QUIC

Design and Internet-Scale Deployment

Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, **Jana Iyengar**, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Jo Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, Zhongyi Shi

Google

A QUIC history

Protocol for HTTPS transport, deployed at Google starting 2014

Between Google services and Chrome / mobile apps

Improves application performance

YouTube Video Rebuffers: 15 - 18%

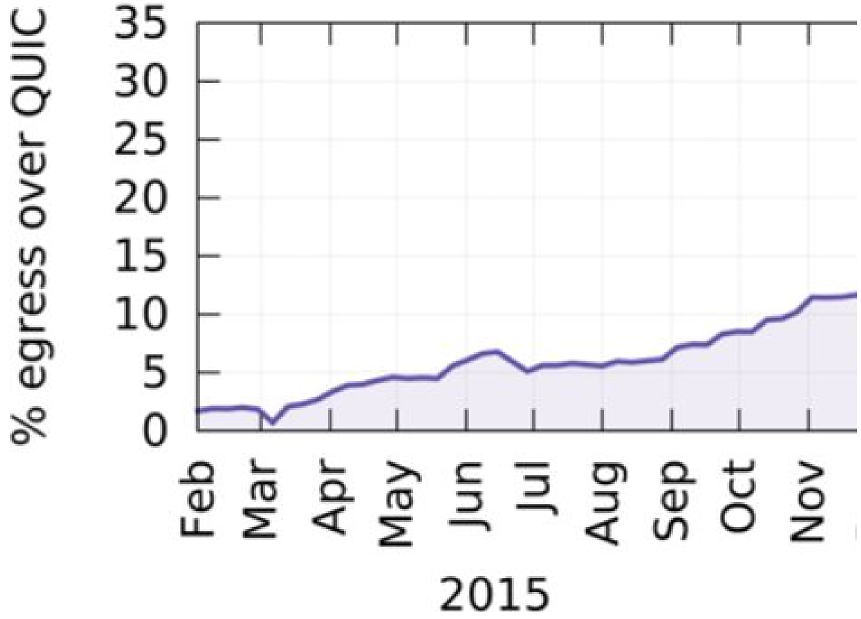
Google Search Latency: 3.6 - 8%

35% of Google's egress traffic (7% of Internet)

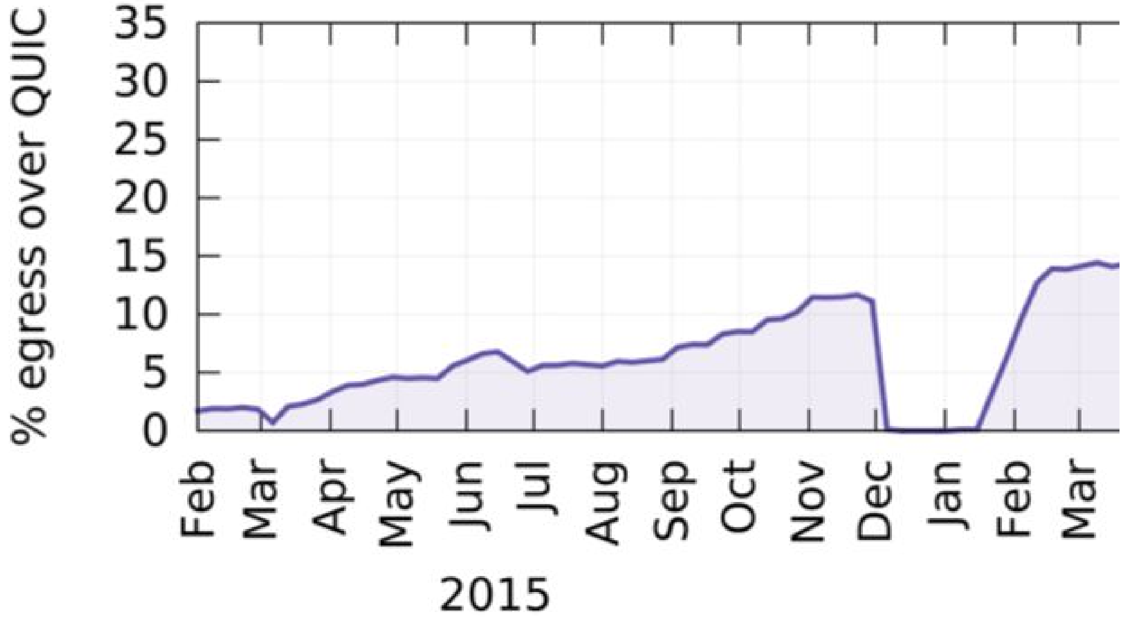
IETF QUIC working group formed in Oct 2016

