTT>100ms, 1:4 if the data rate is greater than 16Mbps and RTT<100ms, and 1:2 otherwise.



```
#define PICOQUIC_BANDWIDTH_MEDIUM 2000000 /* 16 Mbps, threshold for coalescing 10 packets per ACK */
```

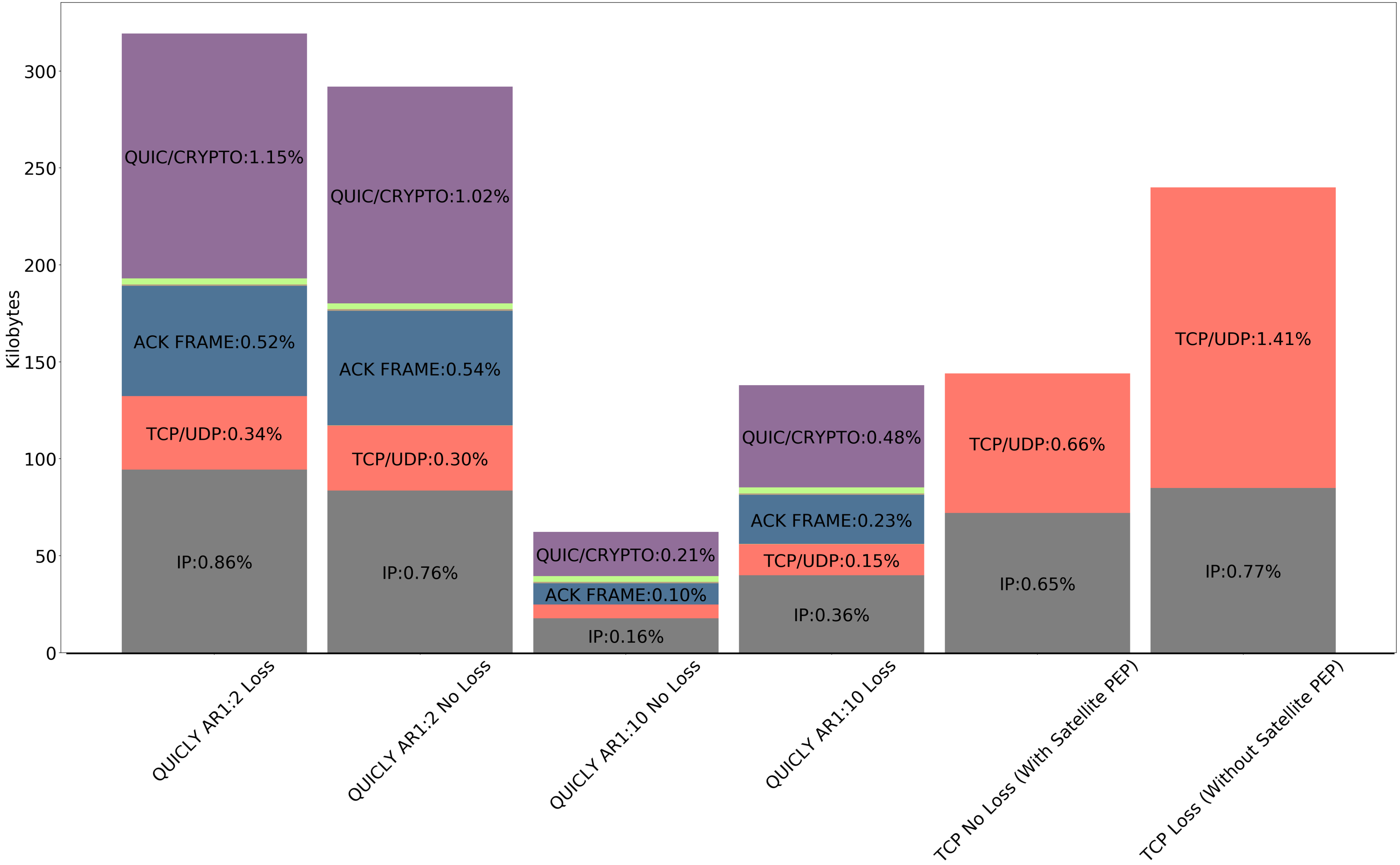
Implementations need only small changes to set a new default ACK policy…

