The QUIC Transport Protocol: Design and Internet-Scale Deployment

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

To appear in ACM SIGCOMM, August 2017

A QUIC history

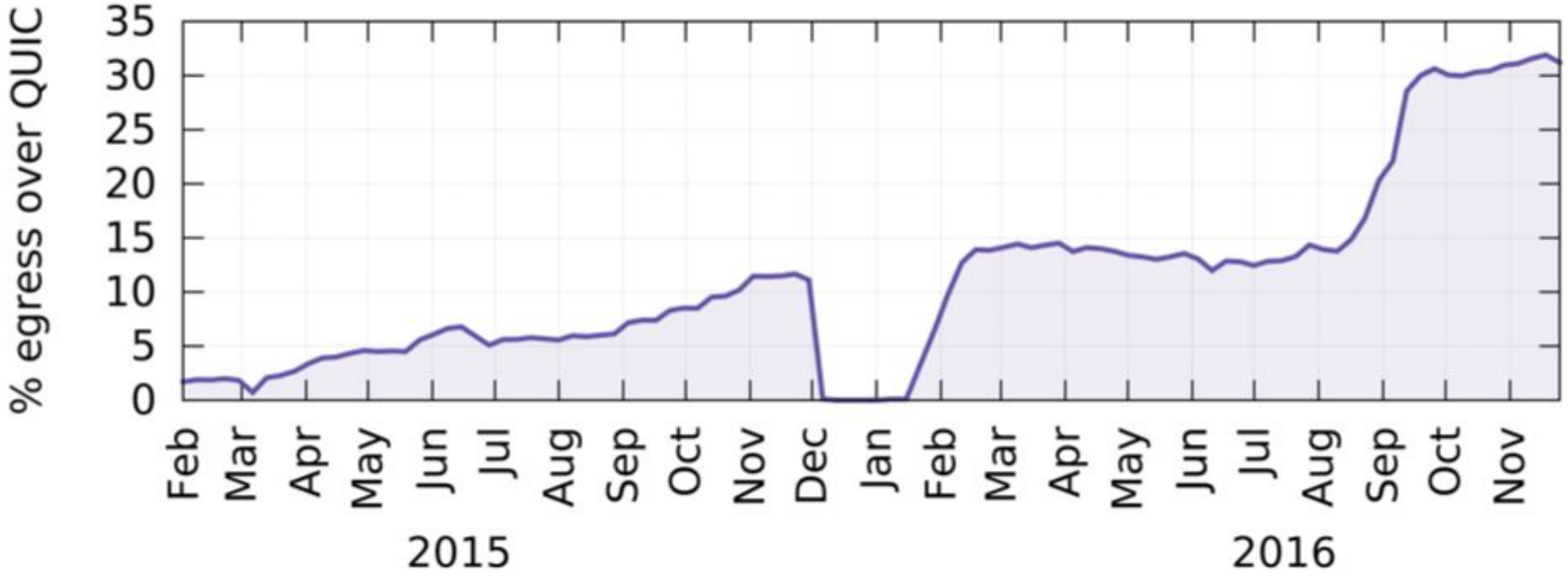
Protocol for HTTP transport, deployed at Google starting 2014

- Between Google services and Chrome / mobile apps
- Reduced page-load latency and video rebuffers
  - YouTube Video Rebuffers: 15 - 18%
  - Google Search Latency: 3.6 - 8%
- 35% of Google's traffic (7% of Internet)

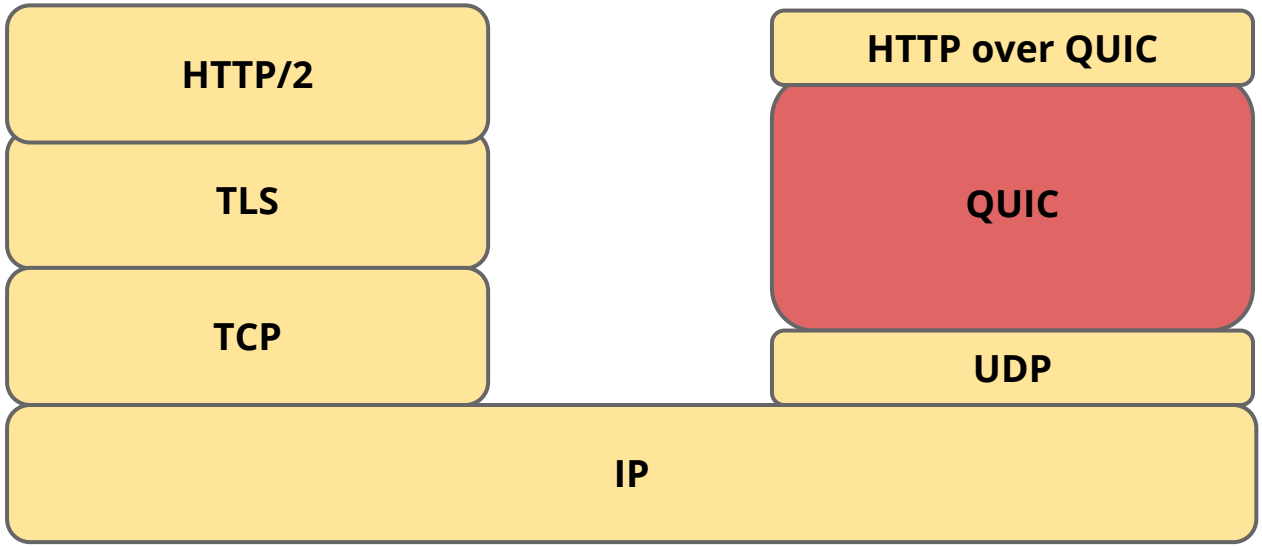
IETF QUIC working group formed in Oct 2016

- Modularize and standardize QUIC

Google's QUIC deployment



What are we talking about?