Modularize and standardize QUIC

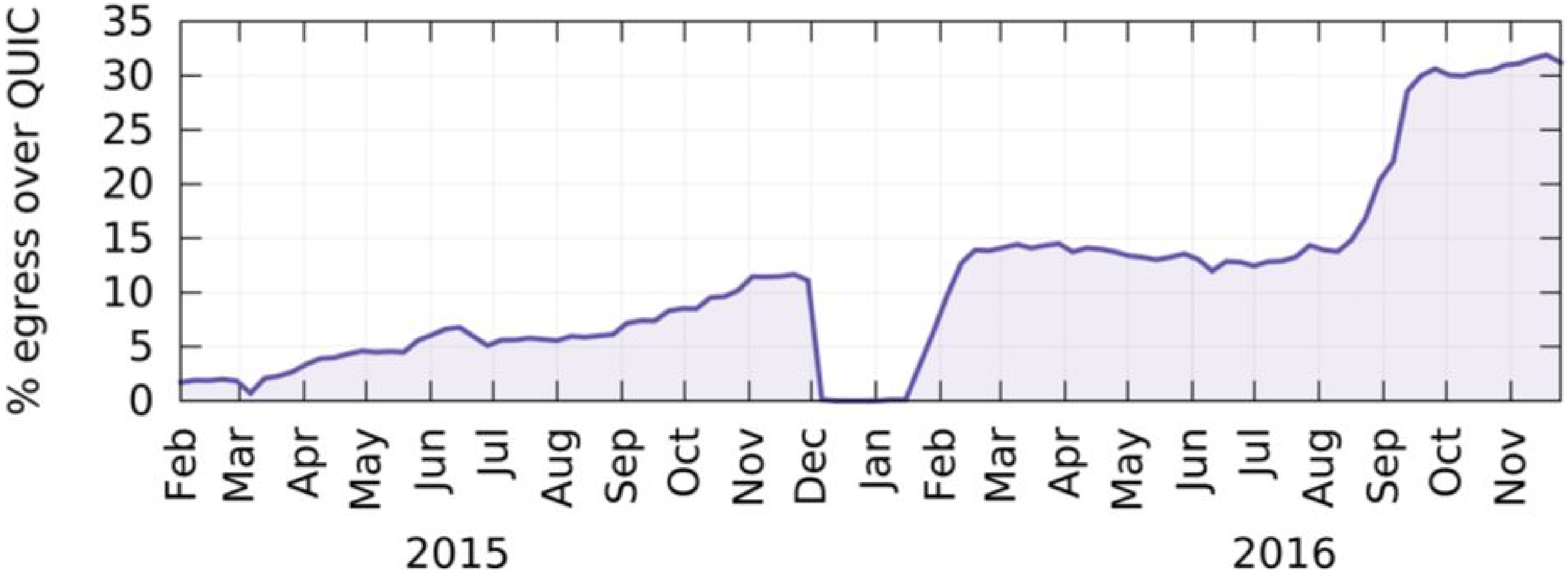
Google's QUIC deployment



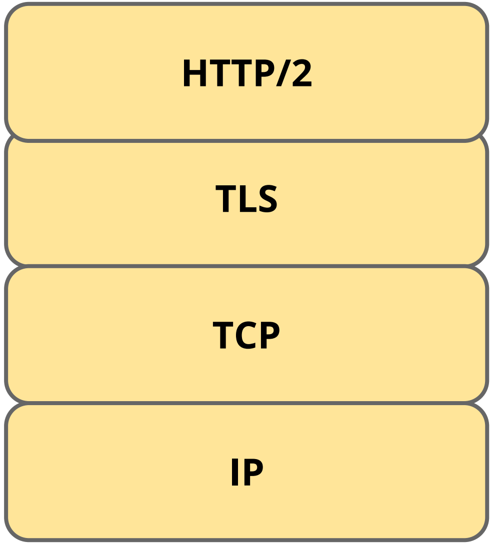
Google's QUIC deployment



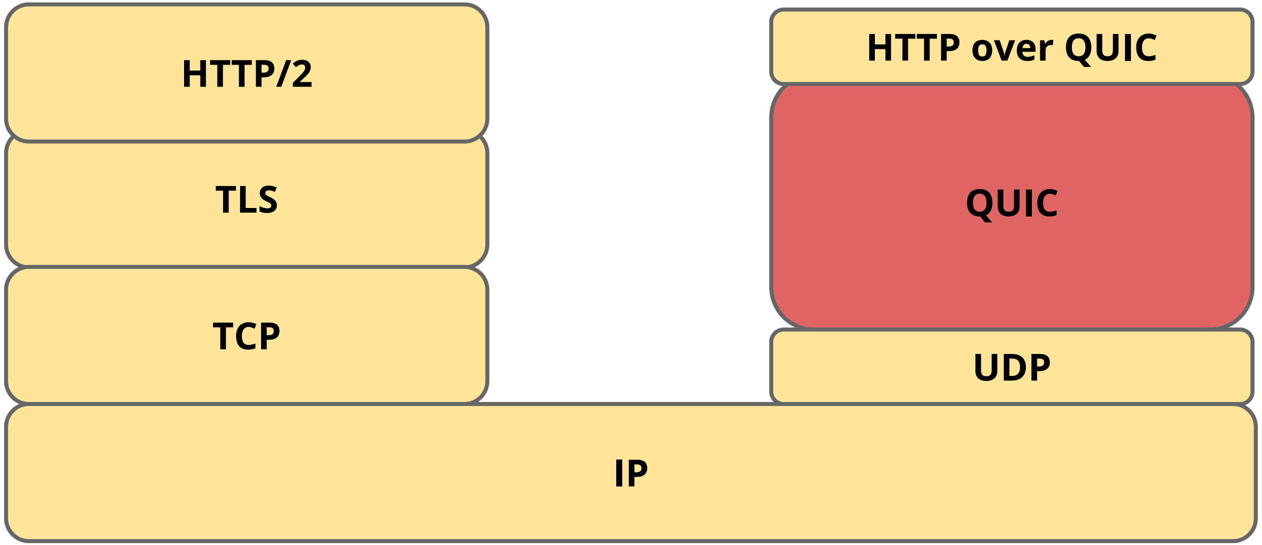
Google's QUIC deployment



What are we talking about?



What are we talking about?



Outline

QUIC design and experimentation

Metrics

Experiences

QUIC Design Goals (1 of 2)

Deployability and evolvability

in userspace, atop UDP

encrypted and authenticated headers

Low-latency secure connection establishment

mostly 0-RTT, sometimes 1-RTT

(similar to TCP Fast Open + TLS 1.3)

Streams and multiplexing

lightweight abstraction within a connection

avoids head-of-line blocking in TCP

QUIC Design Goals (2 of 2)

Better loss recovery and flexible congestion control

unique packet number, receiver timestamp

Resilience to NAT-rebinding

64-bit connection ID

also, connection migration and multipath

Hang on … some of this sounds familiar

We've replayed hits from the 1990s and 2000s...
(TCP Session, CM, SCTP, SST, TCP Fast Open ...)

… and added some new things

Experimentation Framework

Using Chrome

randomly assign users into experiment groups
experiment ID on requests to server
client and server stats tagged with experiment ID

Novel development strategy for a transport protocol

the Internet as the testbed
measure value before deploying *any* feature
rapid disabling when something goes wrong

Measuring Value

Applications drive transport adoption

app metrics define what app cares about

small changes directly connected to revenue

("end-to-end" metrics --- include non-network components)

Performance as improvements (average and percentiles)

percentiles: rank samples in increasing order of metric

interesting behavior typically in tails

Application Metrics

Search Latency

user enters search term --> entire page is loaded

Video Playback Latency

user clicks on cat video --> video starts playing

Video Rebuffer Rate

rebuffer time / (rebuffer time + video play time)

Search and Video Latency

		% latency reduction by percentile						
		Lower latency						Higher latency
	Mean	1%	5%	10%	50%	90%	95%	99%
Search								
Desktop								
Mobile								