Experimental Scenarios

Case	Download Path (Mbps)	Upload Path* (Mbps)	Loss
Small public satellite broadband access	10	2	None
Medium public satellite broadband access	50	10	None
Loss-free Large public satellite broadband access	250	3	None
Lossy Large public satellite broadband access	250	3	1%

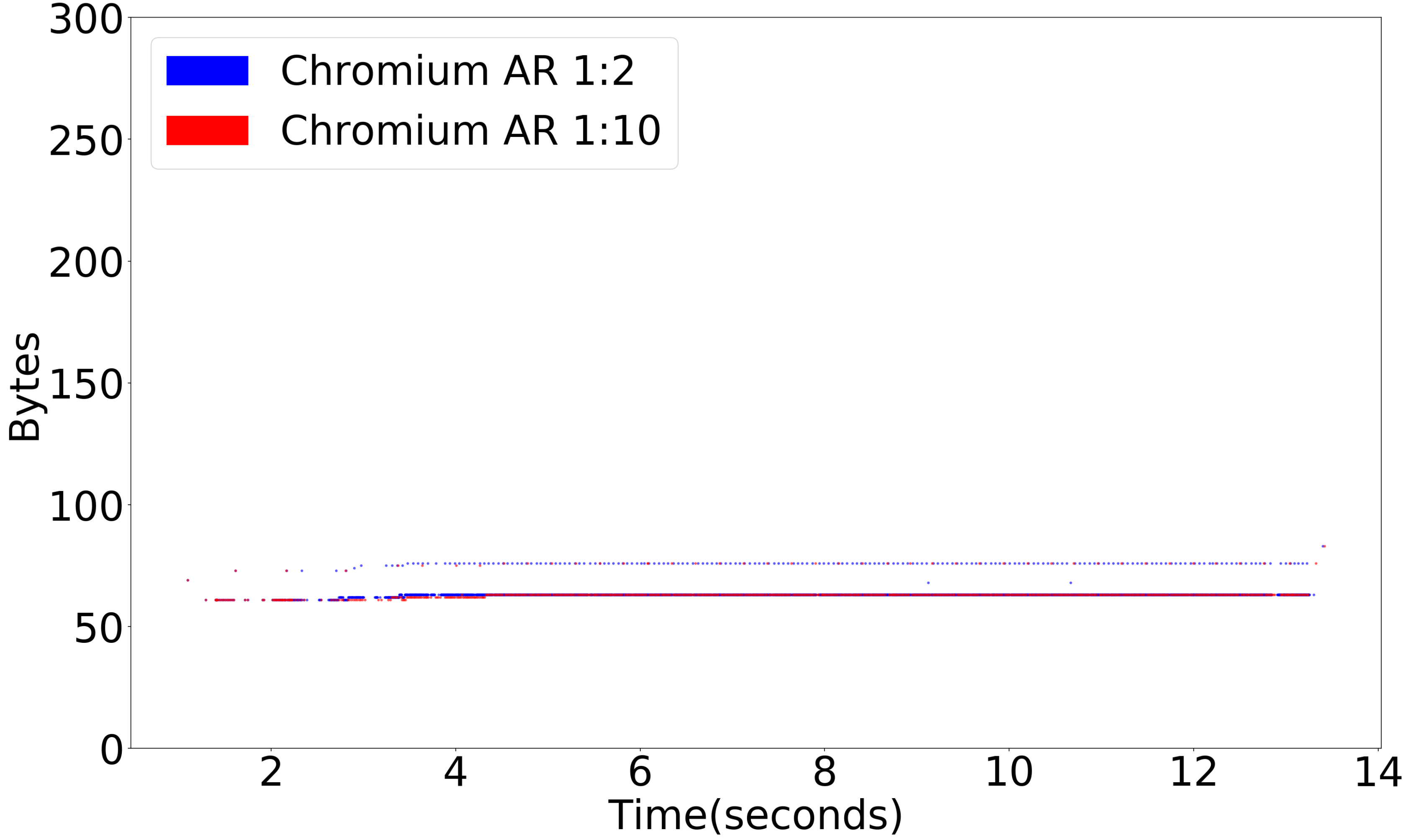
*Path characteristics under ideal “clear sky” conditions.
(e.g., deep fade will reduce by factor of 4-8)

