Understanding ACKs using High BDP Paths

Gorry Fairhurst
Ana Custura
Tom Jones
(University of Aberdeen)

Comparison using spreadsheet analysis and testbed evaluation of QUIC using quicly and Chromium to the performance of TCP Linux using Reno

Return paths are not all the same

Bit-Congestive Paths

- Asymmetric Available Capacity
- Contention Ratios (shared capacity pool)

Packet-Congestive Paths

- Asymmetric "Cost" of Transmission
- Return path packets simply cost more to send

How Symmetric is Traffic?

Many flows send data predominantly in one direction.

So how much ACK traffic is generated in response?

ACK Traffic as percentage of Forward Data Traffic				
AR	**1:1**	**1:2**	**1:4 (Thinned)**	**1:10**
TCP	3.1%	1.5%	0.8%	0.3%
QUIC (1200B)	5.7%	2.8%	1.4%	0.6%
QUIC (1500B)	4.7%	2.3%	1.1%	0.5%

< 1% of forward capacity

Scenarios from QUIC4SAT

Endpoints:

- Linux TCP with Reno CC
- Quicly, draft revision 27
- Chromium, draft revision 26,

Path Capacity (Forward/Return):

- 10/2 (we present data for 8.5/1.5, PRTT 600 ms)
- 10/0.1 (we present data for 8.5/1.5, PRTT 600 ms)
- 50/10 and 50/0.5 (PRTT 650 ms)

capacity 1:5

capacity 1:100

draft-kuhn-quic-4-sat-05.txt

Performance Impact on QUIC

ACK traffic limits both forward and return performance

	Forward rate (utilisation of return link in brackets)			
Path Mbps	**10/2**	**10/0.1**	**50/10**	**50/0.5**
TCP 1:1	10 (16%)	3	50 (16%)	16
TCP 1:2	10 (8%)	6	50 (8%)	32
TCP Thin4	10 (4%)	10 (80%)	50 (4%)	50 (79%)
QUIC 1:1	10 (27%)	2	35	9
QUIC 1:2	10 (14%)	4	50 (14%)	18
QUIC 1:4	10 (6%)	7	50(7%)	36
QUIC 1:10	10 (3%)	10 (49%)	50 (3%)	50 (55%)

The red background shows the Forward Rate Limit

(%age of Return Capacity)

