TECHNISCHE UNIVERSITÄT DRESDEN

HAW HAMBURG

DRESDEN concept

ReACKed QUICer: Measuring the Performance of Instant Acknowledgments in QUIC Handshakes*

Jonas Mücke, Marcin Nawrocki, Raphael Hiesgen, Thomas C. Schmidt, Matthias Wählisch

{jonas.muecke, m.waehlisch}@tu-dresden.de, marcin.nawrocki@netscout.com,
{raphael.hiesgen, t.schmidt}@haw-hamburg.de

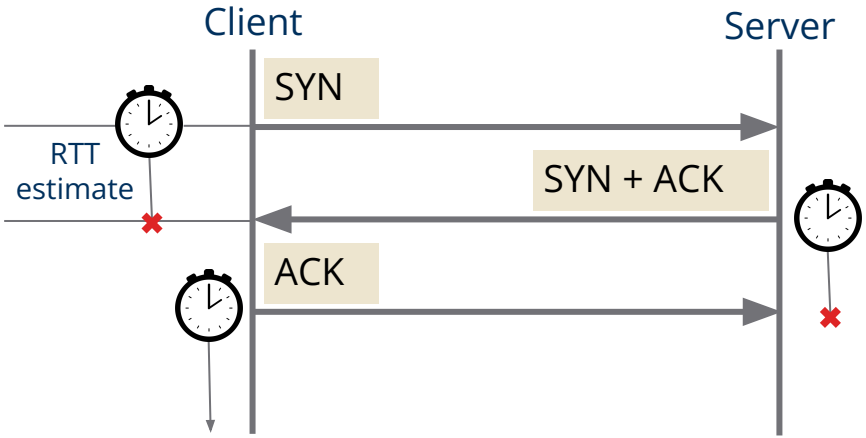
MAPRG, IETF 123 // Madrid, July 25, 2025

*originally presented at ACM IMC 2025

Timeouts are critical for reliable transport

They trigger retransmissions when they expire

TCP handshake

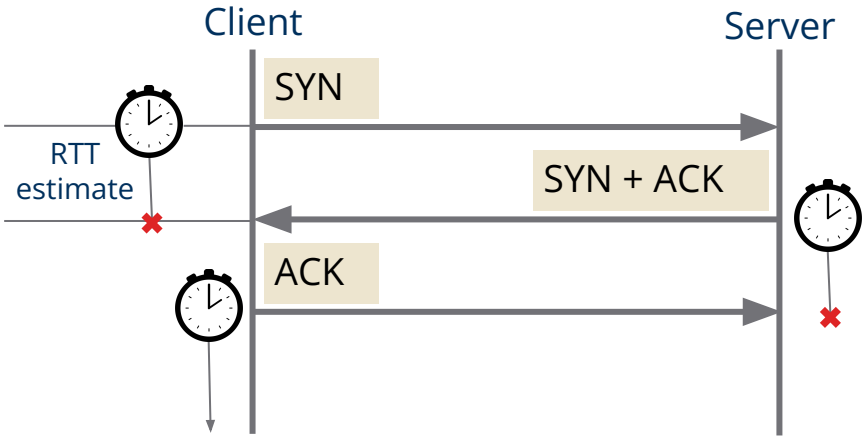


Incorrect timeouts lead to unnecessary delays or spurious retransmits.

Timeouts are critical for reliable transport

They trigger retransmissions when they expire

TCP handshake



Incorrect timeouts lead to unnecessary delays or spurious retransmits.

QUIC combines the transport and TLS handshake.

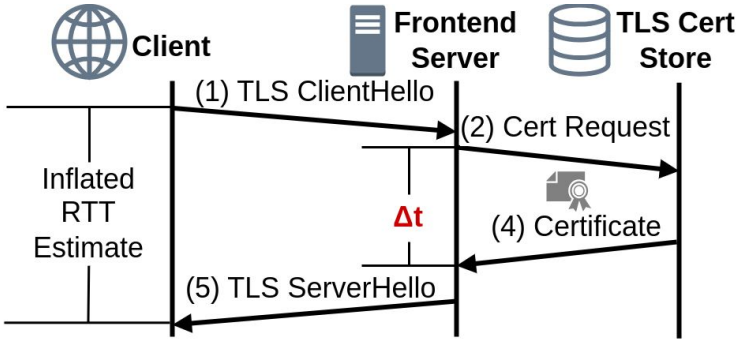
This makes it even more challenging to estimate the RTT correctly.

Precise RTT estimates lead to correct timeouts.

Distributed certificate stores challenge RTT estimate

Common CDN deployments, e.g., Cloudflare, Akamai

Inflated RTT estimate

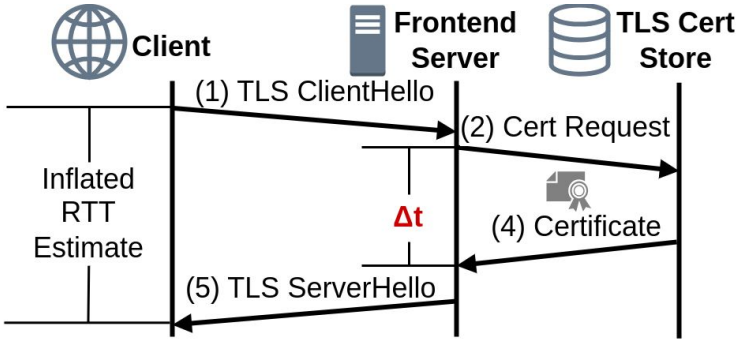


Waiting for the certificate (WFC) from a backend server inflates the client RTT estimate.

Distributed certificate stores challenge RTT estimate

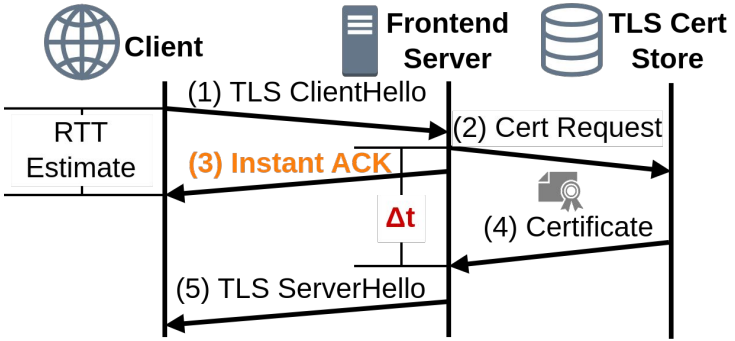
Common CDN deployments, e.g., Cloudflare, Akamai

Inflated RTT estimate



Waiting for the certificate (WFC) from a backend server inflates the client RTT estimate.

Precise RTT estimate



An **instant ACK (IACK)** reduces the RTT estimate of the client.

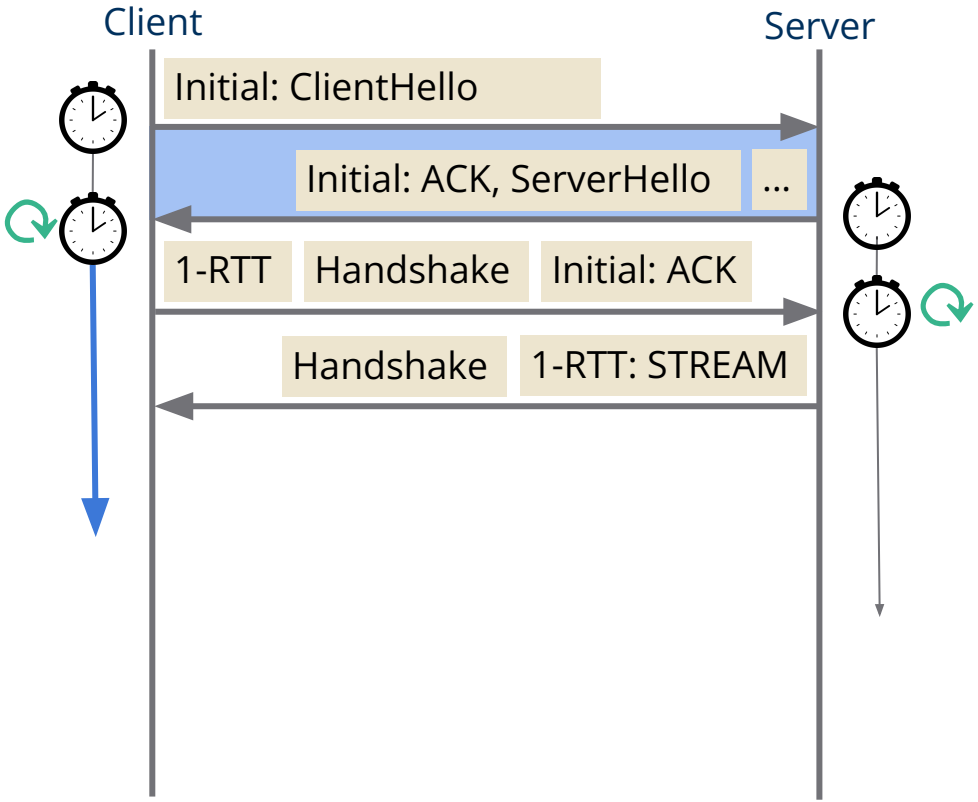
The first ACK initializes QUIC probe timeout (PTO)

Probe Timeout ⏱ determines when to resend packets in QUIC:

PTO depends on prior RTT samples

Default PTO: 1 s

Initialization: 3x FirstRTT



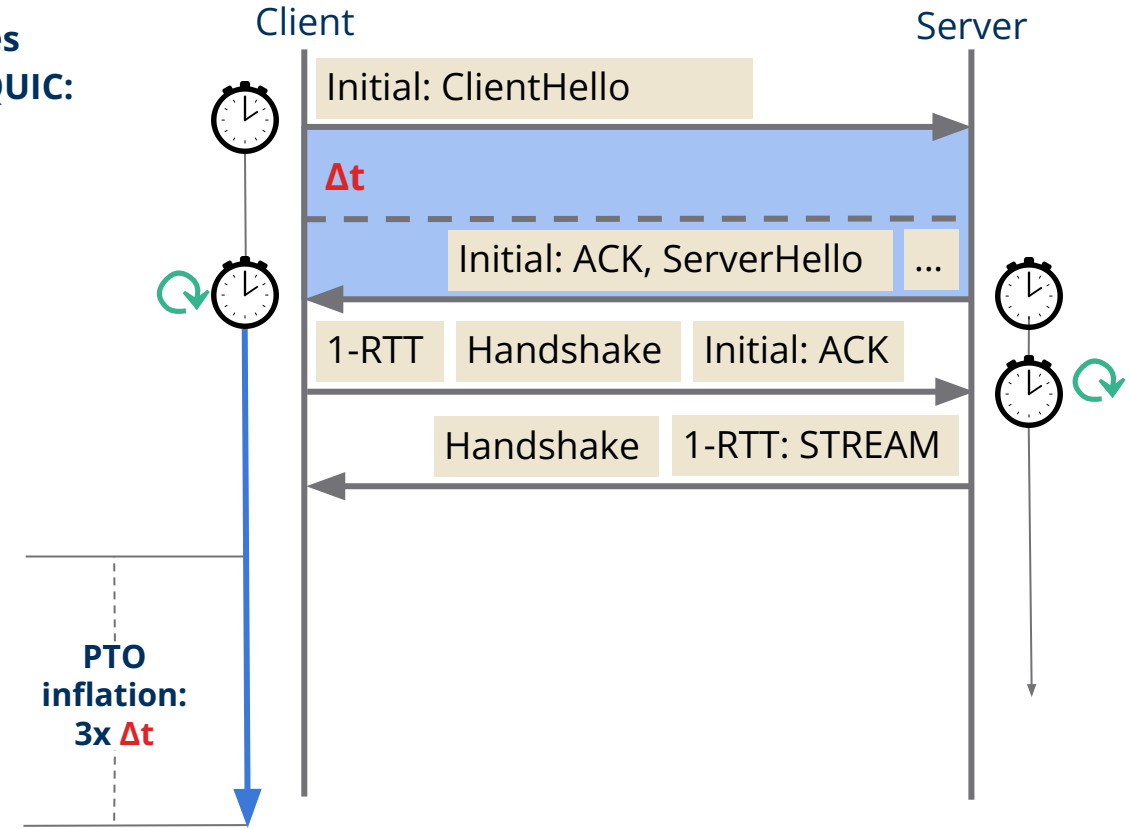
The first ACK initializes QUIC probe timeout (PTO)

Probe Timeout ⏱ determines when to resend packets in QUIC:

PTO depends on prior RTT samples

Default PTO: 1 s

Initialization: 3x FirstRTT



The first ACK initializes QUIC probe timeout (PTO)

Probe Timeout ⏱ determines when to resend packets in QUIC:

PTO depends on prior RTT samples

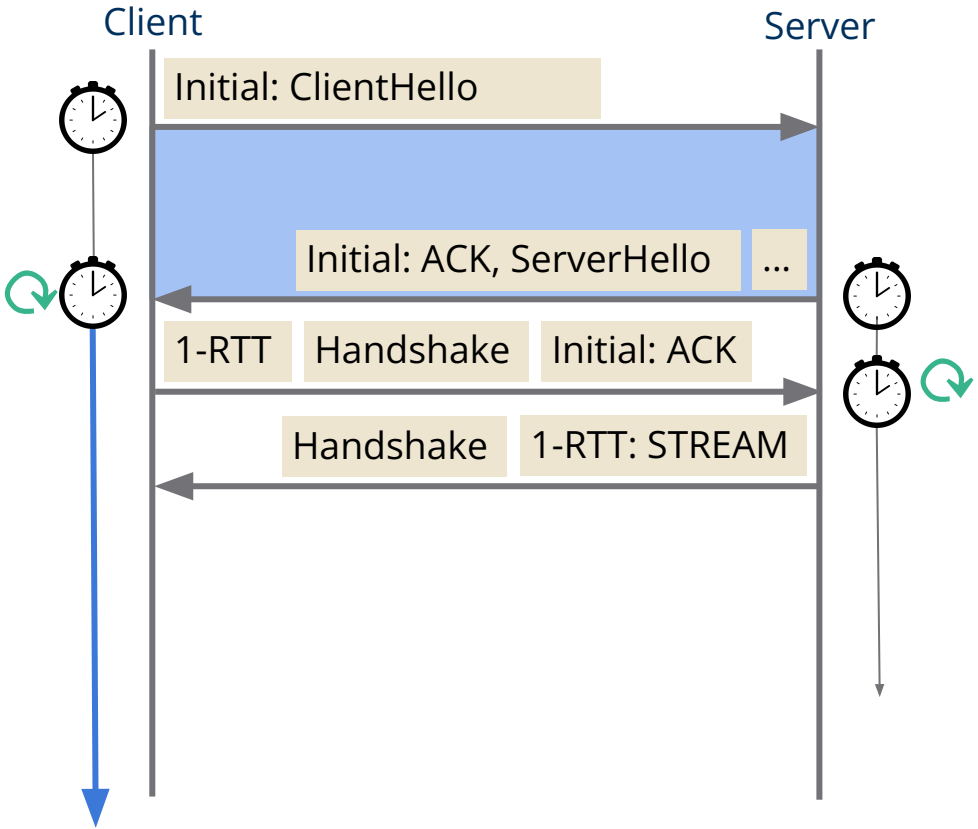
Default PTO: 1 s

Initialization: 3x FirstRTT

PTO is important:

… in case of packet loss

… to grant server sending budget



Research Questions

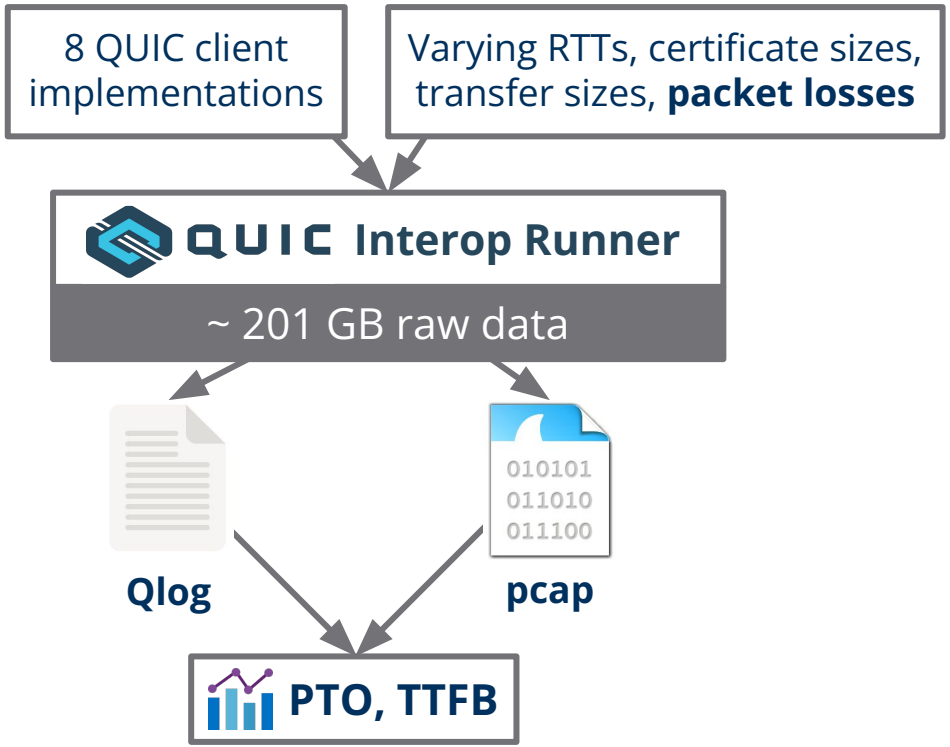
How does instant ACK influence client behavior?