Analysis of Return Traffic



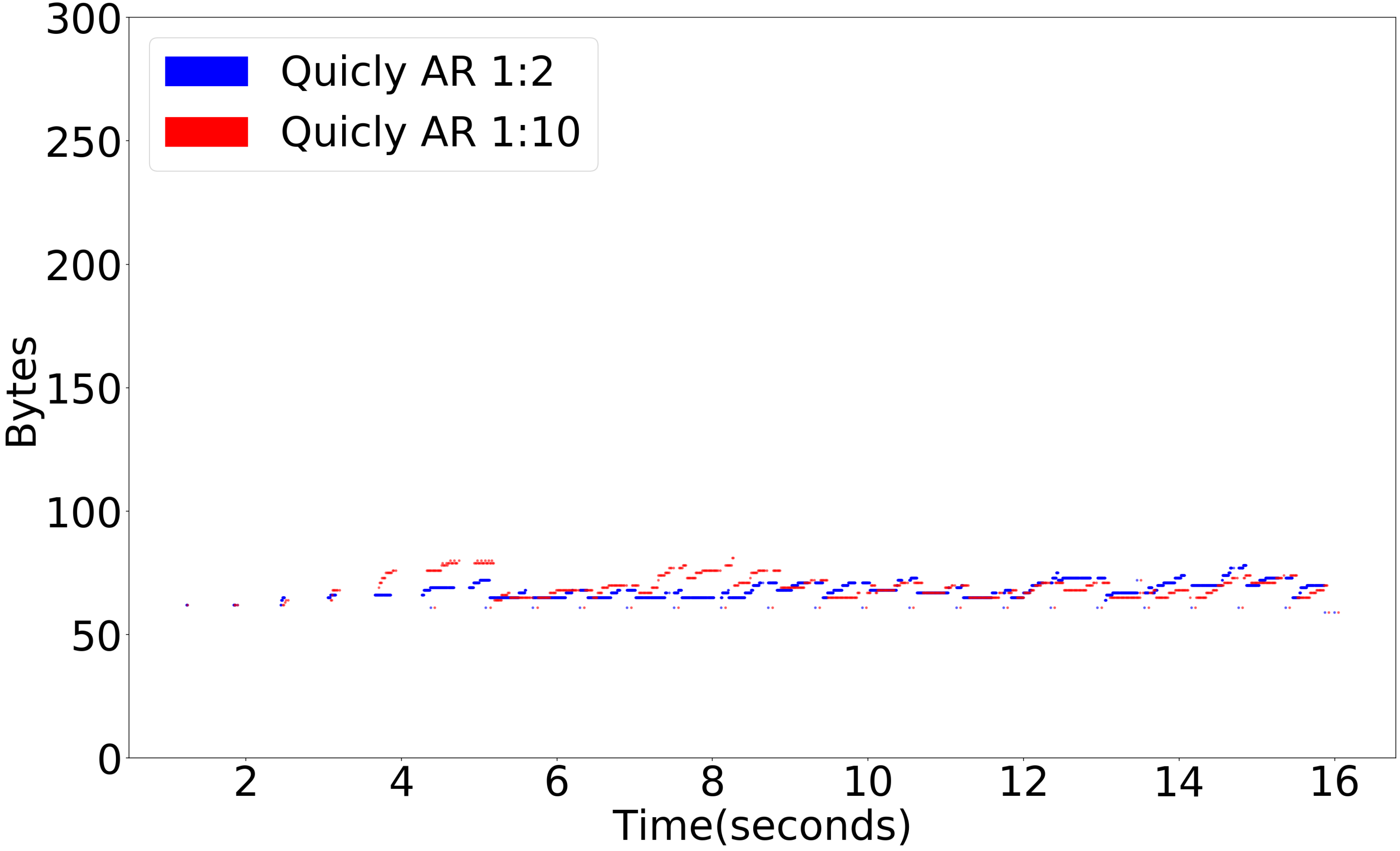
Volume of ACKs measured for a 10MB transfer, with and without link loss, emulated 600ms Path RTT.