Search and Video Latency

	% latency reduction by percentile							
		Lower latency					Higher latency	
	Mean	1%	5%	10%	50%	90%	95%	99%
Search								
Desktop	8.0							
Mobile	3.6							

Search and Video Latency

		% latency reduction by percentile						
		Lower latency						Higher latency
	Mean	1%	5%	10%	50%	90%	95%	99%
Search								
Desktop	8.0	0.4	1.3	1.4	1.5			
Mobile	3.6	-0.6	-0.3	0.3	0.5			

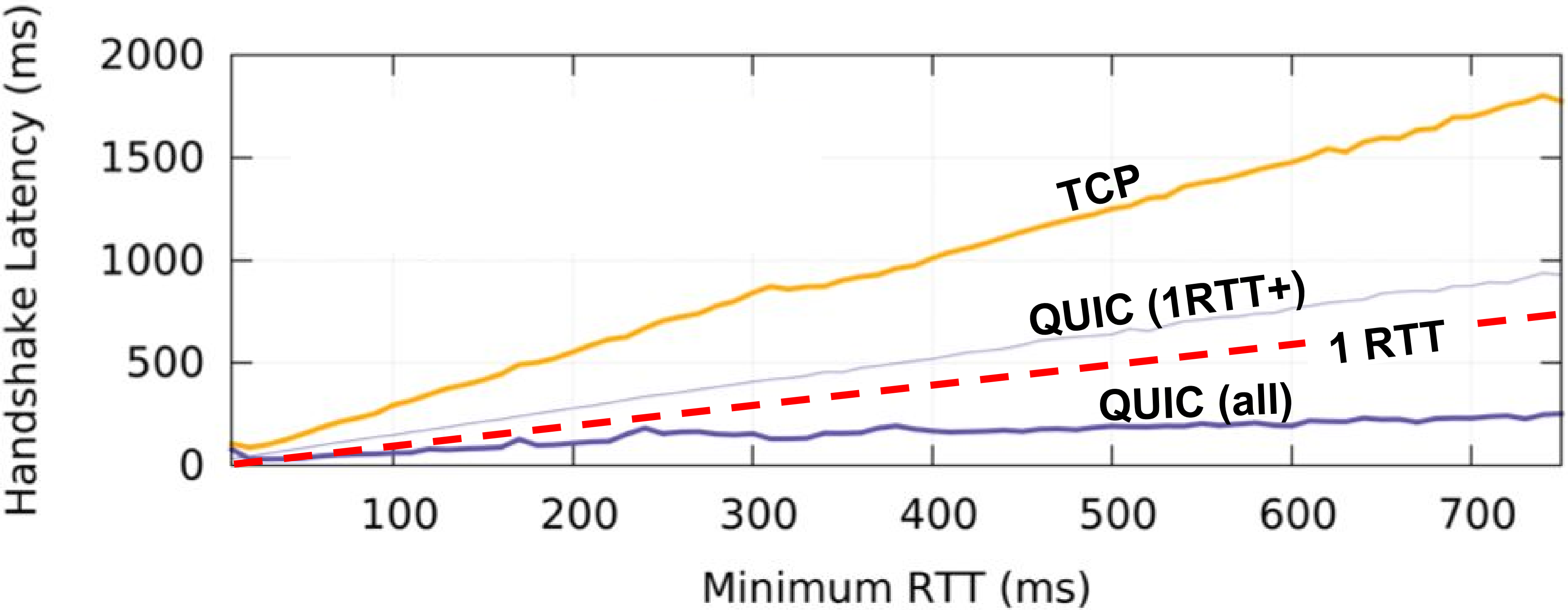
Search and Video Latency

		% latency reduction by percentile						
		Lower latency						Higher latency
	Mean	1%	5%	10%	50%	90%	95%	99%
Search								
Desktop	8.0	0.4	1.3	1.4	1.5	5.8	10.3	16.7
Mobile	3.6	-0.6	-0.3	0.3	0.5	4.5	8.8	14.3

Search and Video Latency

		% latency reduction by percentile						
		Lower latency						Higher latency
	Mean	1%	5%	10%	50%	90%	95%	99%
Search								
Desktop	8.0	0.4	1.3	1.4	1.5	5.8	10.3	16.7
Mobile	3.6	-0.6	-0.3	0.3	0.5	4.5	8.8	14.3
Video								
Desktop	8.0	1.2	3.1	3.3	4.6	8.4	9.0	10.6
Mobile	5.3	0.0	0.6	0.5	1.2	4.4	5.8	7.5

Why is app latency lower?