Estimated results based on size used by Chromium with IPv4

Return Traffic Impacts System Operation

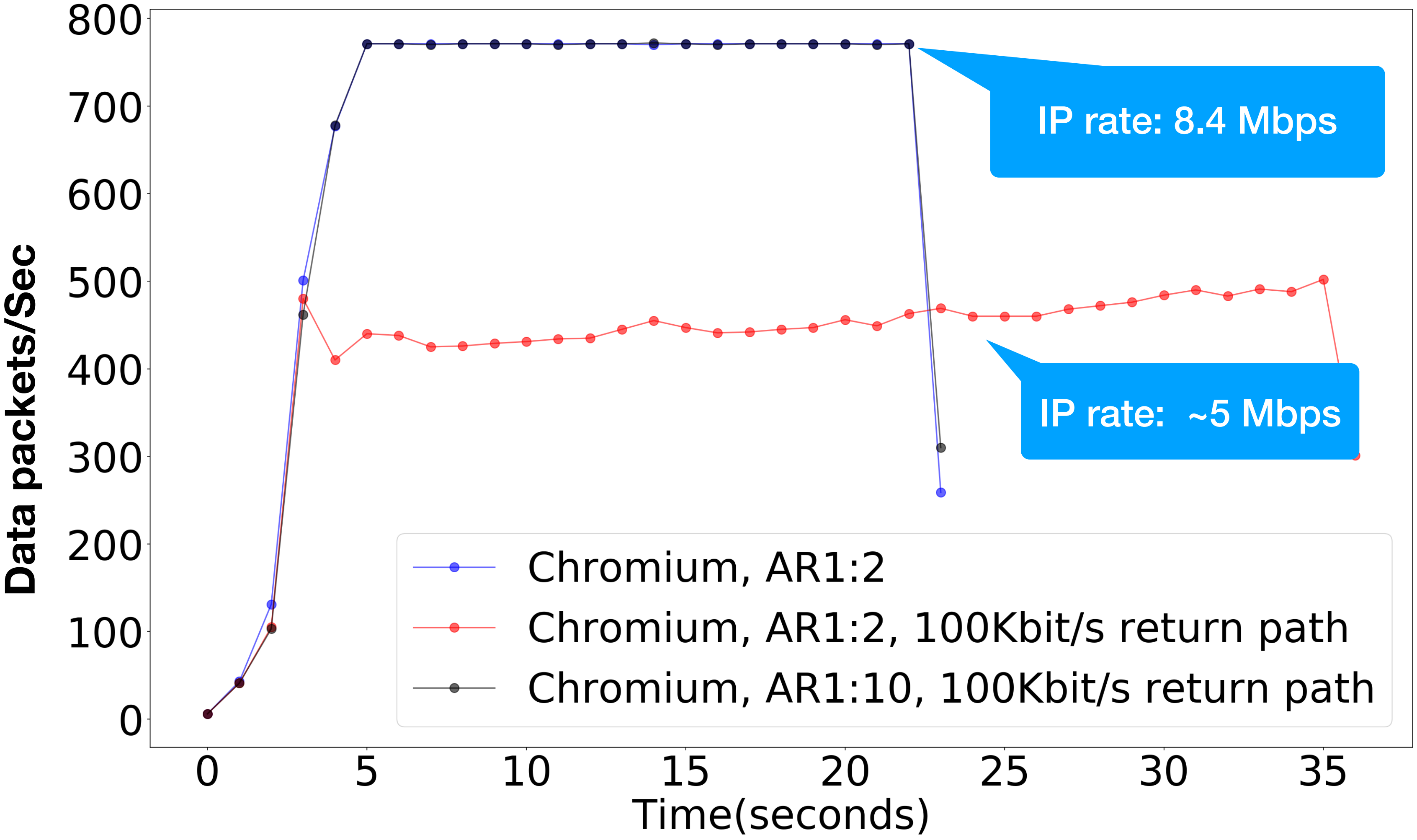
It's not good practice to fill the return path

- ACKs squeeze other traffic that share the return
- This traffic includes social media and video conferencing traffic, etc often shared by a household …an HD call using Skype can be 1.5Mbps!

Other factors also reduce available capacity:

- Return capacity allocated to others within same resource pool
- Weather, terminal design. etc … depends on use *and* location