ACK Ratio 1:10
Return Path ACK Traffic



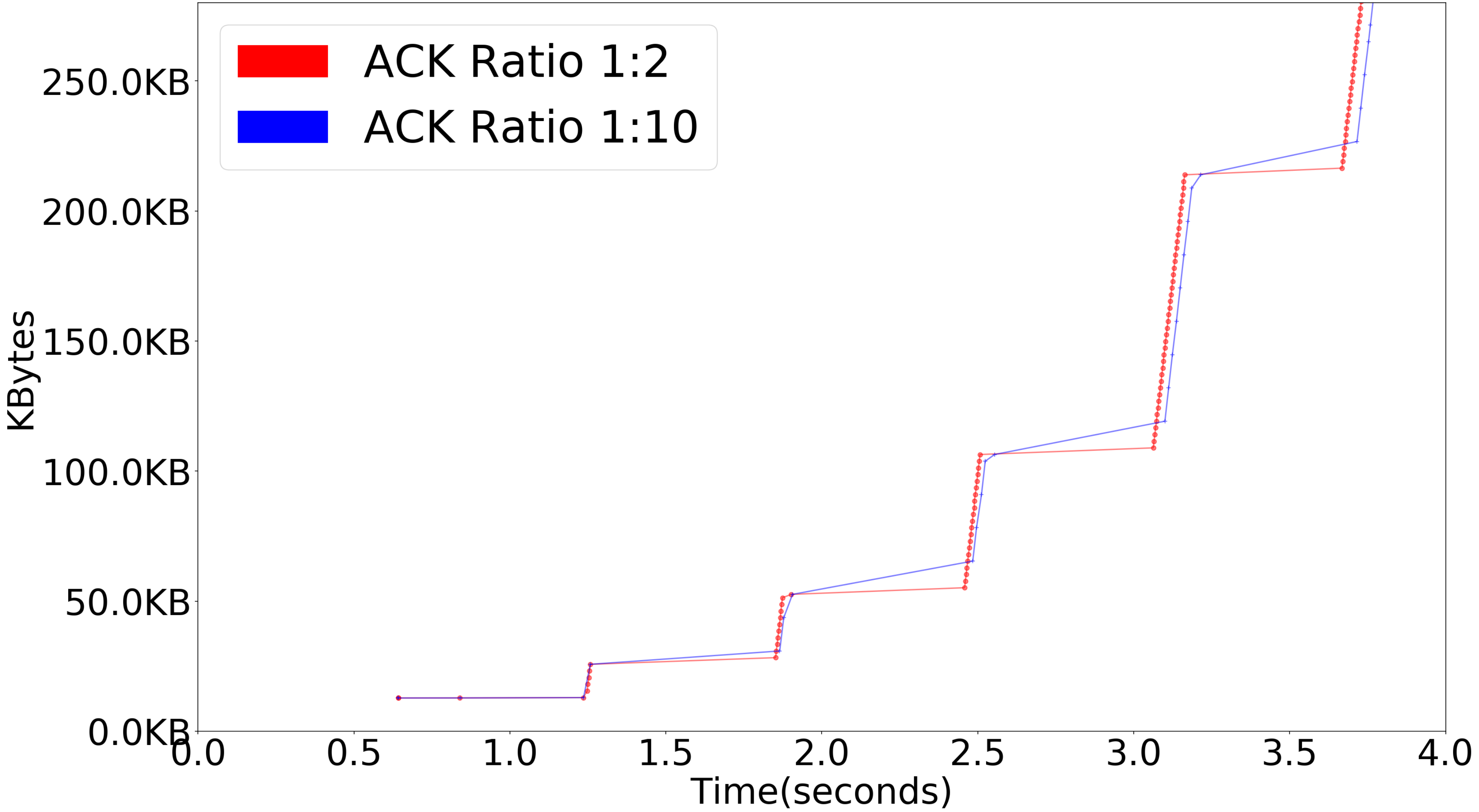
ACKs/time measured for a 10MB transfer,
with no link loss, emulated 600ms Path RTT, using Chromium