Video Rebuffer Rate

		% rebuffer rate reduction by percentile				
		Fewer rebuffers				More rebuffers
	Mean	< 93%	93%	94 %	95%	99%
Desktop						
Mobile						

Video Rebuffer Rate

		% rebuffer rate reduction by percentile				
		Fewer rebuffers				More rebuffers
	Mean	< 93%	93%	94 %	95%	99%
Desktop	18.0					
Mobile	15.3					

Video Rebuffer Rate

		% rebuffer rate reduction by percentile				
		Fewer rebuffers				More rebuffers
	Mean	< 93%	93%	94 %	95%	99%
Desktop	18.0	*				
Mobile	15.3	*				

Video Rebuffer Rate

		% rebuffer rate reduction by percentile				
		Fewer rebuffers				More rebuffers
	Mean	< 93%	93%	94 %	95%	99%
Desktop	18.0	*	100.0			
Mobile	15.3	*	*			

Video Rebuffer Rate

		% rebuffer rate reduction by percentile				
		Fewer rebuffers				More rebuffers
	Mean	< 93%	93%	94 %	95%	99%
Desktop	18.0	*	100.0	70.4	60.0	18.5
Mobile	15.3	*	*	100.0	52.7	8.7

QUIC Improvement by Country

			% Reduction in Rebuffer Rate	
Country	Mean Min RTT (ms)	Mean TCP Rtx %	Desktop	Mobile
South Korea	38	1	0.0	10.1

QUIC Improvement by Country

			% Reduction in Rebuffer Rate	
Country	Mean Min RTT (ms)	Mean TCP Rtx %	Desktop	Mobile
South Korea	38	1	0.0	10.1
USA	50	2	4.1	12.9

QUIC Improvement by Country

			% Reduction in Rebuffer Rate	
Country	Mean Min RTT (ms)	Mean TCP Rtx %	Desktop	Mobile
South Korea	38	1	0.0	10.1
USA	50	2	4.1	12.9
India	188	8	22.1	20.2

Why is video rebuffer rate lower?

Better loss recovery in QUIC

unique packet number avoids retransmission ambiguity

TCP receive window limits throughput

4.6% of connections are limited

Experiments and Experiences: Network Ossification

Firewall used *first byte* of packets for QUIC classification

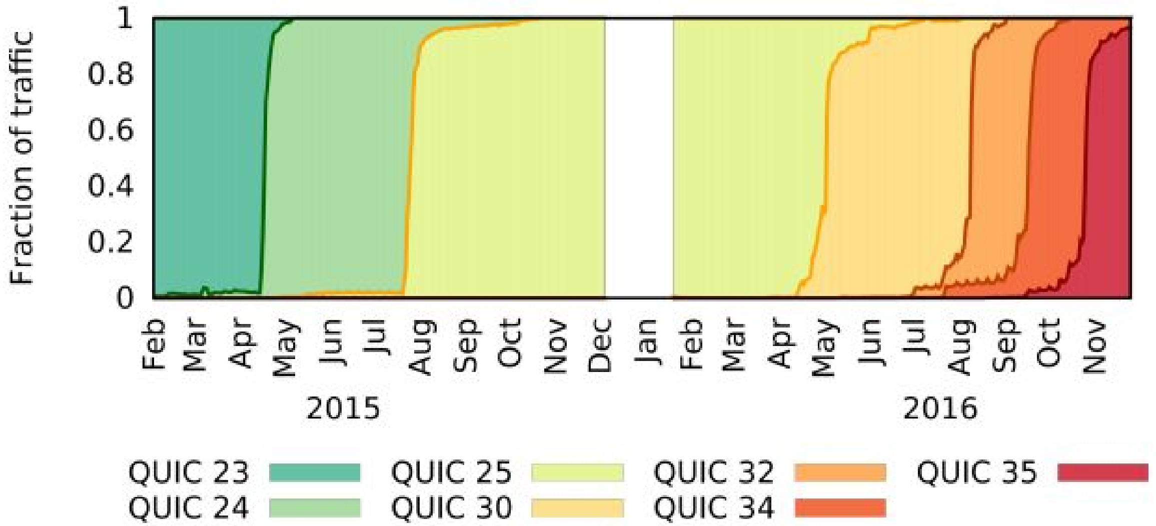
- *flags* byte, was 0x07 at the time
- broke QUIC when we flipped a bit