Forward Path for 8.5/1.5



IP rate: 8.4 Mbps

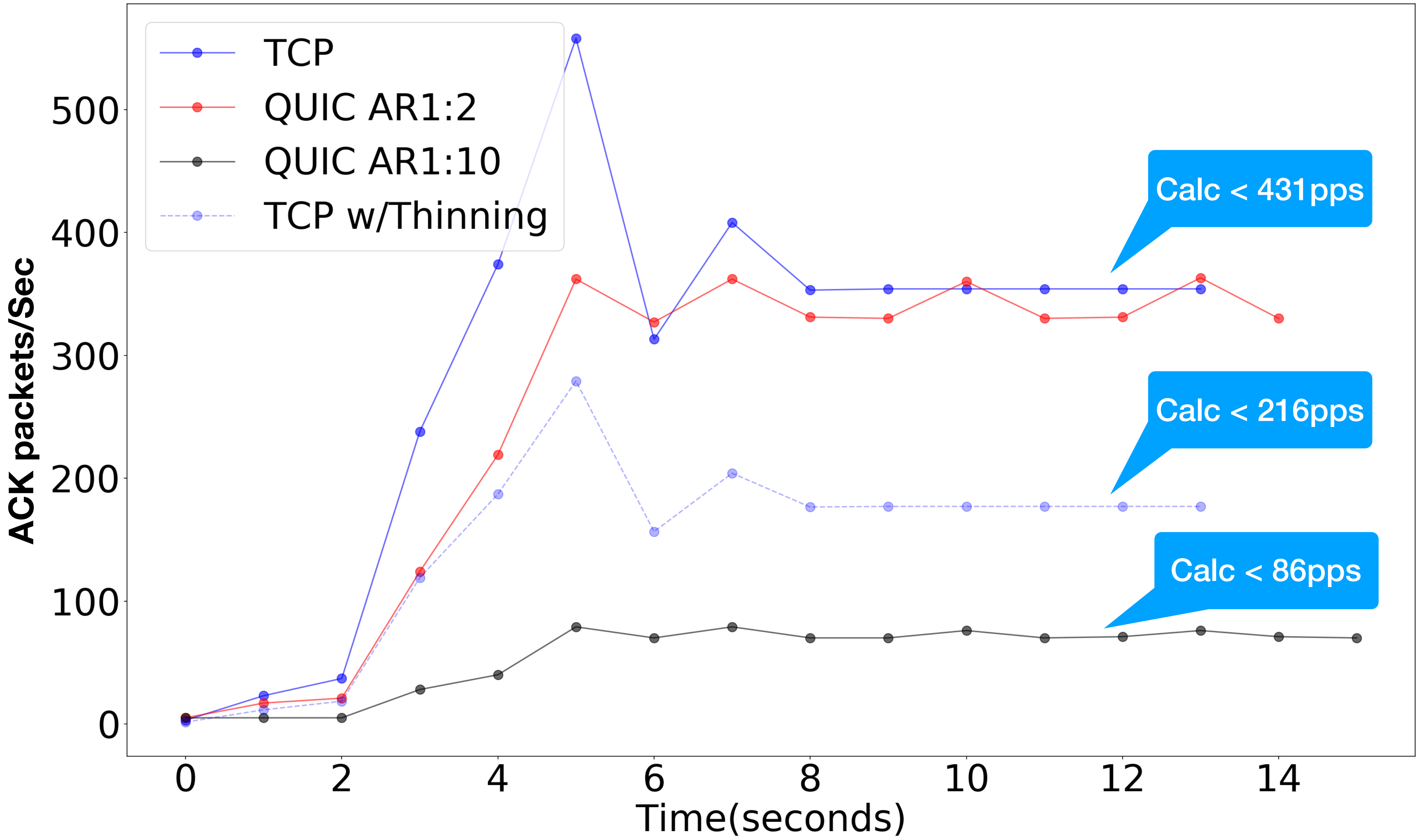
IP rate: ~5 Mbps

Measured data packets/sec using Chromium with AR1:2 and AR1:10

Return path: unconstrained (1.5 Mbps) and capacity-limited (100kbps)

QUIC payload data size was 1350B; and IP QUIC packet 1378B.

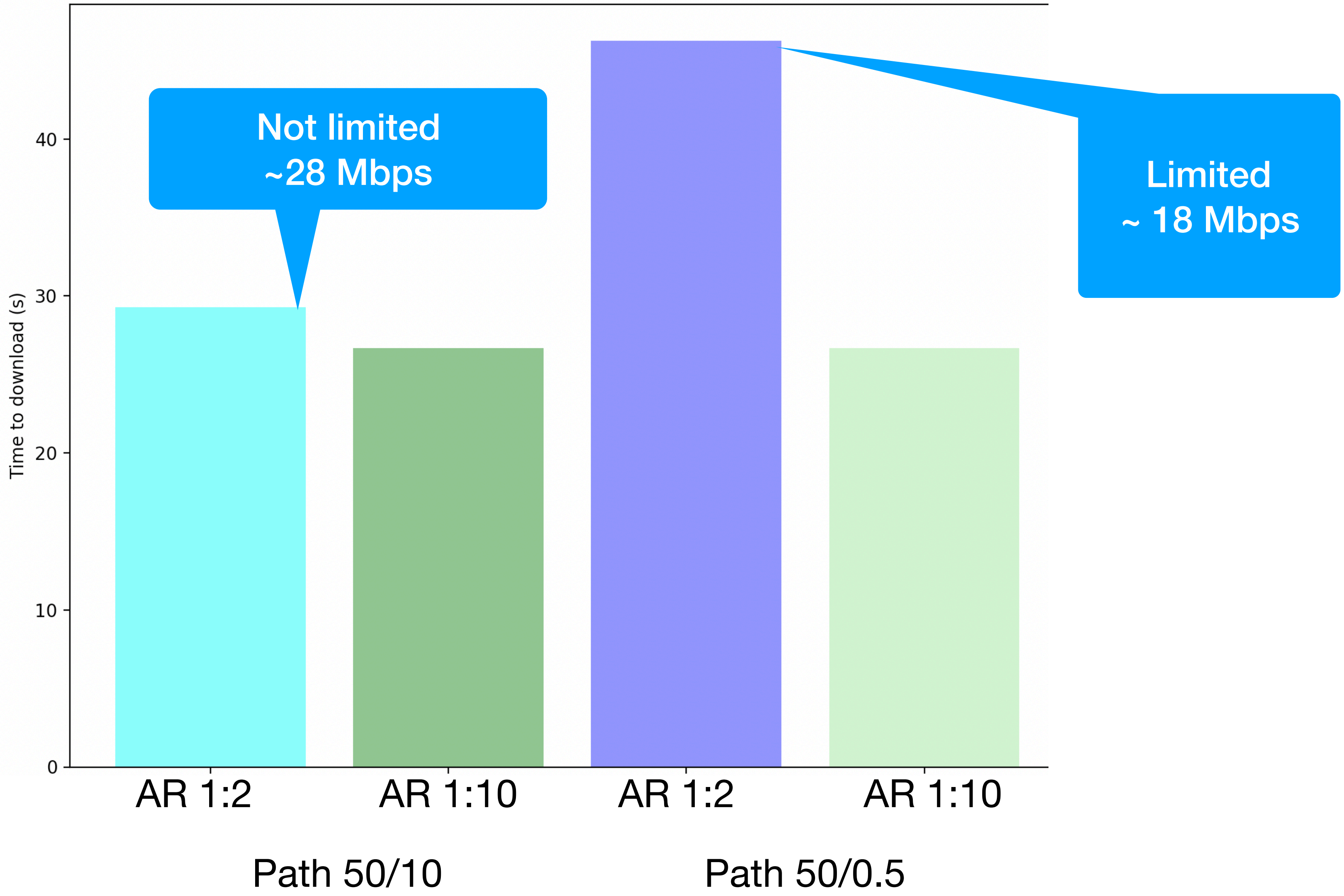
Return Path ACKs for 8.5 Mbps Forward Path