ACK Ratio 1:10
Return Path ACK Traffic



ACKs/time measured for a 10MB transfer,
with no link loss, emulated 600ms Path RTT, using Quickly

ACK Ratio 1:10 did not significantly impact performance



Congestion window measured for a 10MB transfer,
with no link loss, emulated 600ms Path RTT, using quicly

What is the impact of path RTT?

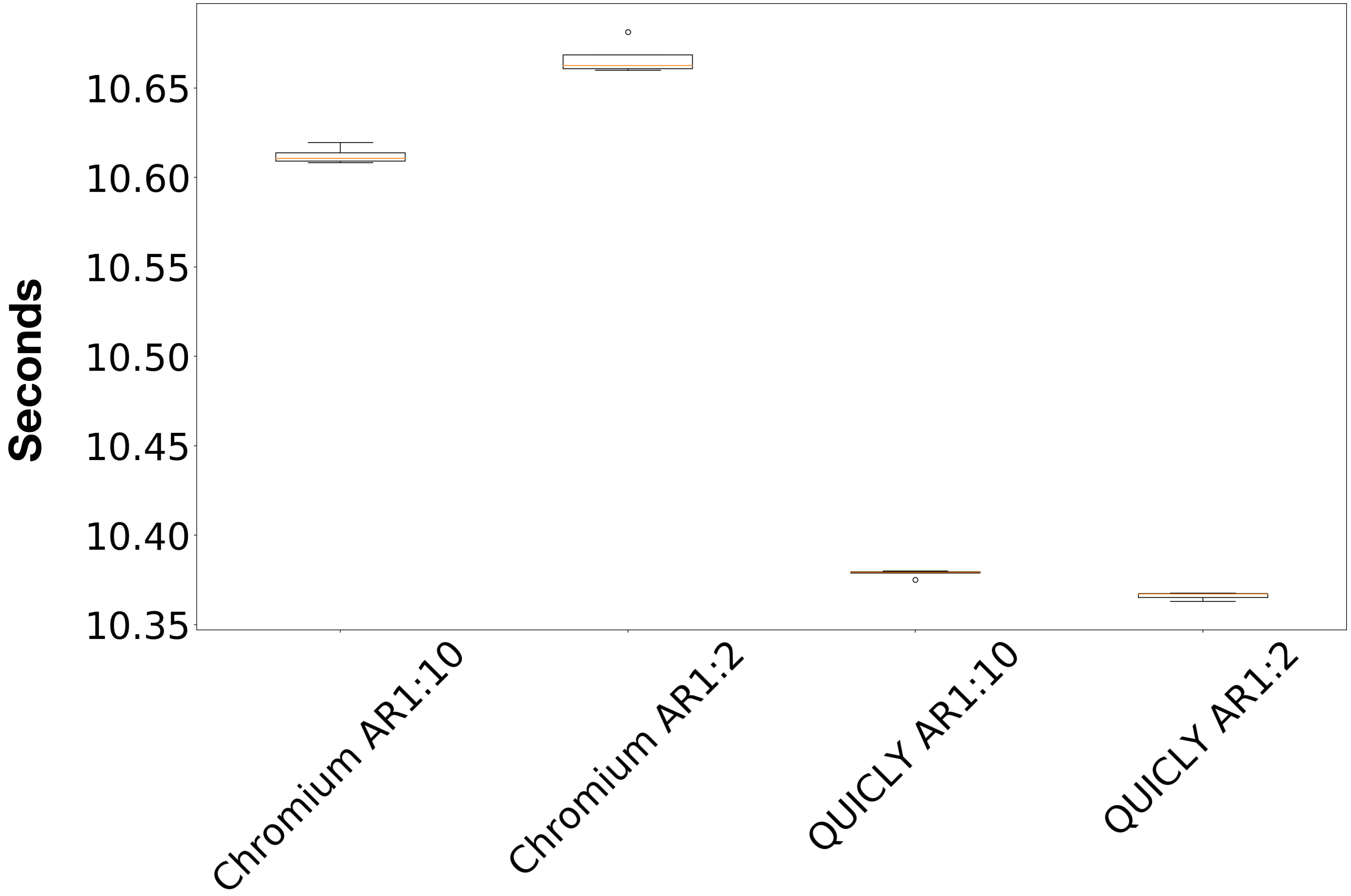
cwnd growth depends on receiving ACKs to know the cwnd was “safe”.