"the ultimate defense of the end to end mode is end to end encryption" *-- D. Clark, J. Wroclawski, K. Sollins, and R. Braden **

* Tussle in Cyberspace: Defining Tomorrow's Internet. IEEE/ACM ToN, 2005.

Experiments and Experiences: Userspace development

- Better practices and tools than kernel
- Better integration with tracing and logging infrastructure
- Rapid deployment and evolution



Extra slides

Experiments and Experiences: FEC in QUIC

Simple XOR-based FEC in QUIC

- 1 FEC packet per protected group
- Timing of FEC packet and size of group controllable

Conclusion: Benefits not worth the pain

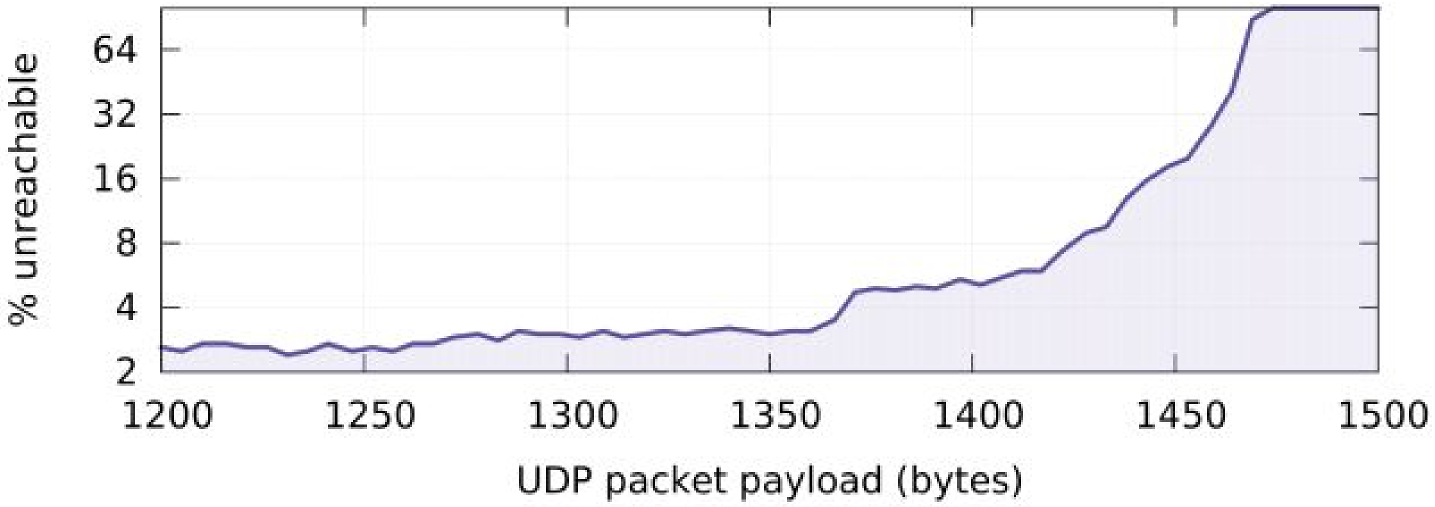
- Multiple packet losses within RTT common
- FEC implementation extremely invasive
- Gains really at tail, where aggressive TLP wins