Rate of return packets/sec for Reno TCP and Quicly using AR 1:2 and 1:10

Return path: unconstrained (1.5 Mbps)

Forward Path for 50/10 and 50/0.5



Not limited
~28 Mbps

Limited
~ 18 Mbps

Measured with quickly for 100 MB transfer with AR1:2 and AR1:10 (flow credit was tuned)

Return path: unconstrained (10 Mbps) and capacity-limited (500kbps)

What about Higher ACK Ratios?

We *could do even better* knowing more about the path (e.g. params from a previous session or signals from the path).

Larger ACK Ratios may benefit high transmission rates or by specific apps and can reduce endpoint processing

>1:10 needs to consider the CC, loss recovery

- Optimum may also be impacted by the path.
- A method defined to support adapting connections in progress: draft-iyengar-quic-delayed-ack

What if QUIC ACK Traffic is still too much?

What happens if there is too much ACK traffic?

- A link can’t see inside a QUIC packet (by design)
- A queue builds, return becomes congested

Could configure a short router queue to control delay

- This would necessarily result in high ACK loss
  - Other non-ACK Frames can also be lost
- May need FQ to share capacity with other flows
  - May lead to other implications

A default AR 1:10 would avoid a lot of mess

Questions?

Please discuss more on ETOSAT mailing list

Directly Related IDs

- **draft-kuhn-quic-4-sat**
  - Characteristics of satellite that impact the operation of QUIC
  - Proposes best practice to improve performance over satellite
- **draft-kuhn-quic-0rtt-bdp**
  - Proposal for exchanging path data from sessions for use with high BDP paths
- draft-kuhn-quic-4-sat
  - Proposal to update QUIC recommendations for ACK Ratio
  - Chromium implements something like this
  - PicoQUIC has something, but not the same
  - Related discussions in QUIC about parameterised change as an extension

Spare Slides on AR 1:10

Seek a ***baseline performance*** at least as good as TCP

Current QUIC transport specifies a default ACK Ratio 1:2

AR 1:10 is in line with IW, and pacing. QUIC will work significantly better over many Internet paths with asymmetry.

Tests with AR 1:10 to examine if this negatively impacted cwnd growth.

This ***does not*** preclude implementations allowing a sender to request another ACK Ratio, or varying to meet the needs of a congestion-controller or capacity-probing technique.

What is the Impact of Path RTT?