- The final packets in each round of growth is “delayed” by ACK delay (more significant when there is pacing?)
- This was a motivation for DAASS in TCP, and applies also to QUIC
- An ACK Ratio of 1:10 means ***more ACKs*** would be subject to this delay

We recommend keeping an ACK Ratio of 1:2 for the first 100 received packets.

- Effect was not discernible for a RTT >> 25 ms (the default ACK_Delay).
- The rule will have benefit for path with a lower RTT

ACK Ratio 1:10 did not significantly impact performance for a path with a 20ms RTT



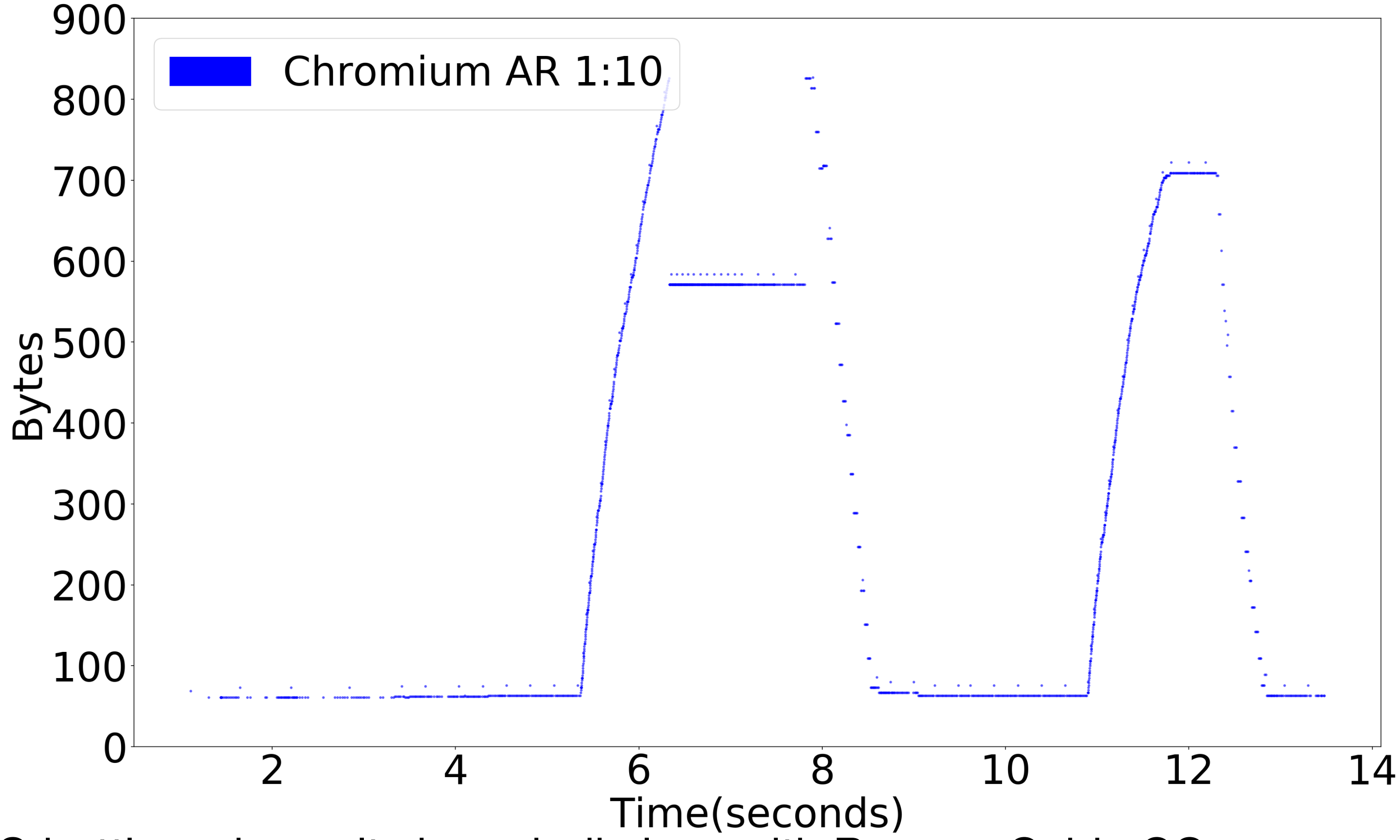
Time to download 10MB, emulated 20ms Path RTT, 8.5Mbps/1.5Mbps, n=6 transfers
Note: Chromium ACKs the first 100 packets, Quickly does not