When is IACK deployment beneficial, when does it harm?

How is IACK performing in CDN deployments?

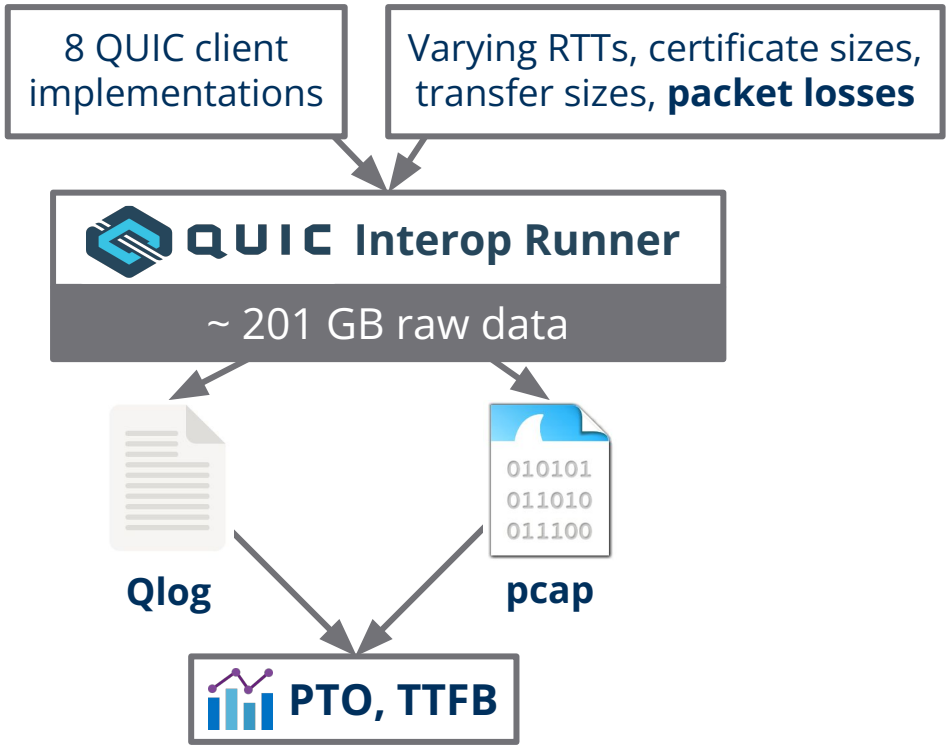
Measurement method and setups

Controlled testbed

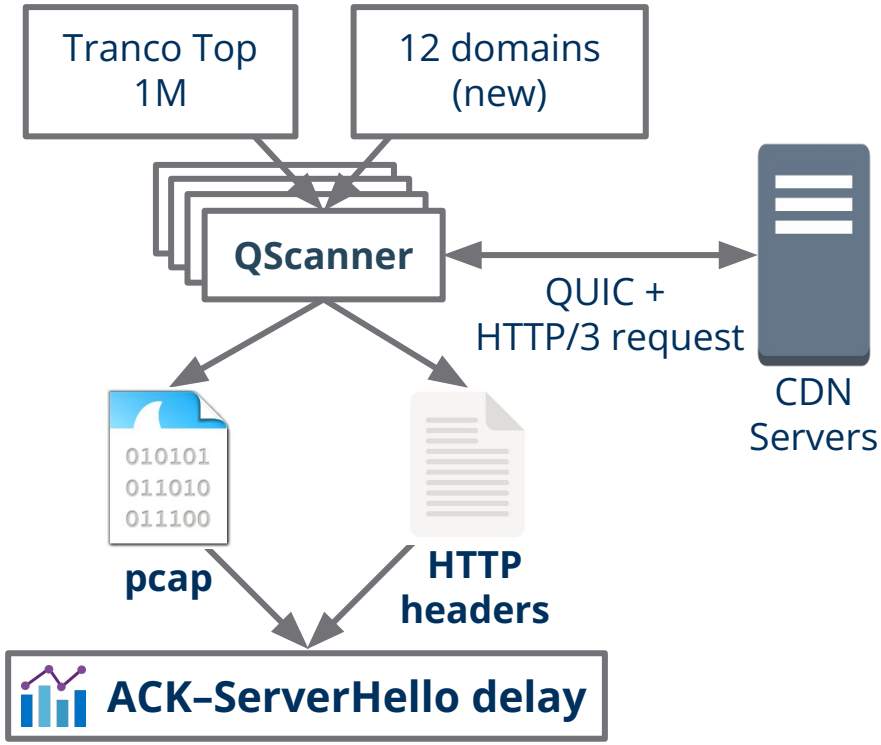


Measurement method and setups

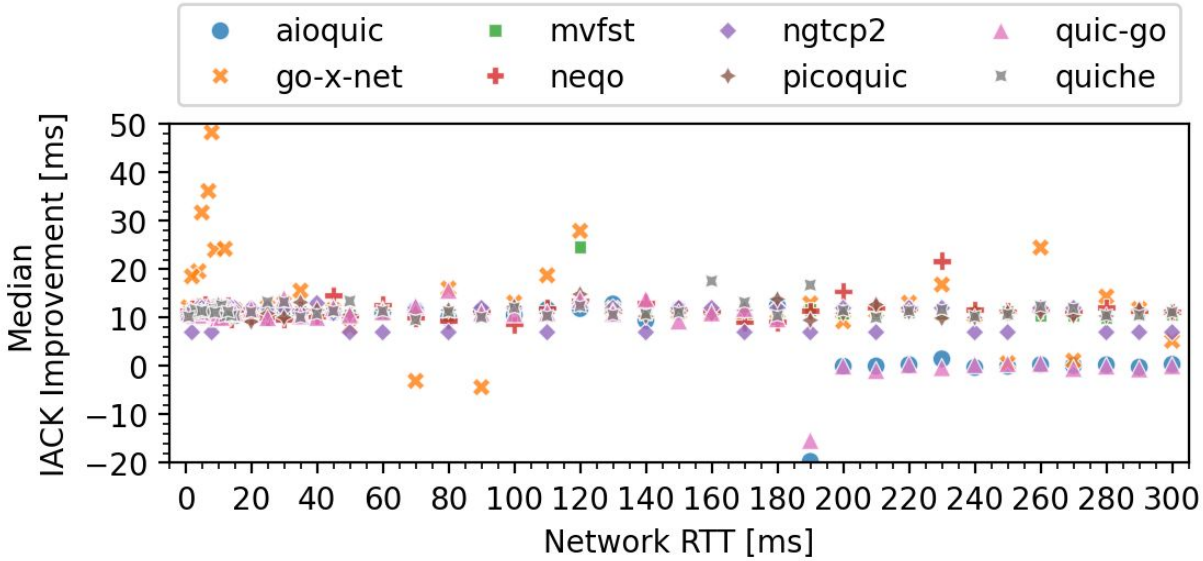
Controlled testbed



CDN deployments



Median improvement of the first PTO when IACK used

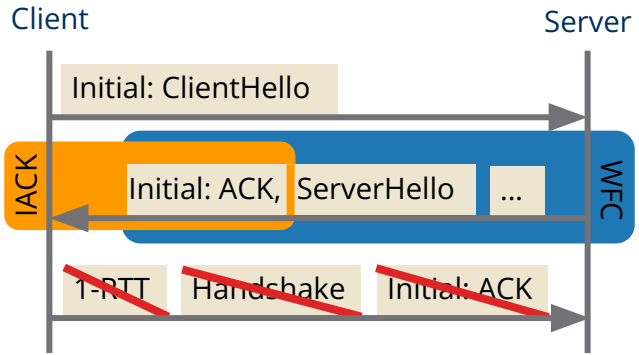


All clients benefit on median from ~10 ms shorter timeouts when **IACK** deployed.

Aioquic and quic-go use shorter default PTOs, benefiting relatively less from **IACK**.