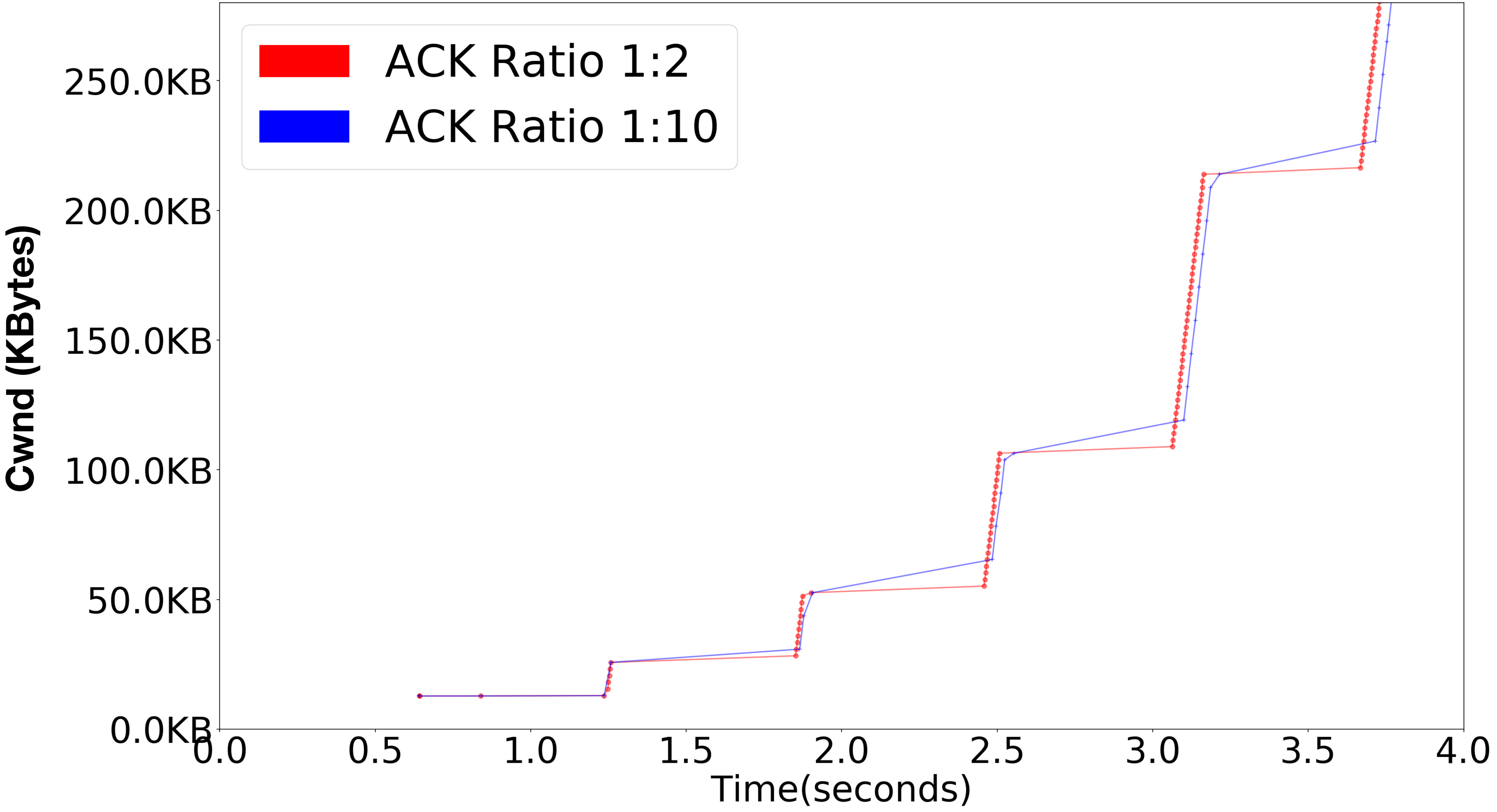
cwnd growth depends on receiving ACKs to know the cwnd was “safe”.