	10/2 Mbps	50/10 Mbps	250/3 Mbps
TCP - no loss *	133 - 346 kbps	650 -1,730 kbps	3,250 - 8,650 kbps
QUIC - 1:2 ACK ratio no loss	144 -438 kbps	720 - 2,190 kbps	3,600 - 10,950 kbps
QUIC - 1:10 ACK ratio no loss	28.8 - 87.6 kbps	144 - 438 kbps	720 - 2,190 kbps

* Rate generated at receiver, with no ACK-Thinning or PEP

Rate of ACK bytes required to fill the forward path.
Cases where just ACKs would consume ***full return capacity*** are highlighted in red

QUIC Variability due to loss scenarios (Chromium)



A FIFO bottleneck results in periodic loss with Reno or Cubic CC

ACKs grow (due to aCK Ranges) up to 900B in size

Proposed Method

Main proposed change:

- Default ACK Ratio becomes 1:10
- Condition 1: Maintain ACK 1:2 for first 100 received packets (DAASS)
- Condition 2: Always ACK at least 4/RTT

Other Good practice:

- Recommendation 1: Don’t over-size the ACK Range limit
- Recommendation 2: Drop ACK Ranges promptly (e.g. Issue #3581)
- Recommendation 2: Re-consider ACK each packet for 1/8 RTT after reordering

See: draft-fairhurst-quic-ack-scaling

What about a still higher ACK Ratios after connection establishment?

- 1:10 is in line with IW, and pacing designed for at least this.
- Larger ACK ratios could be used for high transmission rates where it can reduce processing at the endpoints
  - >1:10 needs to consider the CC, loss recovery - can’t just use a large default without considering impact. Optimum may also be impacted by the path.
  - A method defined to support adapting connections in progress: draft-iyengar-quic-delayed-ack

Conclusion

- QUIC currently suffers performance penalties compared to TCP when used over asymmetric paths because of the larger volume of ACKs.
    - In-network TCP ACK Thinning does not help QUIC.
    - Total ACK traffic on an asymmetric link can ~x5 larger (actual impact depends on way TCP is enhanced and radio properties).
- QUIC transport needs a better default ACK Policy! (we recommend 1:10)
- QUIC connections can ***still*** adapt to allow a sender to use a higher or lower ACK Ratio for a connection, or varying this to meet the needs of a congestion-controller or capacity-probing technique