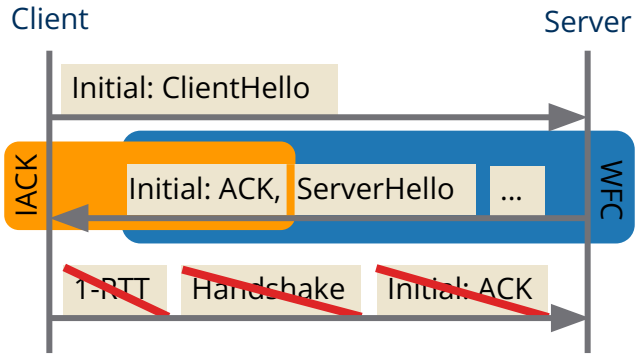
Emulation of packet loss

Loss from client → server

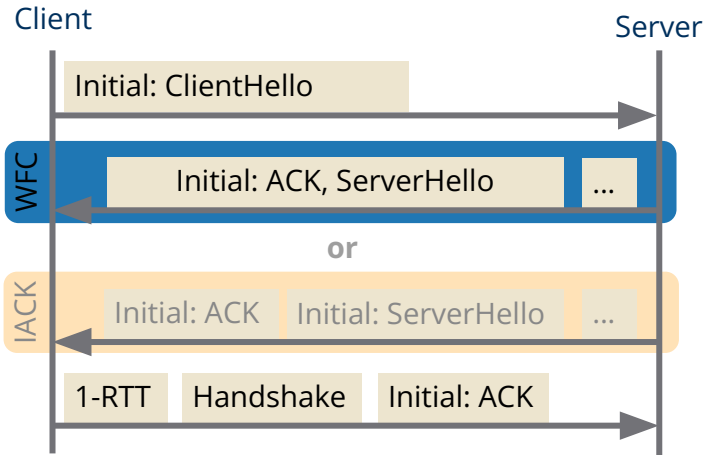


Emulation of packet loss

Loss from client → server

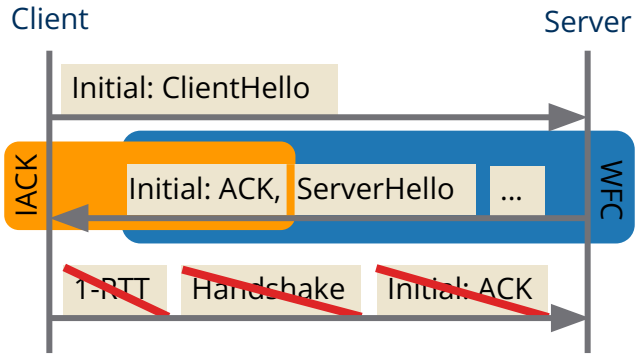


Loss from server → client

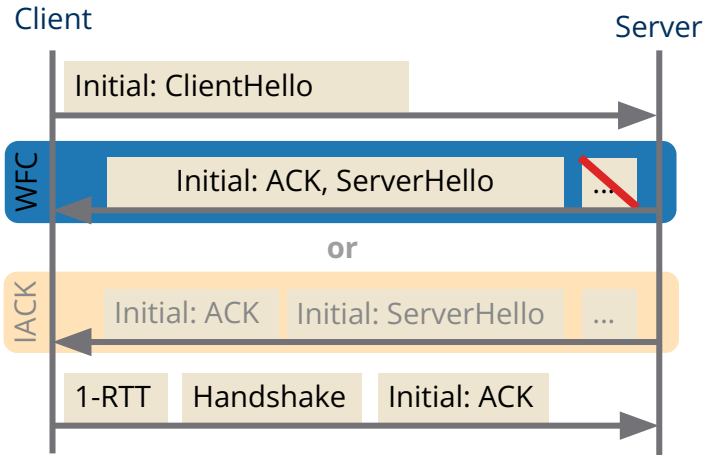


Emulation of packet loss

Loss from client → server

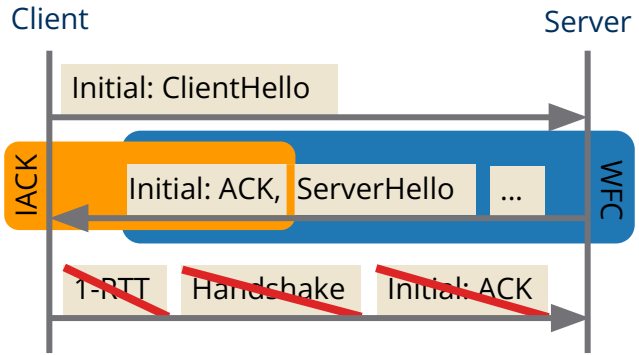


Loss from server → client

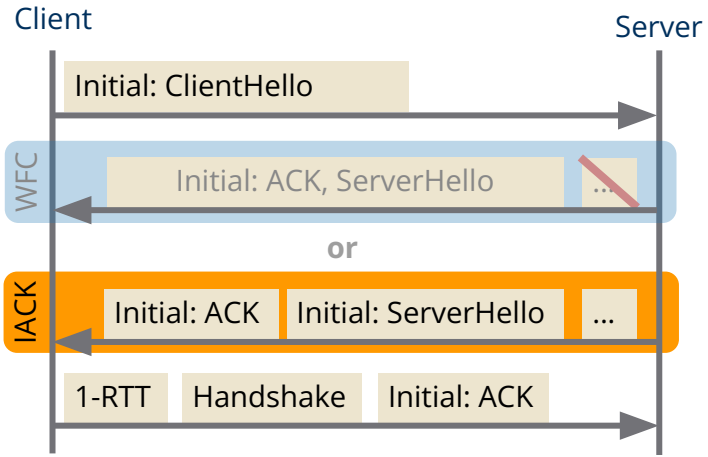


Emulation of packet loss

Loss from client → server

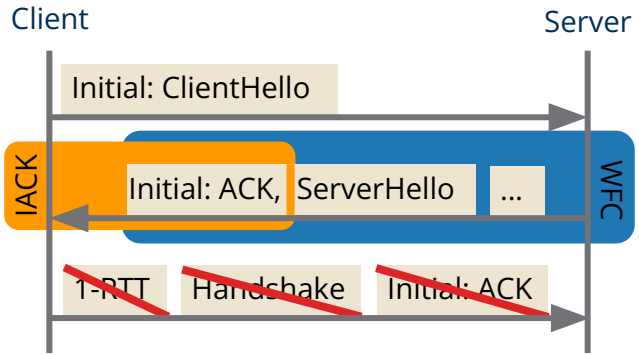


Loss from server → client

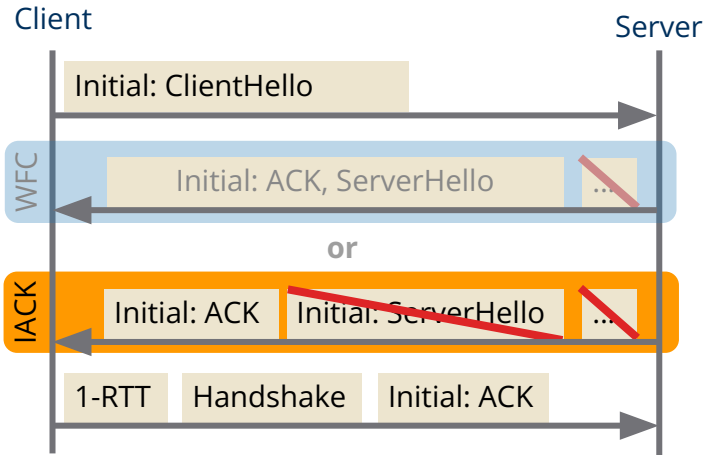


Emulation of packet loss

Loss from client → server

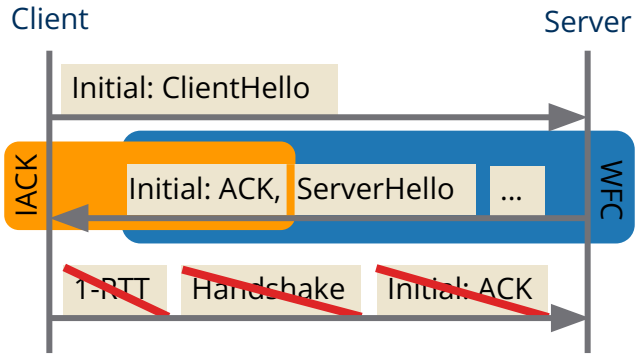


Loss from server → client

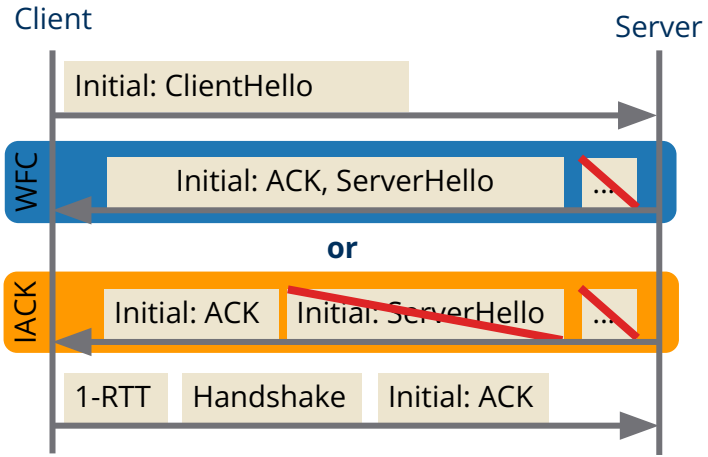


Emulation of packet loss

Loss from client → server

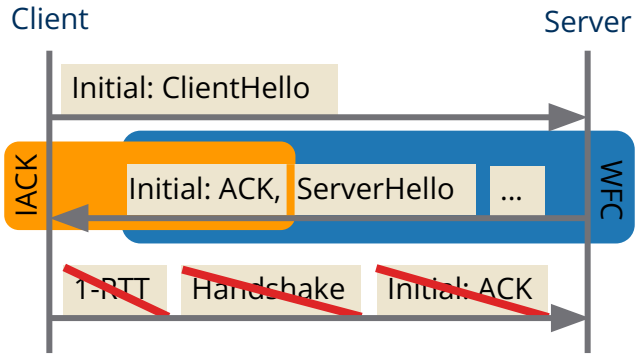


Loss from server → client

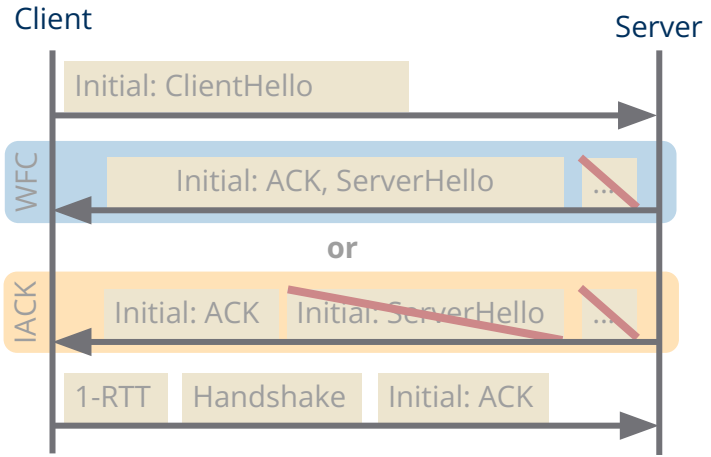


Emulation of packet loss

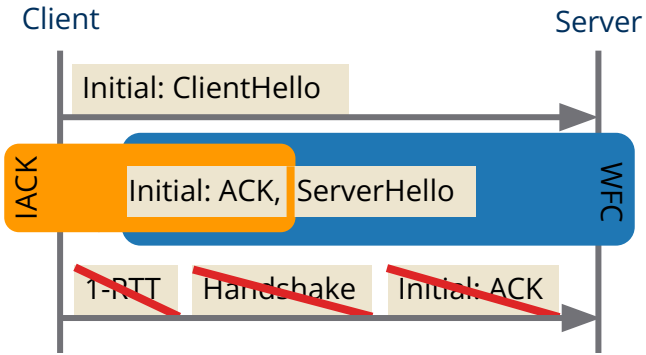
Loss from client → server

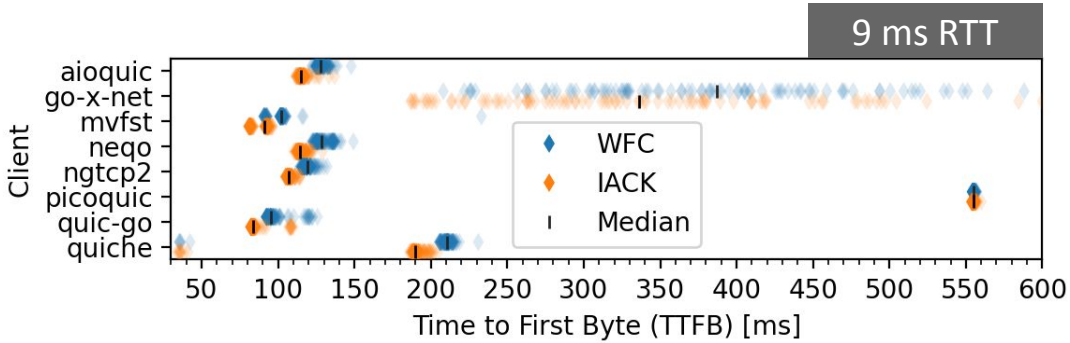


Loss from server → client



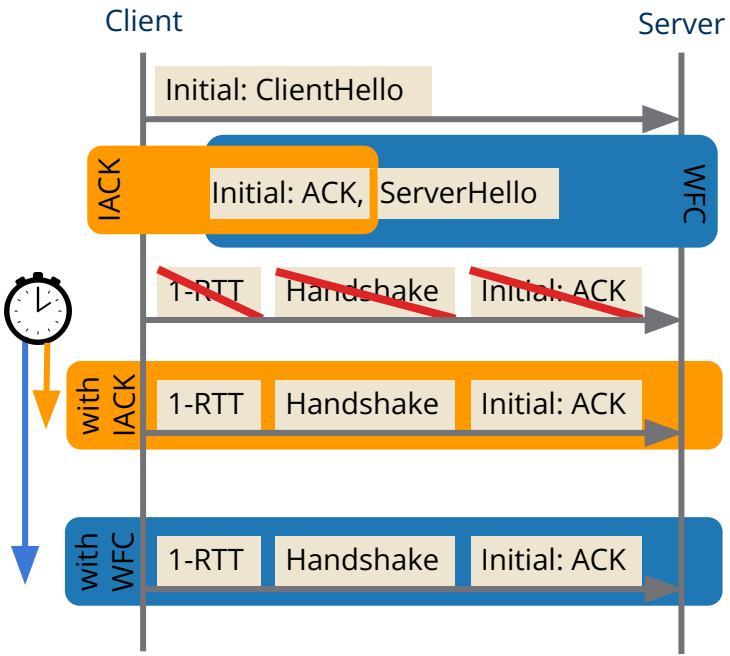
Client → server: Second client flight is lost

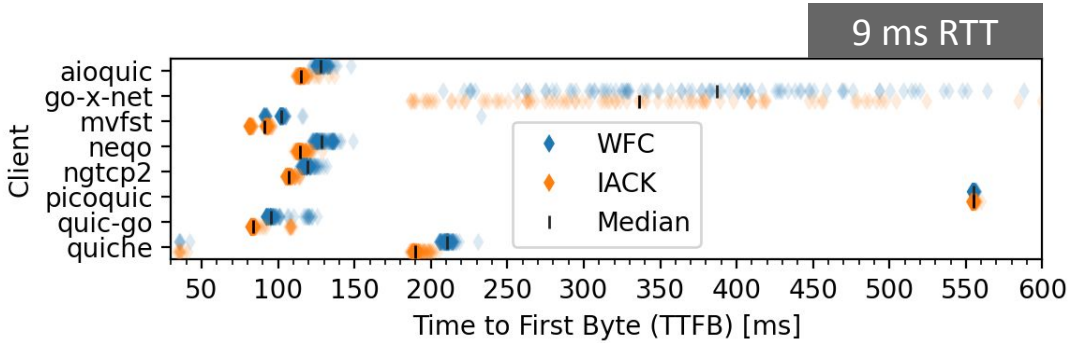




Using IACK, clients need to wait on median between 10 (mvfst) and 28 ms (go-x-net) less.

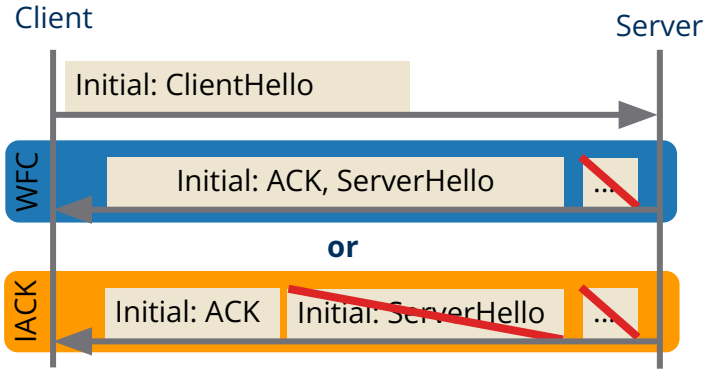
Client → server: Second client flight is lost

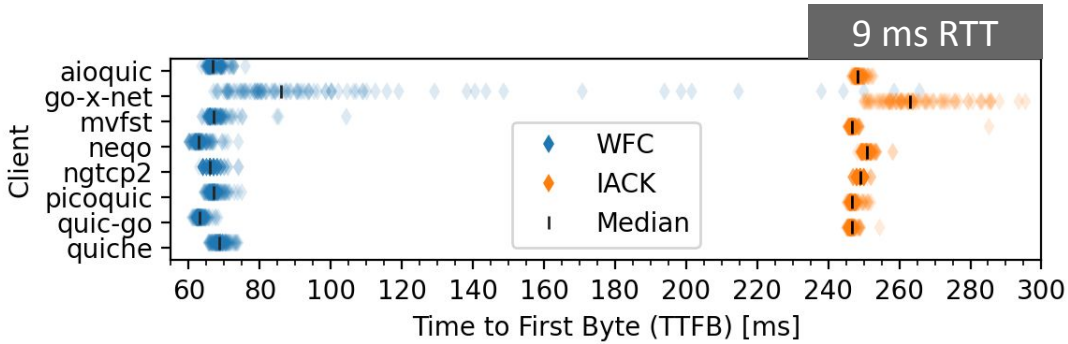




Using IACK, clients need to wait on median between 10 (mvfst) and 28 ms (go-x-net) less.

Server → client: Remaining first server flight is lost



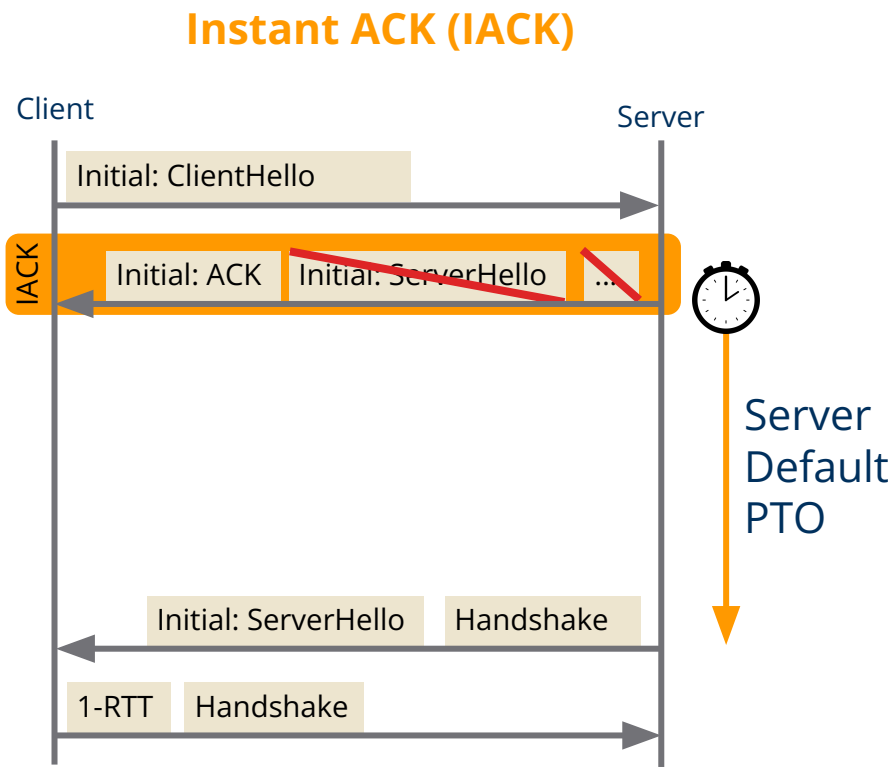


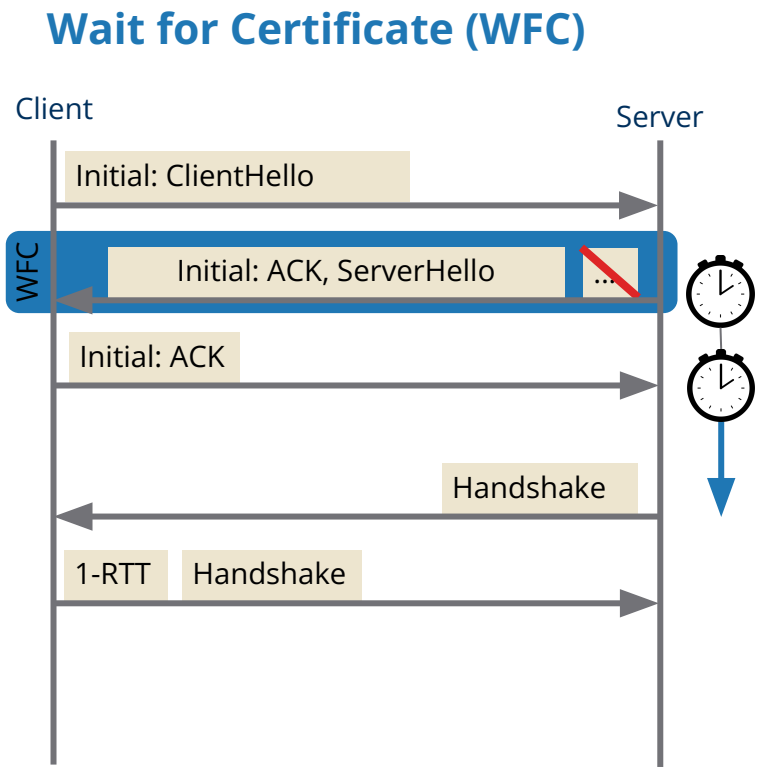
IACK prevents RTT estimation. Server uses default PTO.