Experiments and Experiences: UDP Blockage

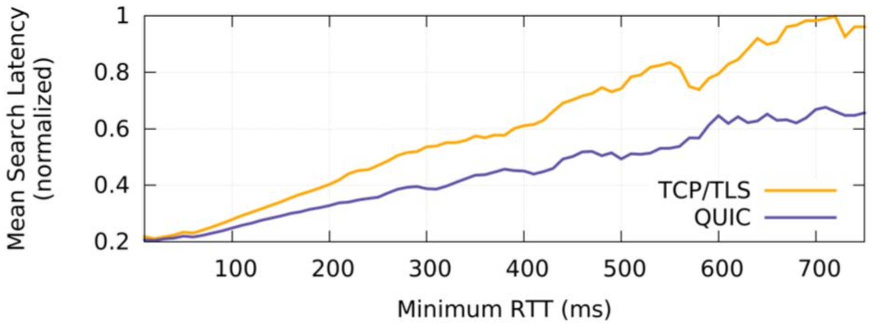
- QUIC successfully used: 95.3% of clients
- Blocked (or packet size too large): 4.4%
- QUIC performs poorly: 0.3%
  - Networks that rate limit UDP
  - Manually turn QUIC off for such ASes

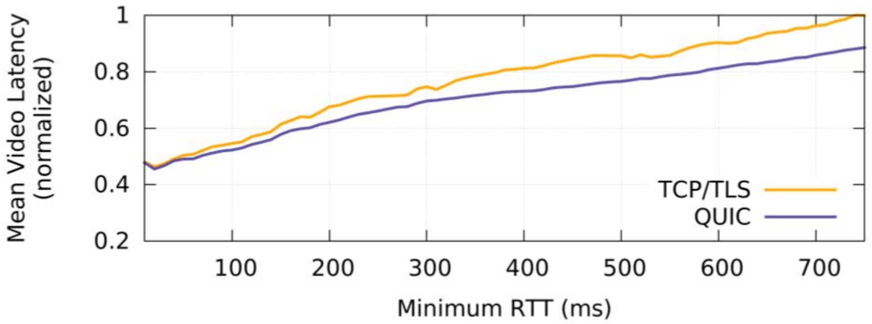
Experiments and Experiences: Packet Size Considerations

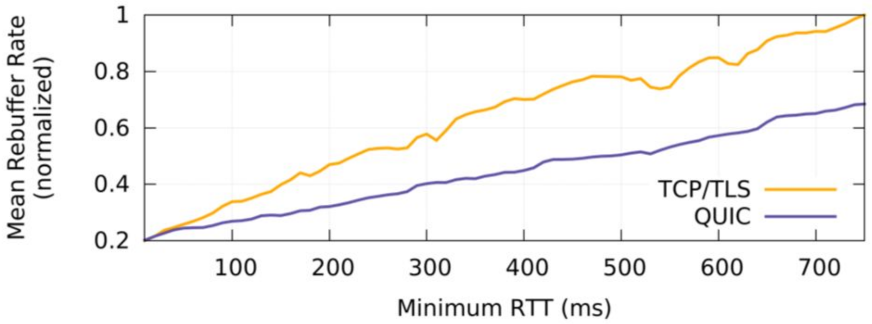
- UDP packet train experiment, send and echo packets
- Measure reachability from Chrome users to Google servers



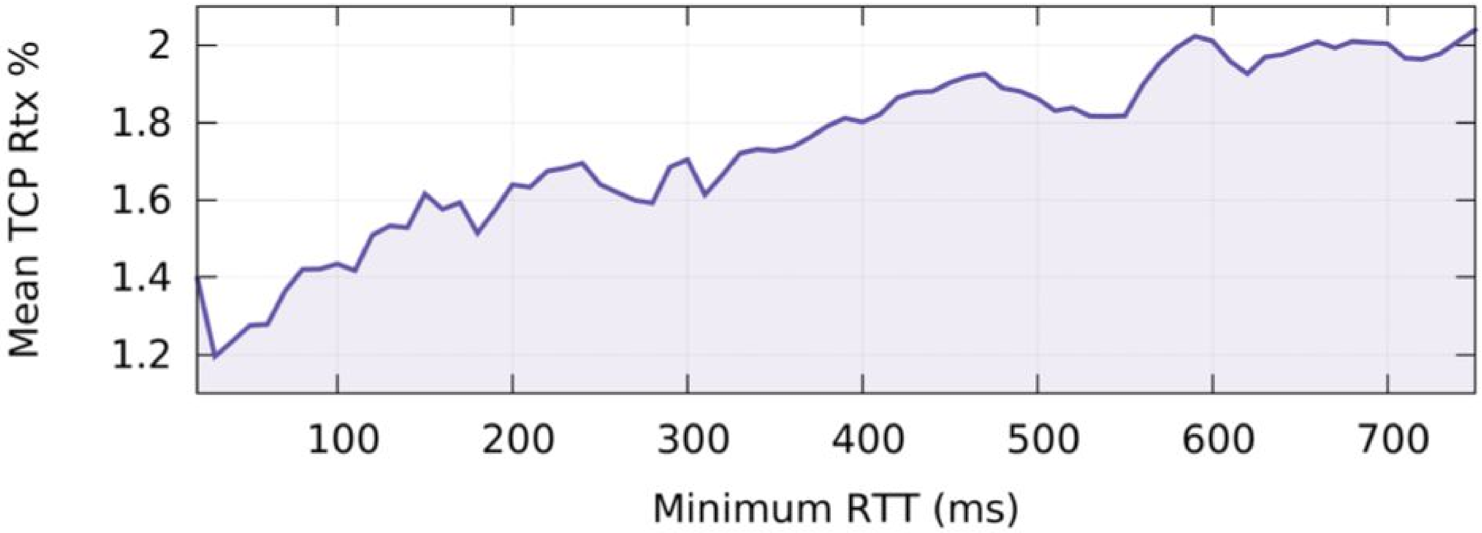
All metrics improve more as RTT increases ...