- The final packets in each round of growth is “delayed” by ACK delay
- 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

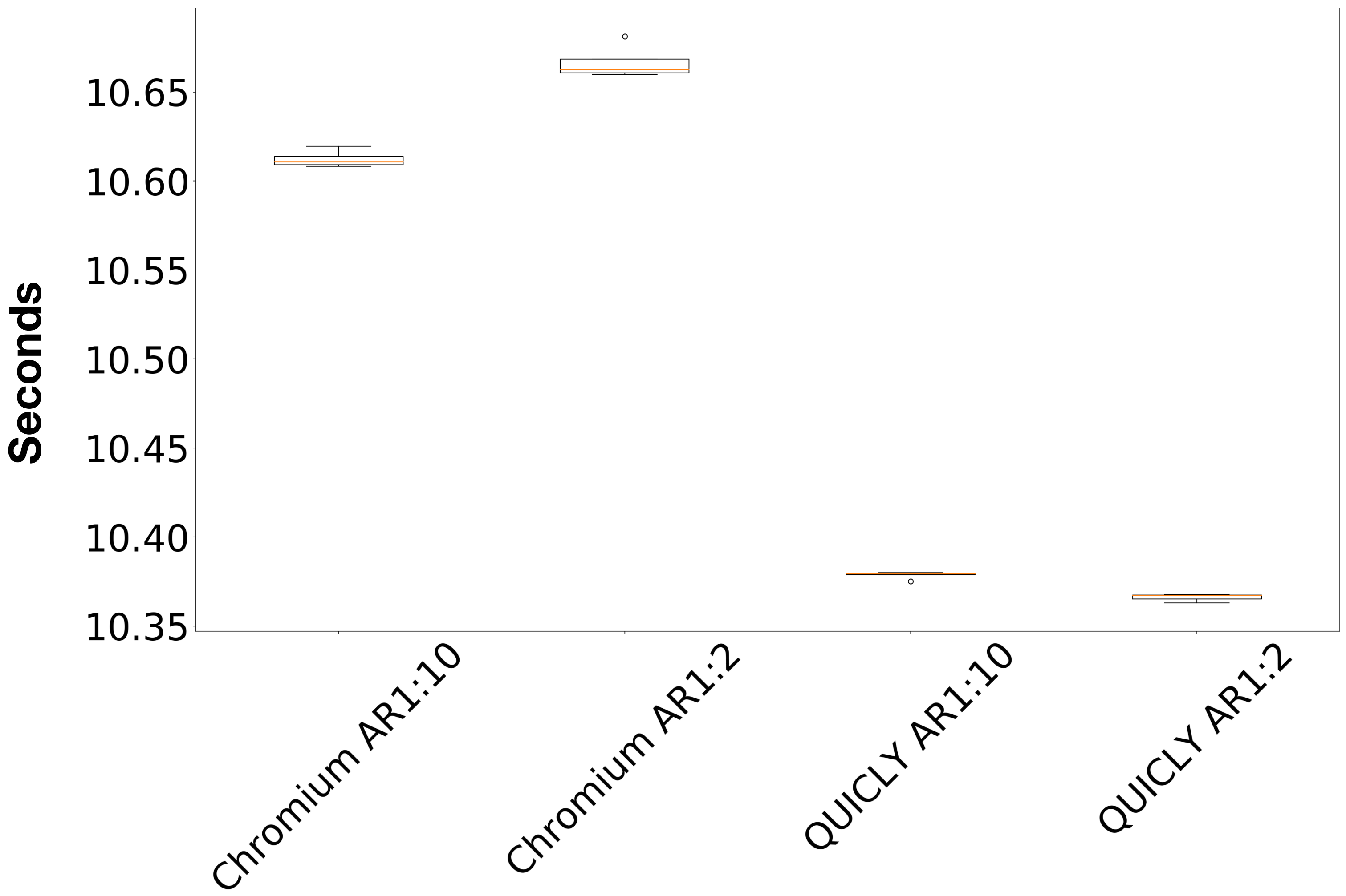
ACK Ratio 1:10 did not Negatively Impact cwnd Growth



10MB transfer, 8.5/1.5 Mbs
with no link loss, emulated 600ms Path RTT, using quicly

University of Aberdeen

ACK Ratio 1:10 did not negatively impact cwnd for a Path with a 20ms RTT



Time to download 10MB, emulated 20ms Path RTT, 8.5/1.5Mbps, n=6 transfers
Note: Chromium uses AR 1:1 for the first 100 packets, Quickly does not, however this does not impact cwnd growth when RTT > delayed ACK value

University of Aberdeen