WFC allows RTT estimation. Server uses shorter PTO.

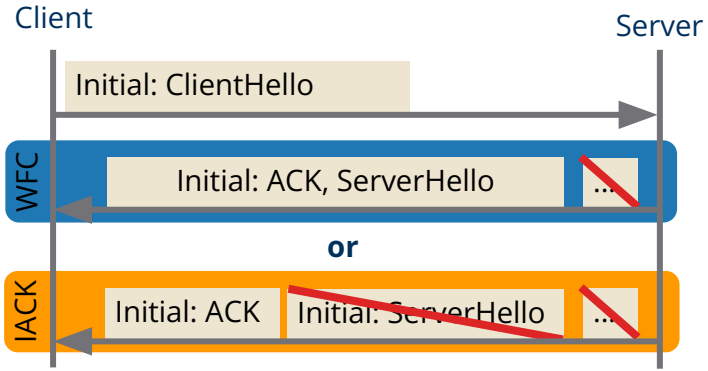
WFC outperforms IACK. WFC accelerates by the server default PTO.

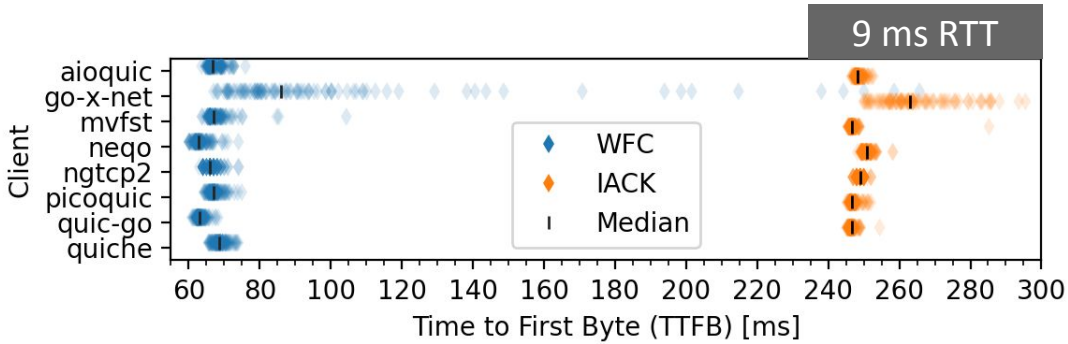
Server → client: Remaining first server flight is lost





Server → client: Remaining first server flight is lost





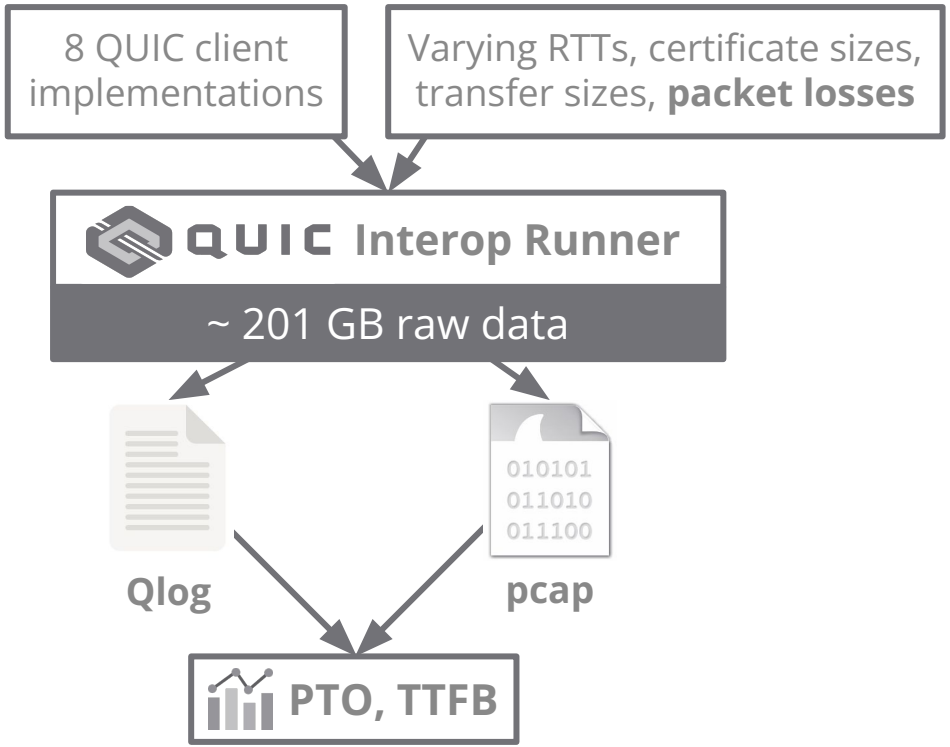
IACK prevents RTT estimation. Server uses default PTO.

WFC allows RTT estimation. Server uses shorter PTO.

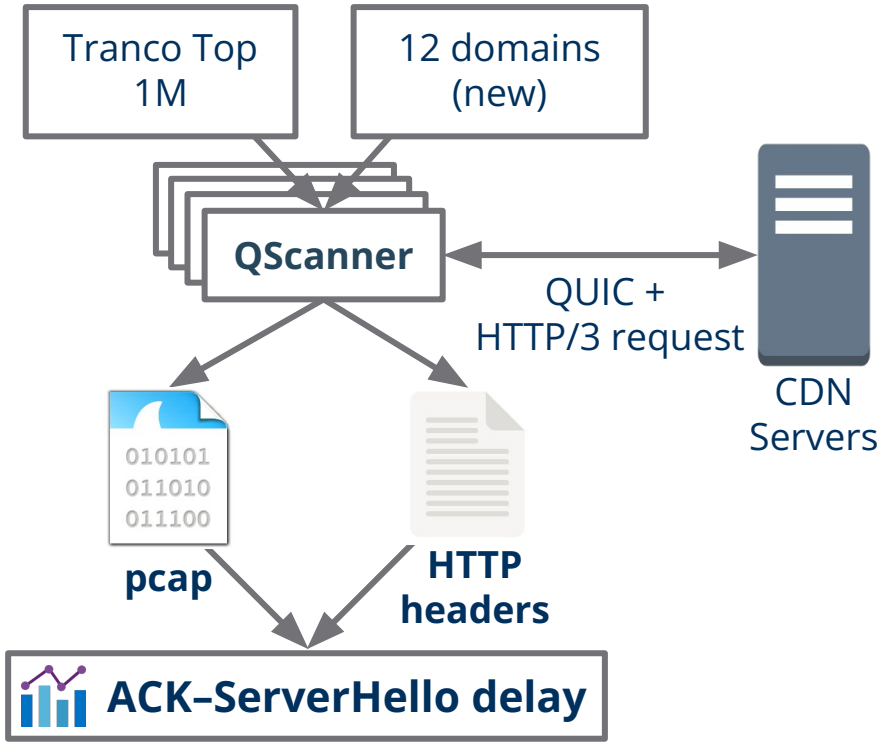
WFC outperforms IACK. WFC accelerates by the server default PTO.

Measurement method and setups

Controlled testbed



CDN deployments



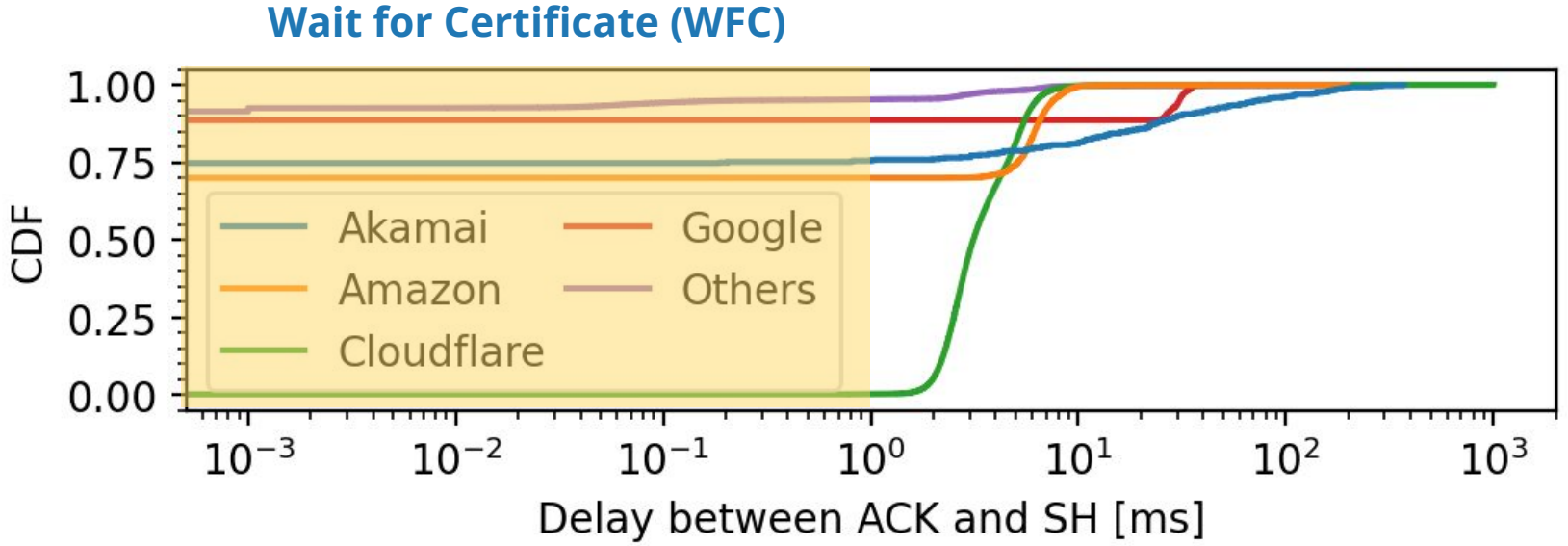
CDNs deploy instant ACK

Akamai, Amazon, Google, and Cloudflare deploy instant ACK.

Fastly, Meta, and Microsoft do not deploy instant ACK.

CDNs deploy instant ACK … but differently

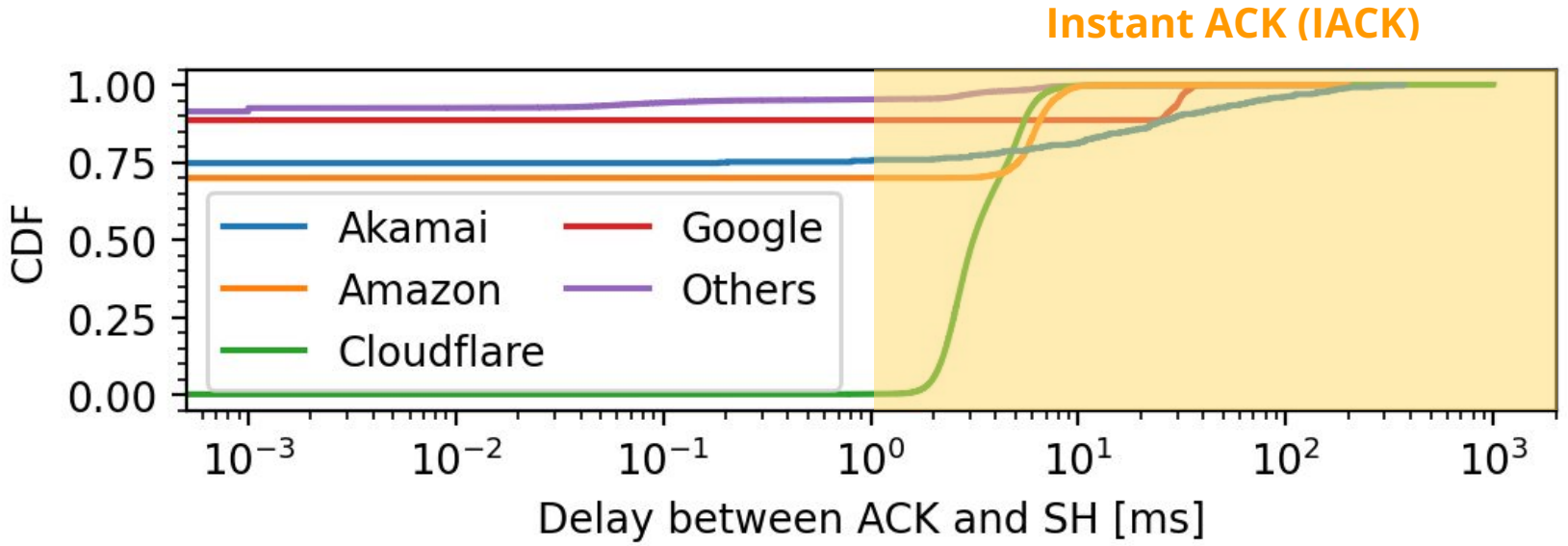
Akamai, Amazon, and Google deploy IACK relatively rarely.



CDNs deploy instant ACK … but differently

Akamai, Amazon, and Google deploy IACK relatively rarely.

Cloudflare uses IACK almost always.



Conclusion

Instant ACK is an RFC-compliant approach to more accurately set PTOs in QUIC. This improves performance not always but often.

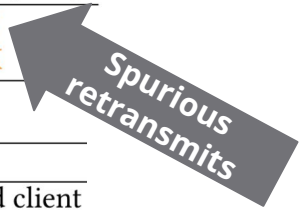
First server flight exceeds anti-amplification limit	No Loss	
	Frontend-certificate store delay Δt	
	Δt < 3 RTT (PTO)	$\Delta t \geq$ 3 RTT (PTO)
No	IACK	WFC
Yes	IACK	IACK

First server flight exceeds anti-amplification limit	Loss	
	First server flight except first datagram	Second client flight
No	WFC	IACK
Yes	IACK	IACK

Conclusion

Instant ACK is an RFC-compliant approach to more accurately set PTOs in QUIC. This improves performance not always but often.

First server flight exceeds anti-amplification limit	No Loss: Frontend-certificate store delay Δt	
	Δt < 3 RTT (PTO)	$\Delta t \geq$ 3 RTT (PTO)
No	IACK	WFC
Yes	IACK	IACK



First server flight exceeds anti-amplification limit	Loss: First server flight except first datagram	Loss: Second client flight
No	WFC	IACK
Yes	IACK	IACK

Conclusion

Instant ACK is an RFC-compliant approach to more accurately set PTOs in QUIC. This improves performance not always but often.