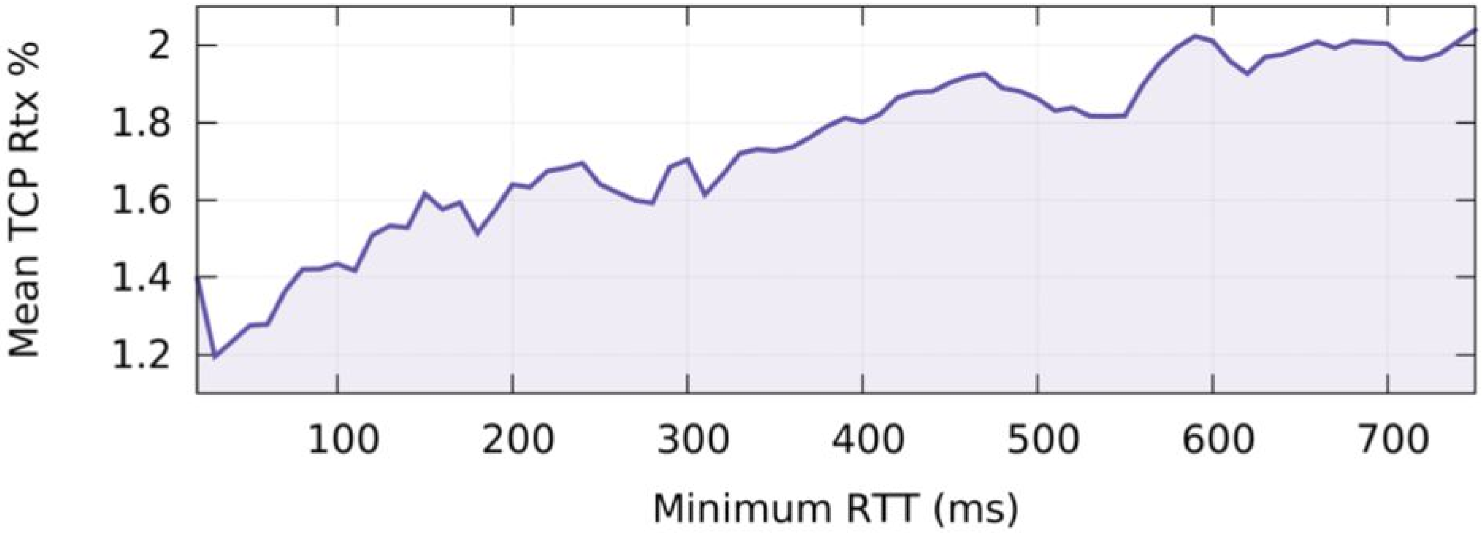




Network loss rate increases with RTT



Network loss rate increases with RTT



Reason 1: QUIC's improved loss recovery helps more with increased RTT and loss rate

Reason 2: TCP receive window limit

4.6% of connections have
server's max cwnd == client's max rwnd