First server flight exceeds anti-amplification limit	No Loss: Frontend-certificate store delay Δt	
	Δt < 3 RTT (PTO)	$\Delta t \geq$ 3 RTT (PTO)
No	IACK	WFC
Yes	IACK	IACK

Spurious retransmits

First server flight exceeds anti-amplification limit	Loss: First server flight except first datagram	Loss: Second client flight
No	WFC	IACK
Yes	IACK	IACK

Combine with PING

Conclusion

Instant ACK is an RFC-compliant approach to more accurately set PTOs in QUIC. This improves performance not always but often.

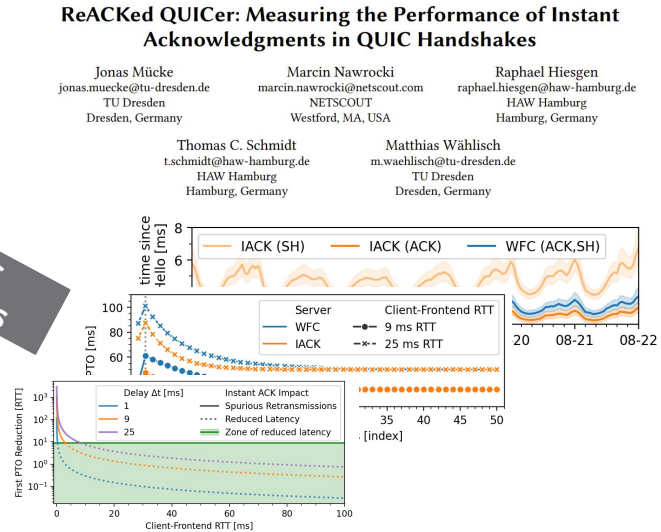
First server flight exceeds anti-amplification limit	No Loss: Frontend-certificate store delay Δt Δt < 3 RTT (PTO)	No Loss: Frontend-certificate store delay Δt $\Delta t \geq$ 3 RTT (PTO)
No	IACK	WFC
Yes	IACK	IACK

Spurious retransmits

First server flight exceeds anti-amplification limit	Loss: First server flight except first datagram	Loss: Second client flight
No	WFC	IACK
Yes	IACK	IACK

Combine with PING

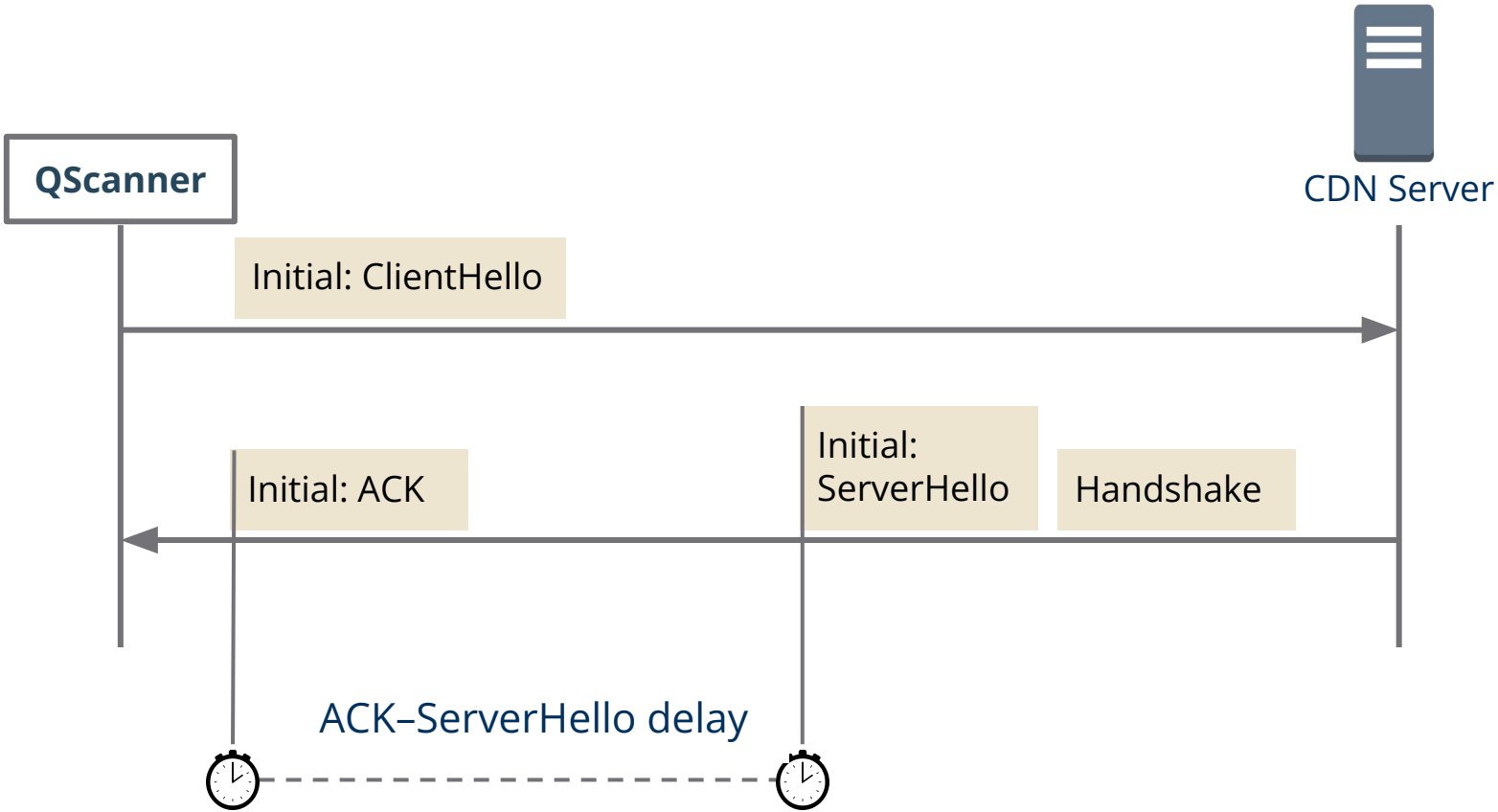
More results in our IMC'25 paper, e.g., picoquic and mvfst conflict with IACK



https://doi.org/10.1145/3646547.3689022

Backup

Measuring improvements by IACK in the wild



Deployment guidelines

Alternatives to instant ACK

(1) Acknowledgement delay of QUIC ACKs

- Ignored by PTO initialization
- Not all implementations send ACK of the ClientHello
- Current deployments: ACK delay often exceeds connection RTT of coalesced ACK, SH

(2) Reinitialization after the first ACK

- Accurately represents connection RTT
- Effective only after the next ACK

The acknowledgment delay of QUIC ACKs or reinitialization strategy cannot provide precise RTT estimates from the first RTT sample.

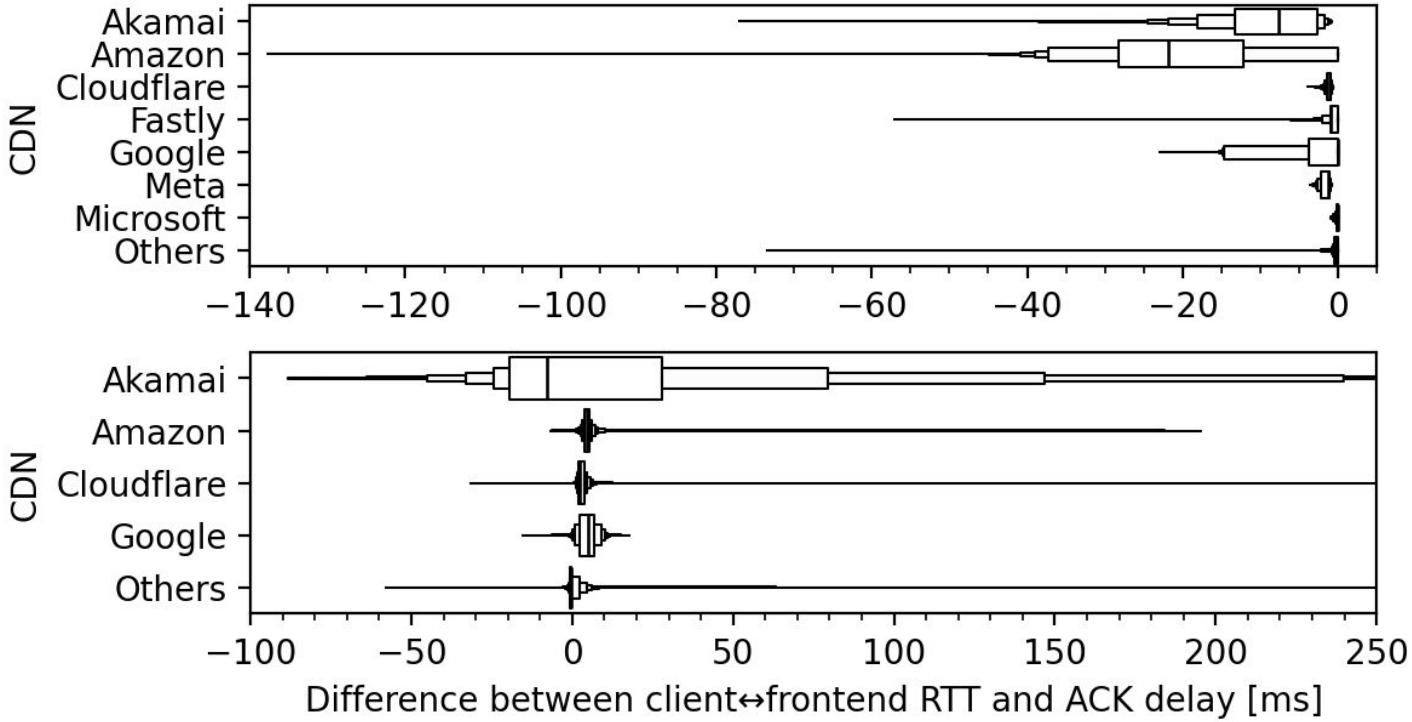
Acknowledgment delay of the first ACK received from QUIC servers

	First ACK Delay [ms]					
	Repetition [index]					
	1	2	3	1	2	3
Server	Initial Packet			Handshake Packet		
aioquic	3.3	3.4	3.3	-	-	-
go-x-net	0.0	0.0	-	-	-	-
haproxy	1.0	1.0	-	0.0	0.0	-
kwik	0.0	0.0	0.0	-	-	-
lsquic	1.2	1.1	1.2	0.2	0.2	0.2
msquic	-	-	-	-	-	-
mvfst	0.8	-	0.7	0.2	-	0.1
neqo	0.0	0.0	-	0.0	0.0	-
nginx	0.0	0.0	0.0	-	-	-
ngtcp2	0.0	0.0	0.0	-	-	-
picoquic	0.8	0.7	0.8	-	-	-
quic-go	0.0	0.0	0.0	-	-	-
quiche	1.4	1.4	1.5	-	-	-
quinn	0.4	-	0.4	-	-	-
s2n-quic	14.0	15.2	14.1	-	-	-
xquic	1.3	1.1	1.2	-	0.5	0.5

QUIC ACK delay vs. client-frontend RTT

Coalesced ACK,SH

Instant ACK



Which deployments benefit from instant ACK?

Which deployments benefits from instant ACK?

Improvements are not limited to CDN setups with separate certificate servers. All servers with delays in sending ACKs to the client profit from instant ACK.

Do alternatives to instant ACK exist?

There is no alternative to provide the client with precise RTT information from the beginning of the connection. The acknowledgment delay is ignored by the PTO initialization and often set incorrectly by server implementations.

The source of QUIC stack delays

Waiting for the certificate (WFC)

- Calculate Initial keys
- Calculate Handshake keys
- Construct and send TLS ServerHello
- Load and send certificate
- Calculate and send signature

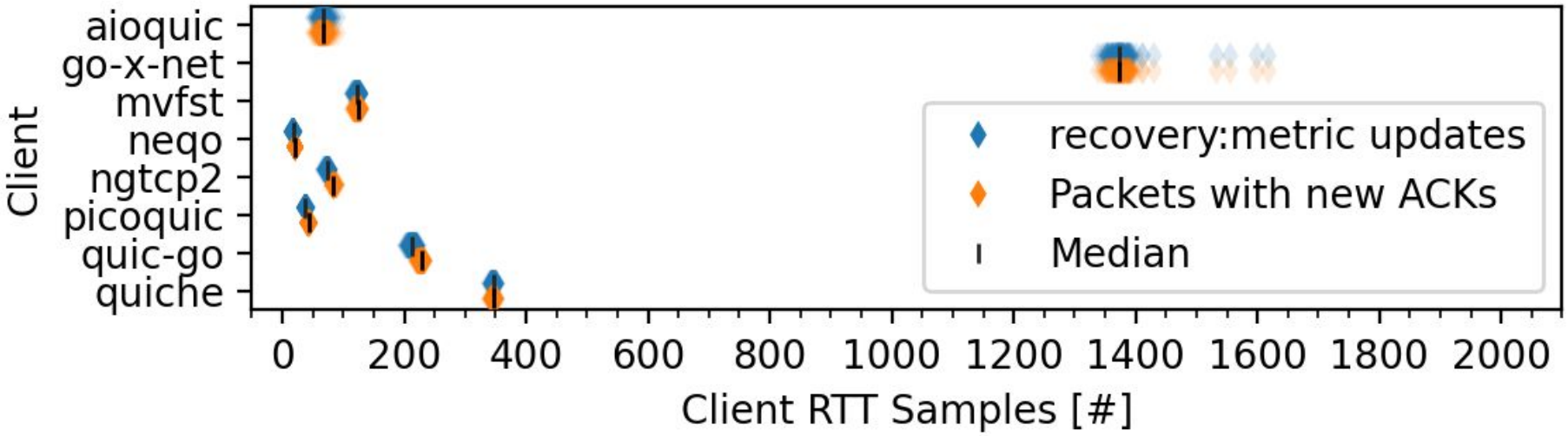
Profiling our server implementation quic-go, signature calculation was the single most CPU consuming function.

Instant ACK (IACK)

- Calculate Initial keys

Without additional delay, client RTT estimates differ on median between 3 and 8 ms.

Number of exposed RTT samples and newly acknowledging ACKs for 10 MB file transfer at 100 ms RTT, WFC



Does instant ACK generalize to 0-RTT and Retry handshakes?

0-RTT connections perform a 1-RTT QUIC handshake along with the 0-RTT exchange.

Retry packets cause the client to resend the ServerHello with a token.

In 0-RTT and Retry handshakes, instant ACK helps to set a more precise RTT estimate at the client.

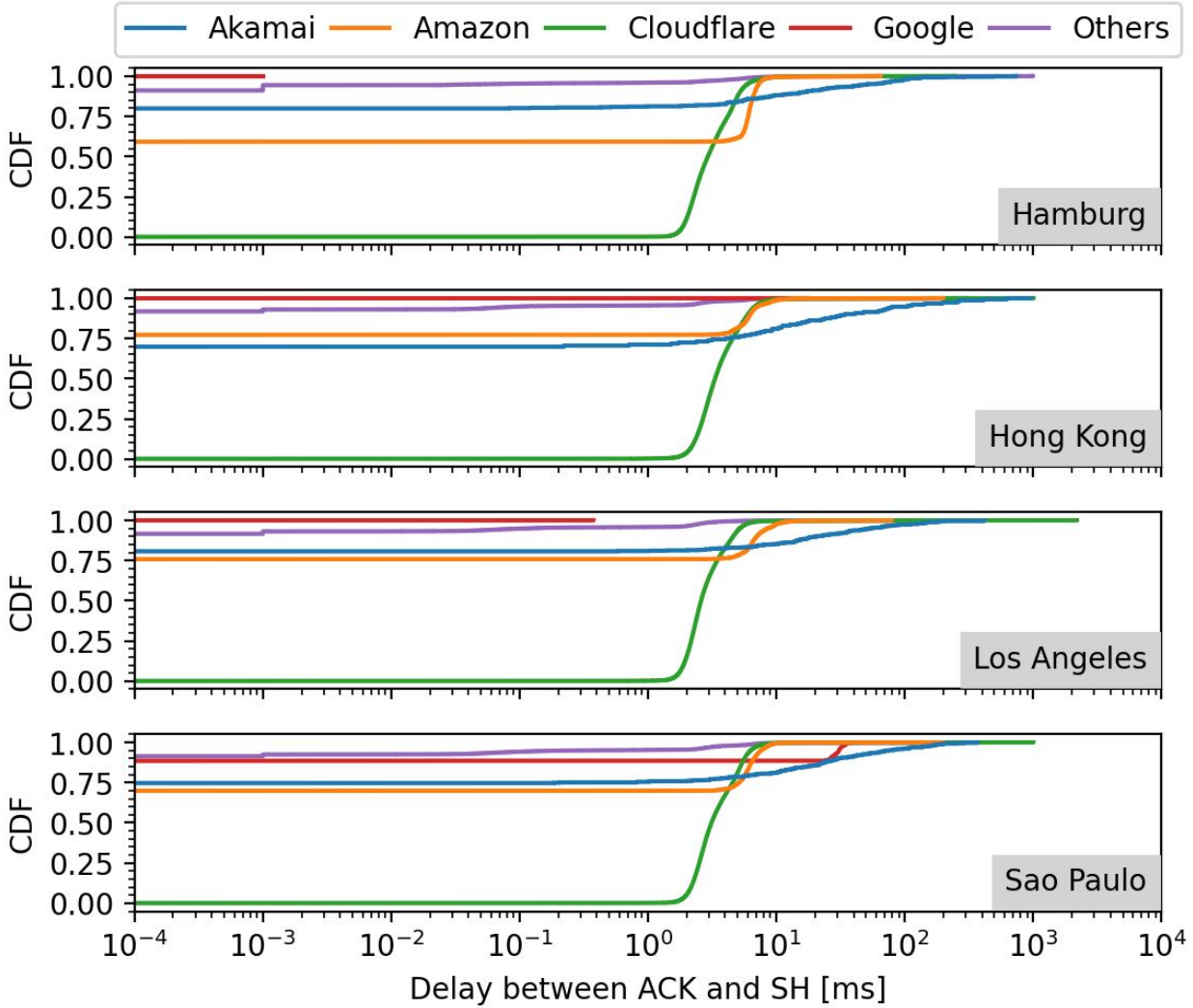
CDN deployment

Instant ACK support of CDNs

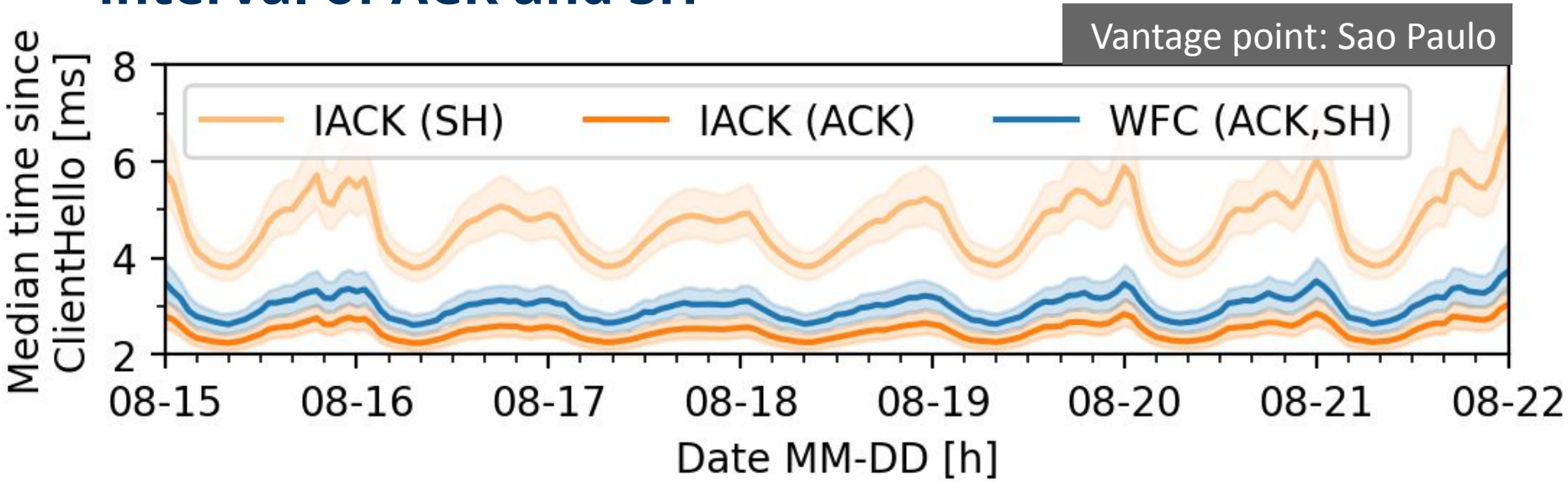
Domains from the Tranco Top 1M, Aug 06, 2024

		IACK deployment enabled (max.)	
CDN	Domains [#]	Domains [%]	Variation [%]
Akamai	533	32.2	12.9
Amazon	4338	41.0	18.0
Cloudflare	247407	99.9	0.1
Fastly	3960	0.0	0.0
Google	6062	11.5	11.5
Meta	112	0.0	0.0
Microsoft	34	0.0	0.0
Others	26404	21.5	2.3

Tranco Top 1M: Delay between ACK and SH [ms]



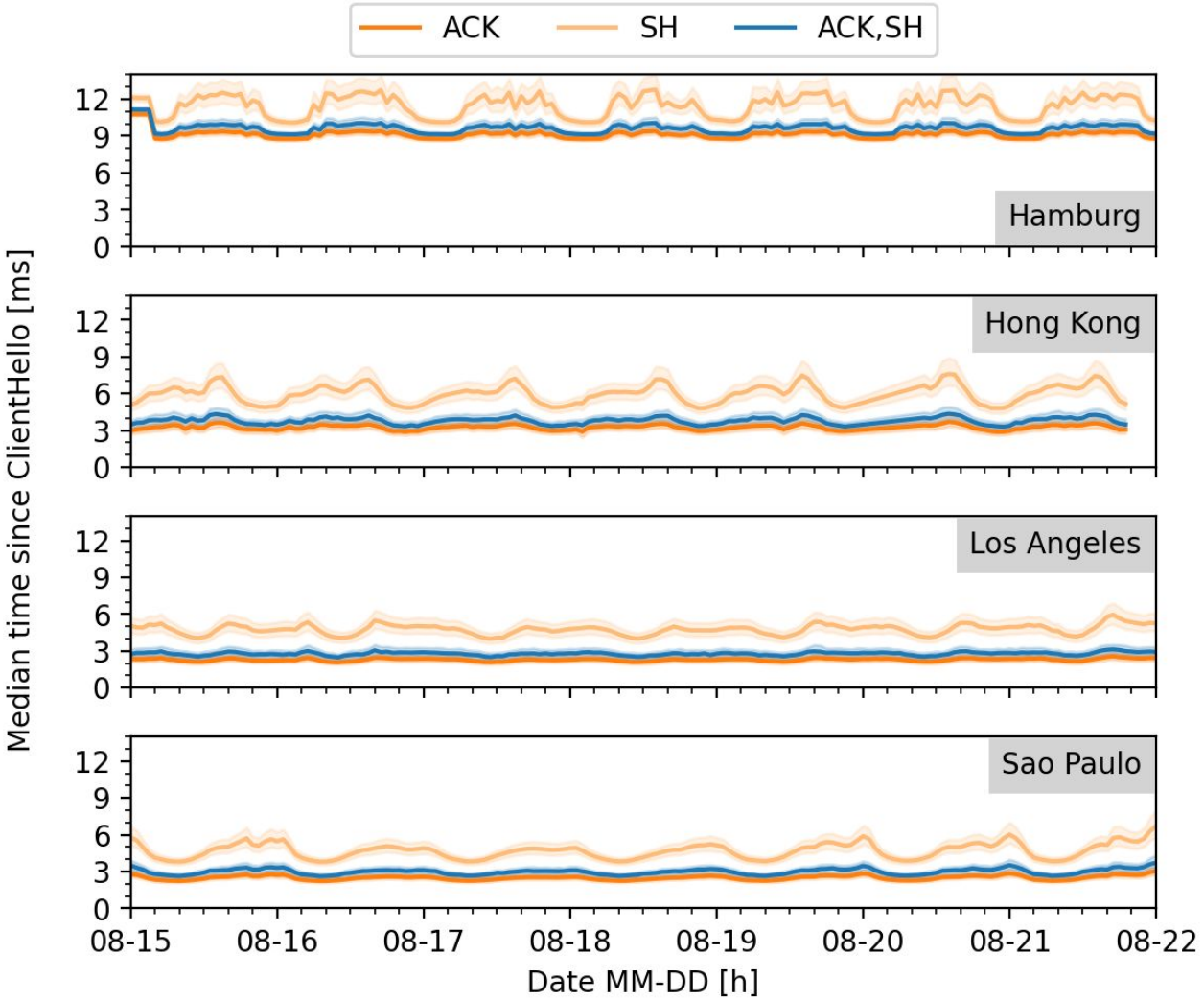
Cloudflare: Reception latency and 50% percentile interval of ACK and SH



WFC is significantly more common for frequently requested domains.

WFC shows similar performance to IACK (□ likely cached certificate).

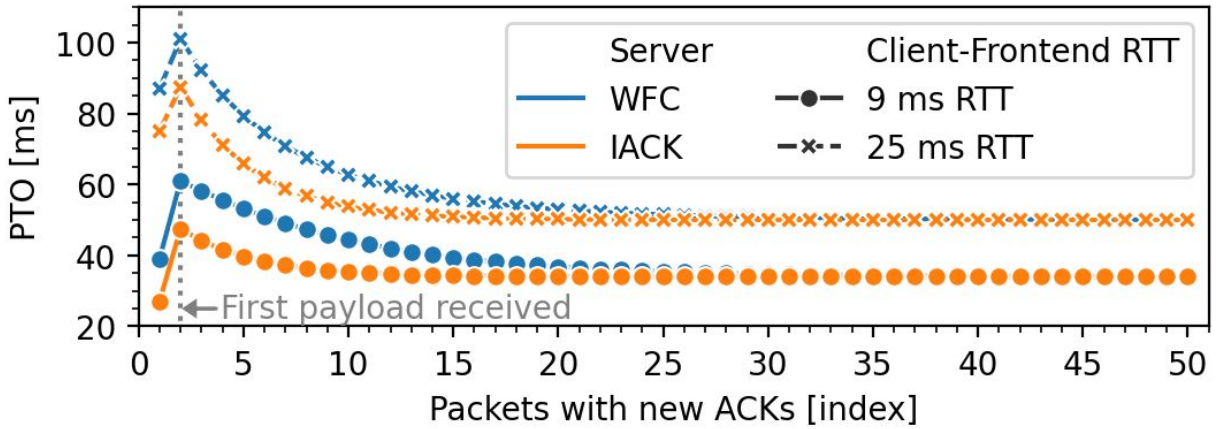
Cloudflare: Median time since ClientHello [ms]



Testbed emulation

PTO evolution in a setting with constant RTTs

Numerical analysis

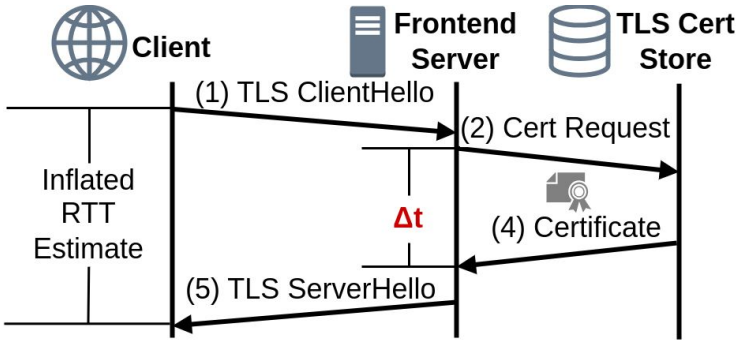


Emulation: IACK reception 4 ms earlier than WFC

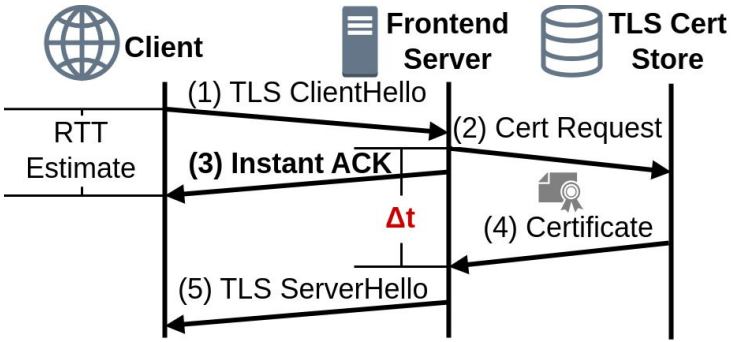
Initial PTO improvement IACK: 12 ms

Scenario: Δt = 200ms, large certificate, no loss

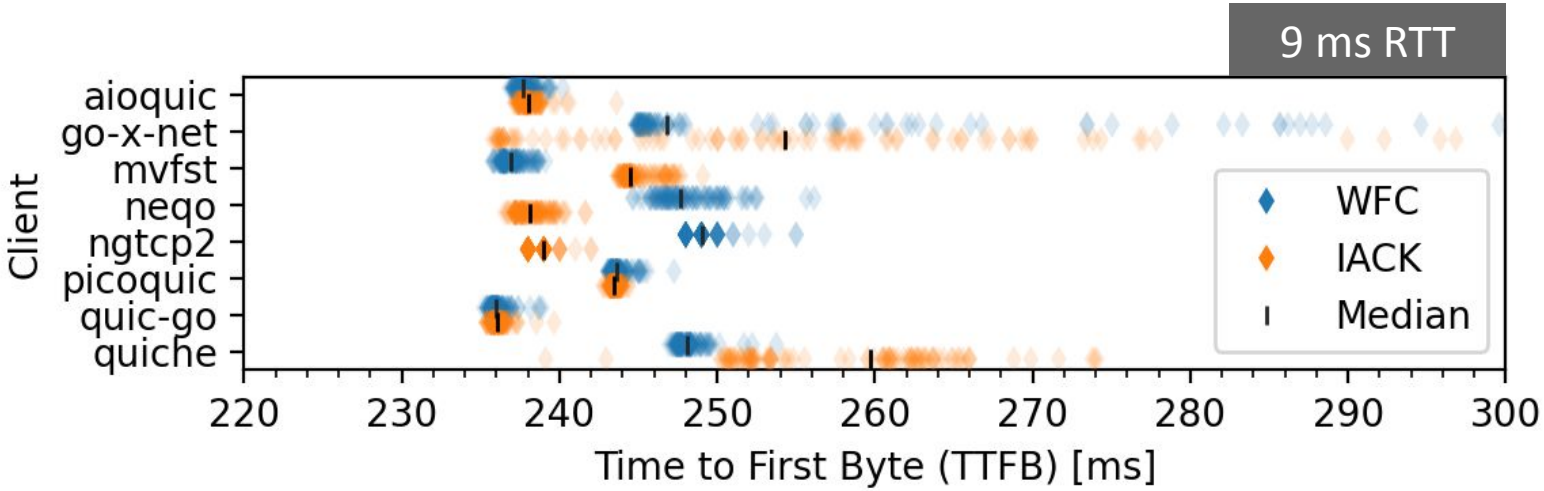
Waiting for the certificate (WFC)



Instant ACK (IACK)



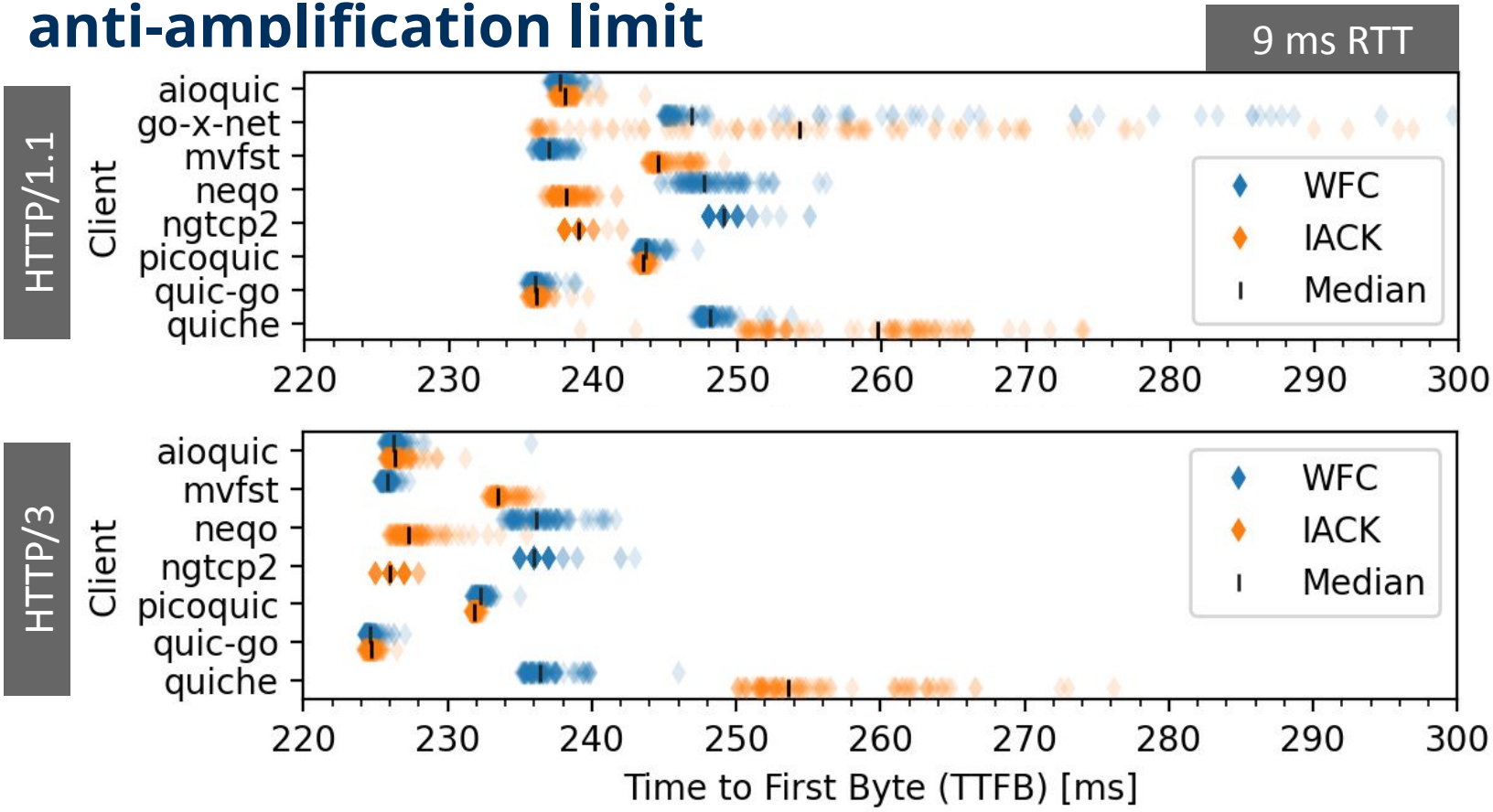
Scenario: No loss, server blocked by anti-amplification limit



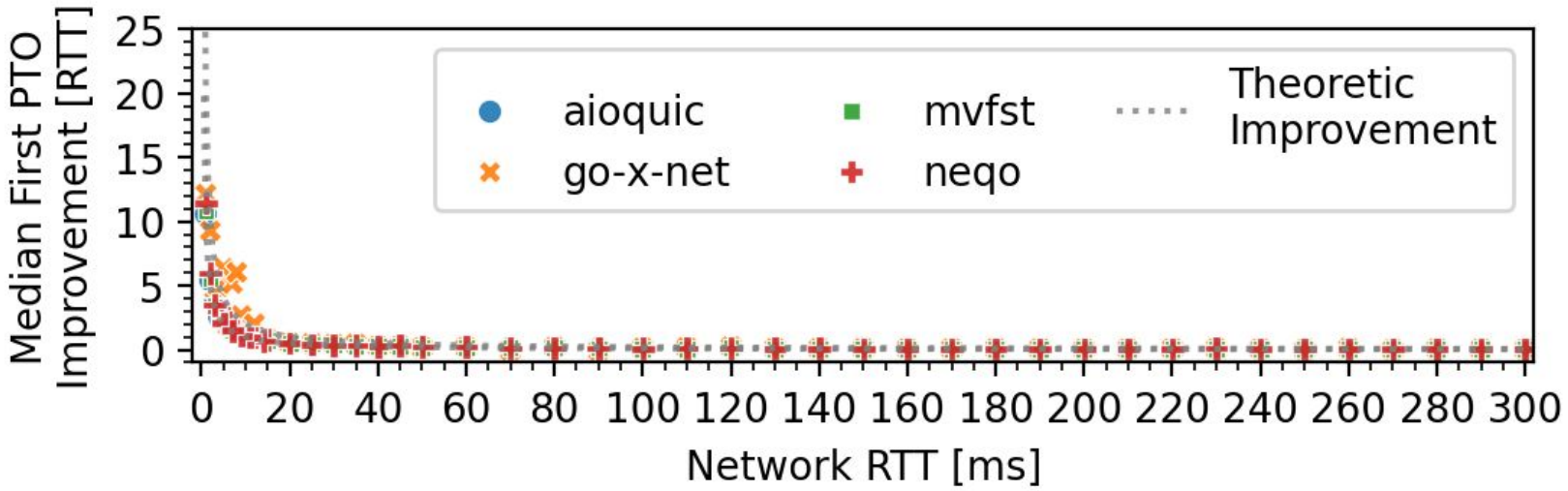
TTFB reduced by 9.6 ms (*neqo*) and 10 ms (*ngtcp2*).

quiche drops replies to PING frames as invalid together with coalesced packets, which costs additional resends.

Scenario: No loss, server blocked by anti-amplification limit

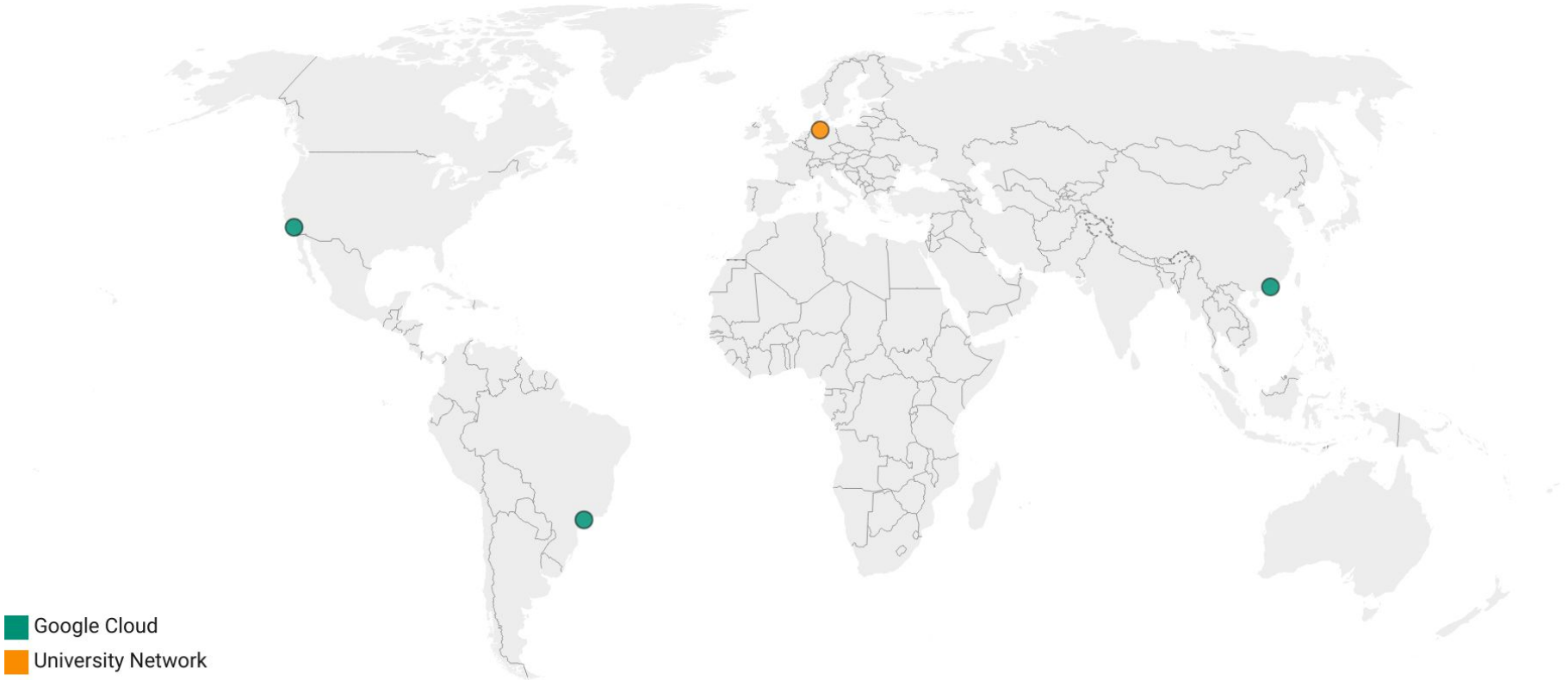


Median improvement of the first PTO



Measurement setup (extended)

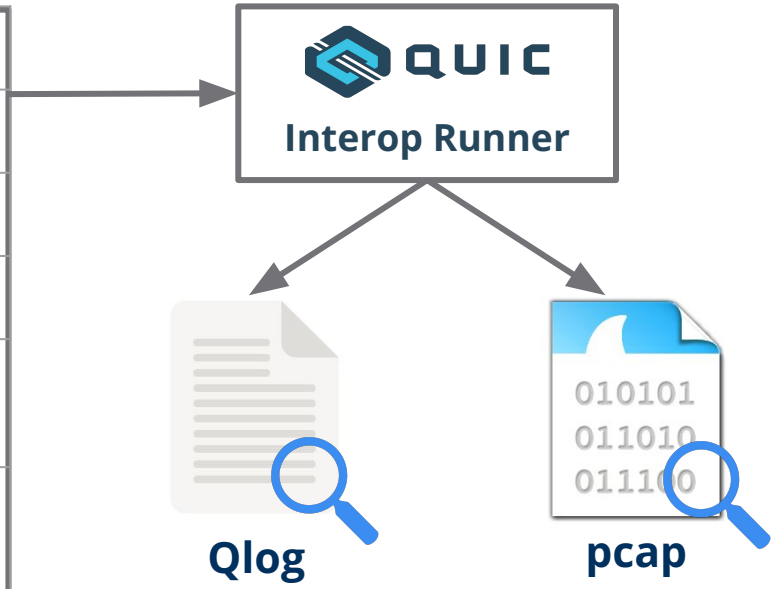
Vantage points locations



Measurement method and setup

Emulation with the QUIC Interop Runner

Parameter	Value range
RTT	1-300 ms
Transfer size	10 KB, 10 MB
Certificate size	1,212 B, 5113 B
Scenarios without loss	Server blocked by anti-amplification limit
Scenarios with packet loss	- second client flight lost - remaining first server flight lost
Repetitions	100



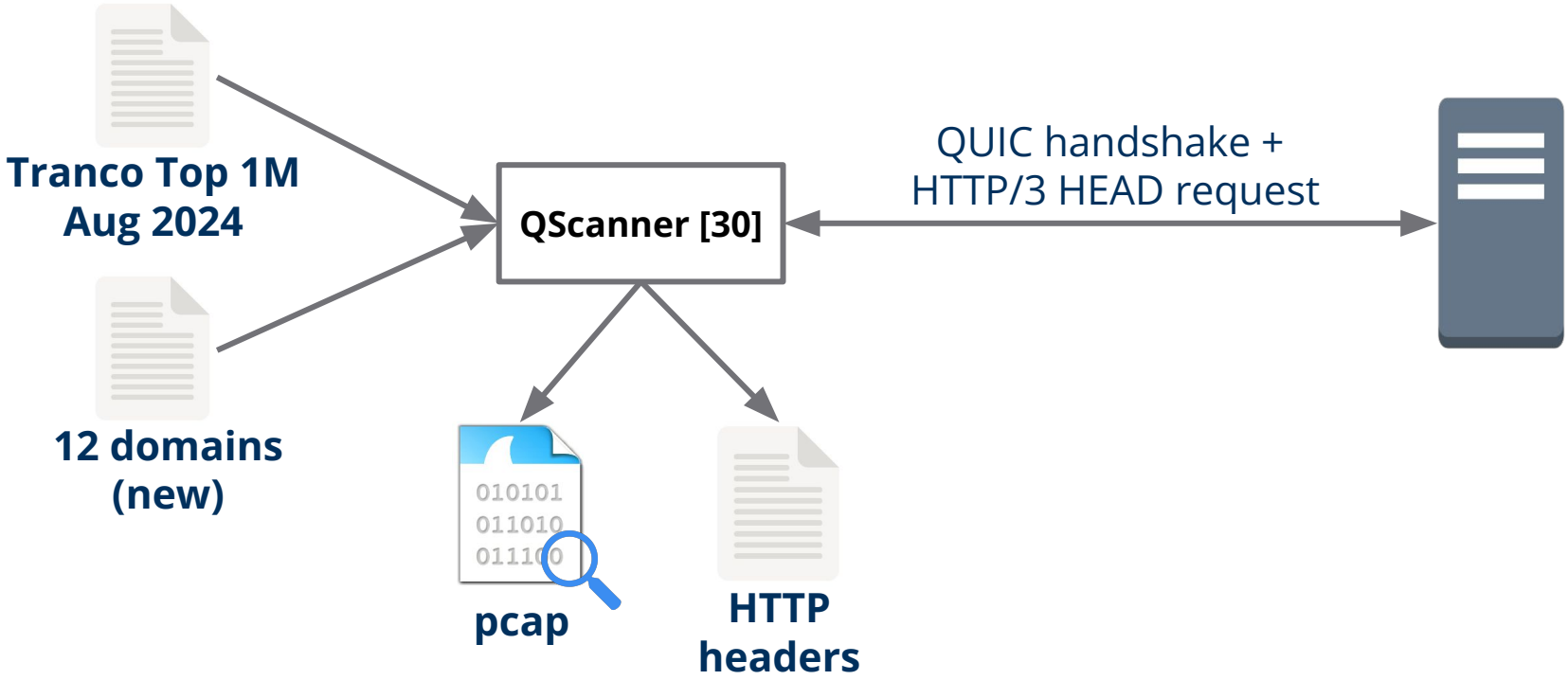
Why use Qlog over packet captures?

Qlog represents the view the implementation has on the connection. This includes all kind of system stack delays.

Drawbacks:

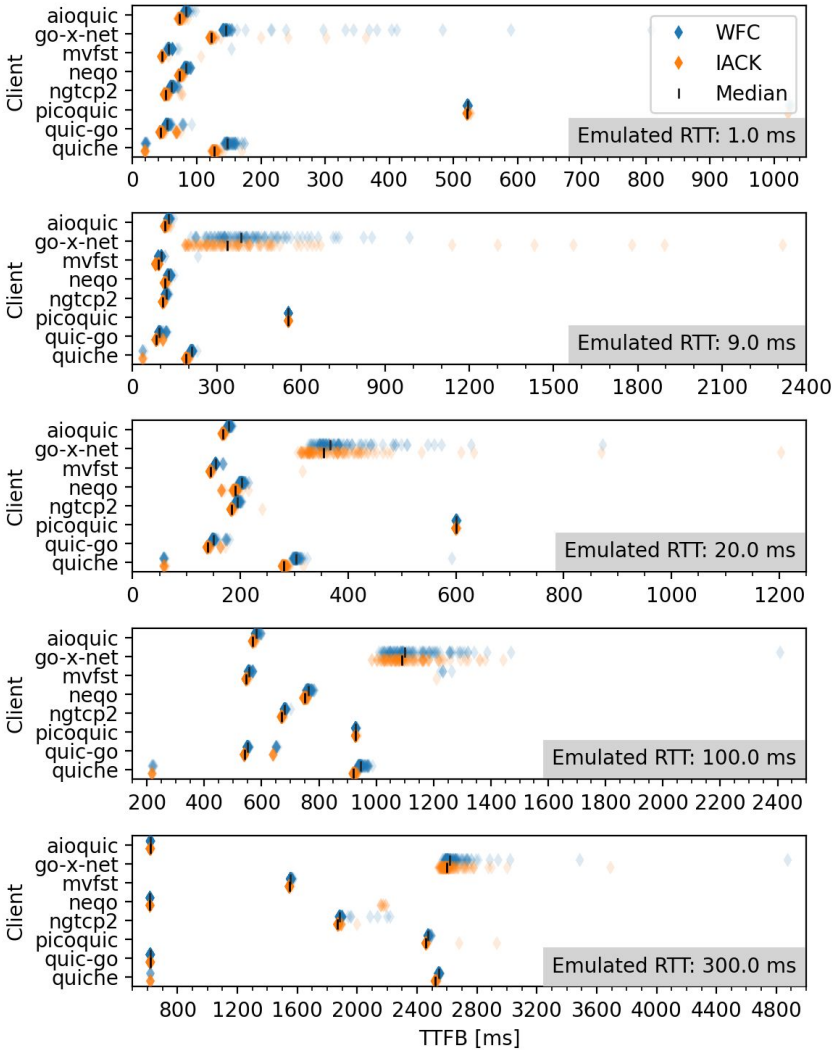
- Not all implementations expose all information in Qlog.
  - Missing RTT variance

Measurement of instant ACK deployments

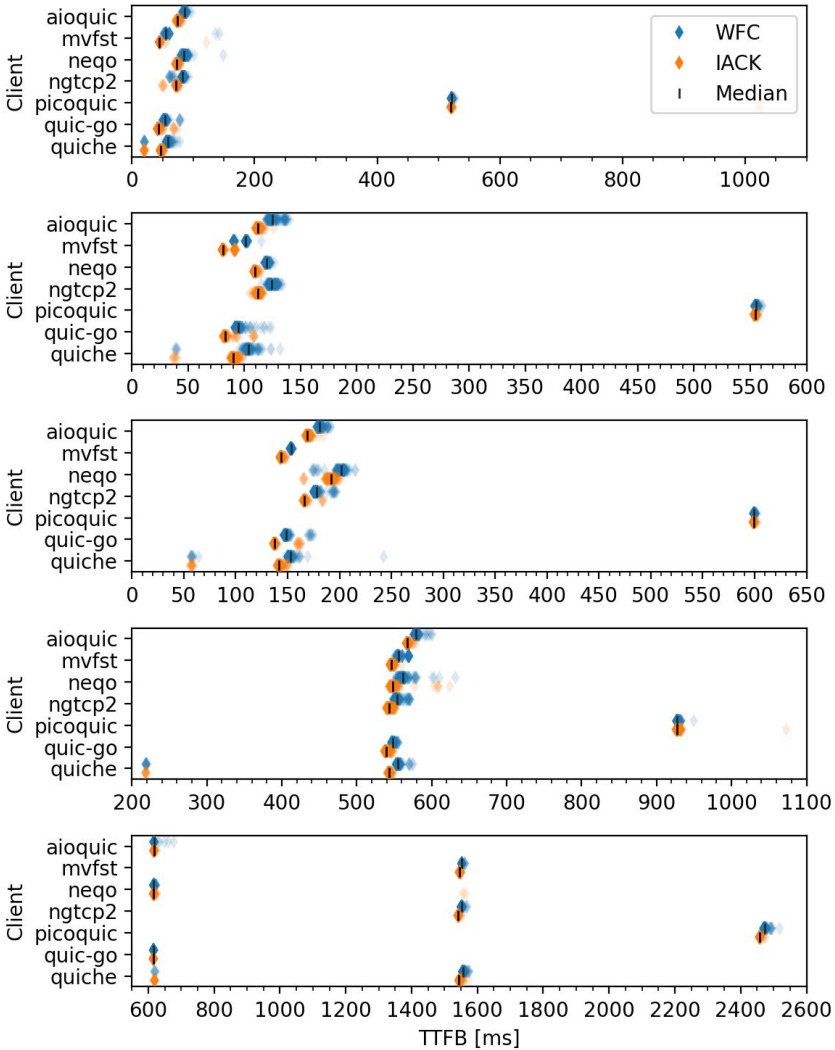


Emulation of packet loss at different RTTs

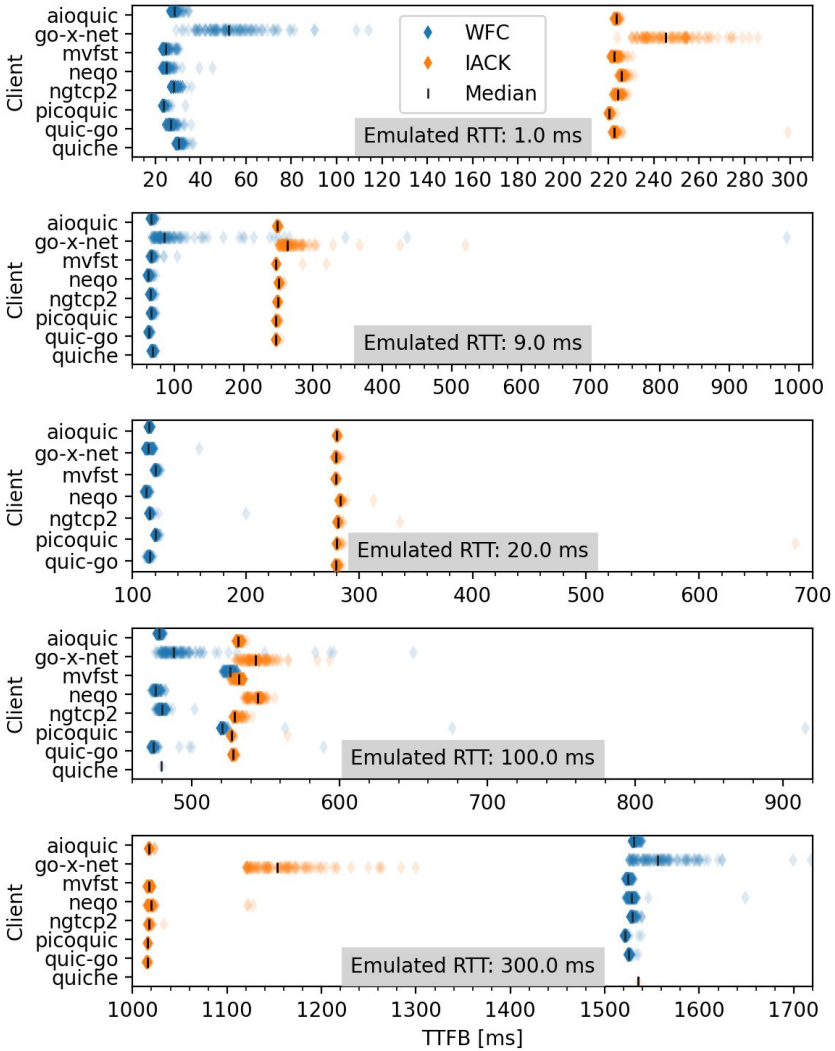
Loss of second client flight: HTTP/1.1



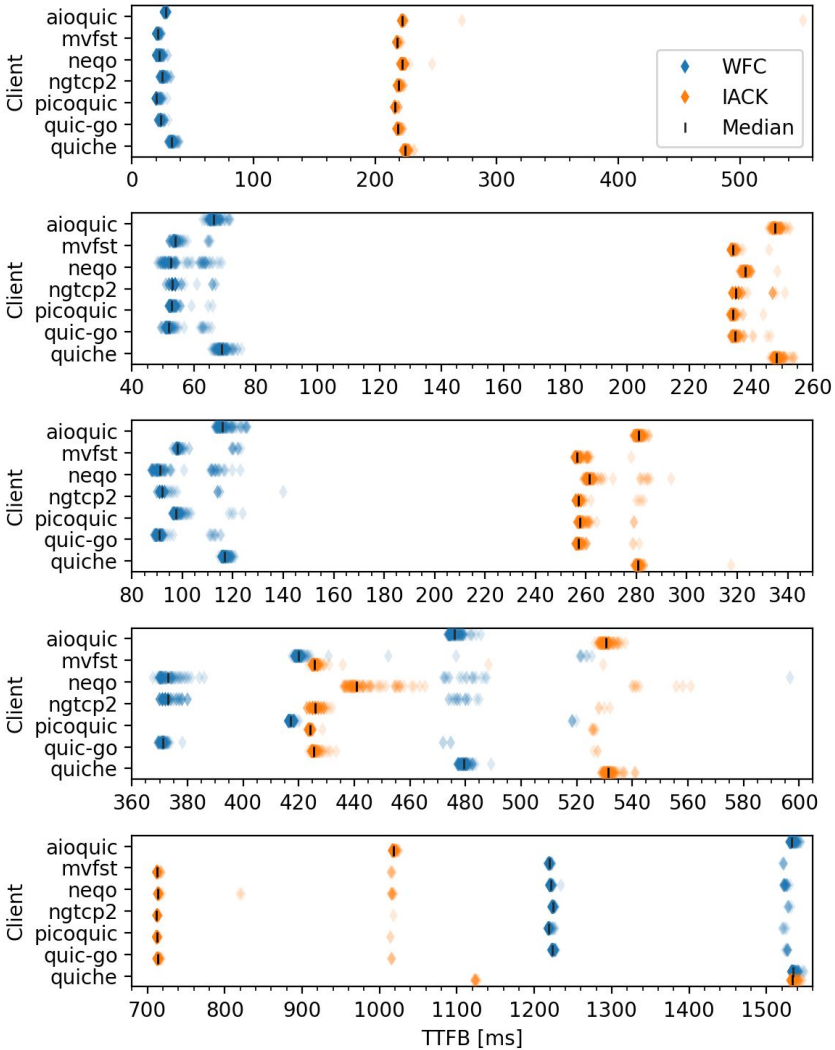
Loss of second client flight: HTTP/3



Loss of remaining first server flight: HTTP/1.1

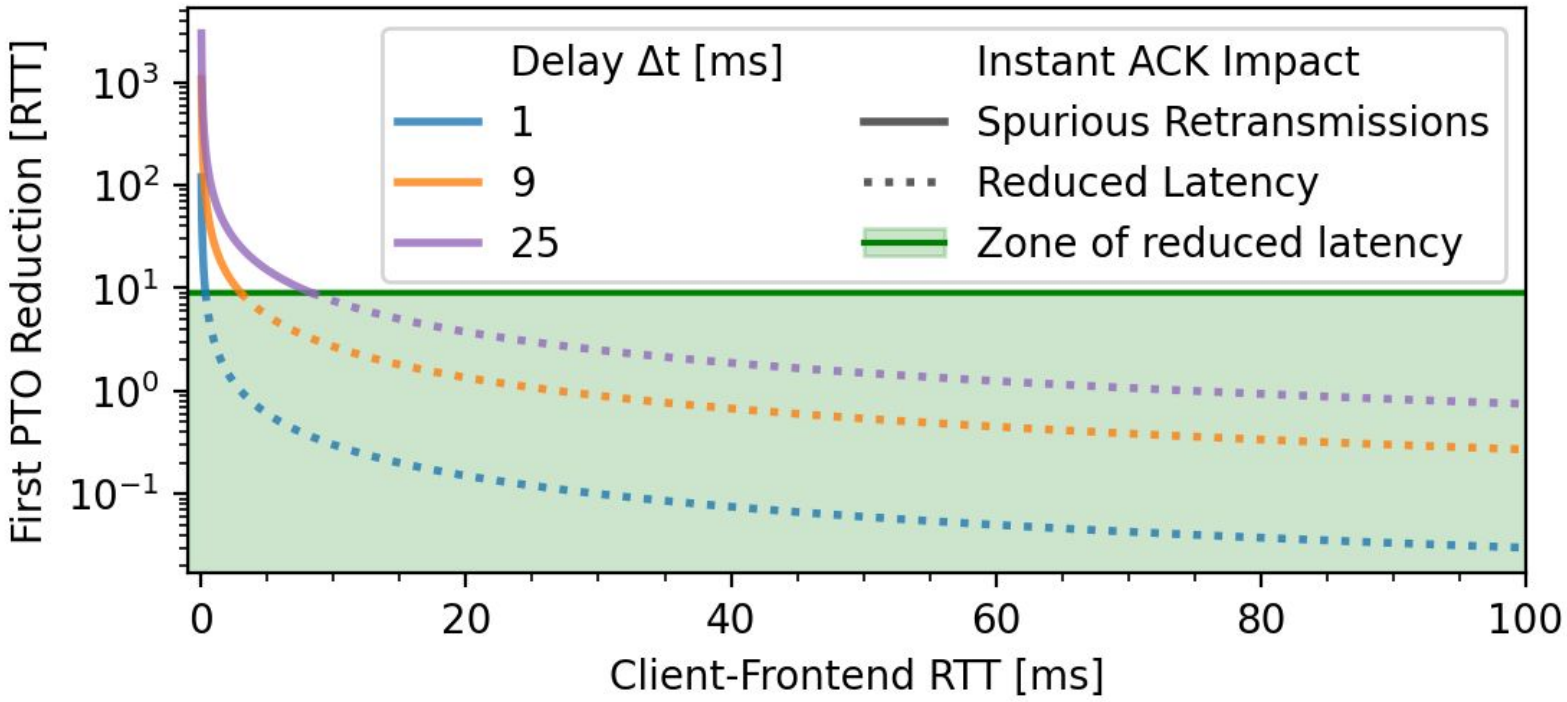


Loss of remaining first server flight: HTTP/3



Numerical analysis

First PTO improvement according to RFC9002 [6]



Probe Timeout (PTO)

```
RTTVar = 3/4 * RTTVar + 1/4 * abs(SmoothedRTT -
         adjusted_rtt)
SmoothedRTT = 7/8 * SmoothedRTT + 1/8 * adjusted_rtt
```

```
PTO = SmoothedRTT + max(4*RTTVar, kGranularity) +
      max_ack_delay
```