QUIC Design Aspirations

- Deployability and evolvability
- Low latency connection establishment
  - mostly 0-RTT, sometimes 1-RTT
- Multistreaming and per-stream flow control
- Better loss recovery and flexible congestion control
  - Richer signaling (unique packet number)
  - Better RTT estimates
- Resilience to NAT-rebinding

Metrics

- Latency
  - Search
  - Video Playback
- Video Rebuffer Rate
- Application-defined metrics
  - Matter to apps, drive adoption
  - Include non-network components

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

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

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

Handshake Latency

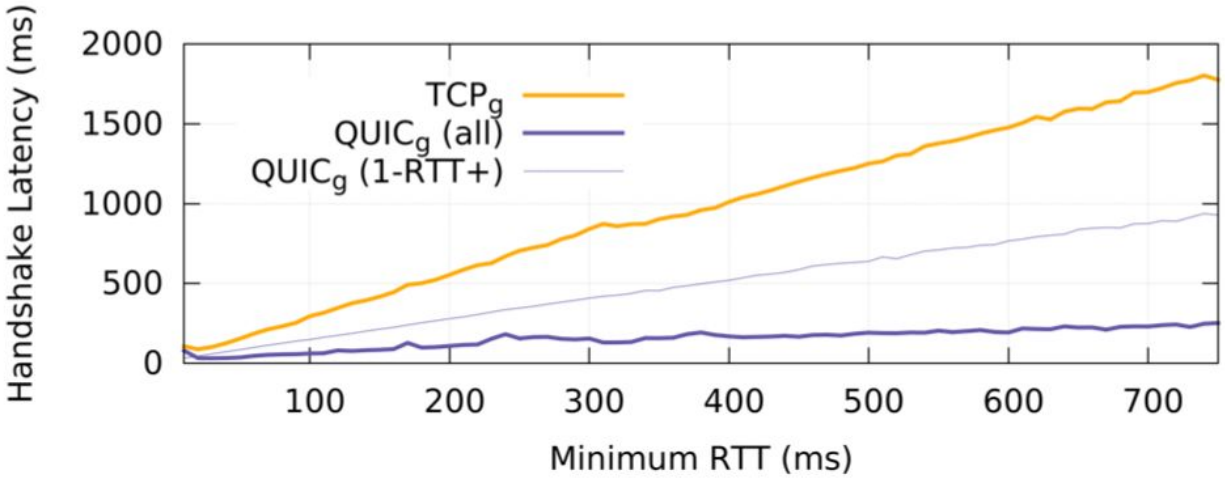
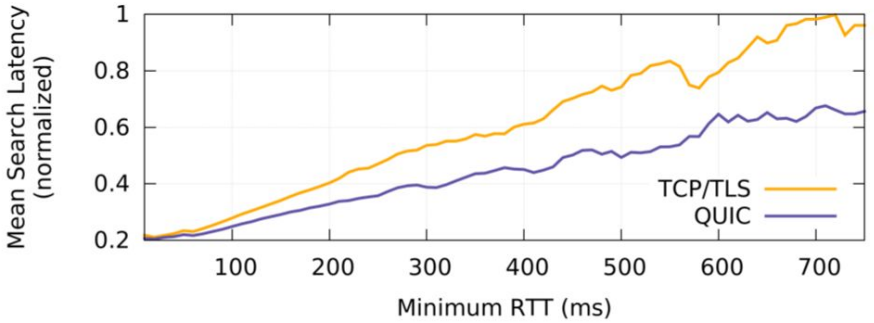


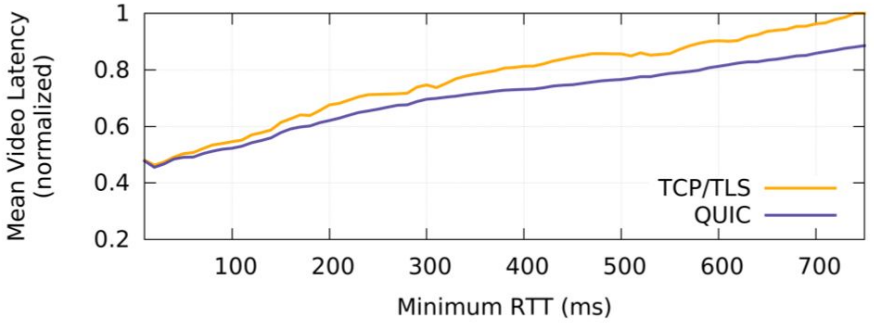
Figure 7: Comparison of handshake latency for $QUIC_g$ and TCP_g versus the minimum RTT of the connection. Solid lines indicate the mean handshake latency for all connections, including 0-RTT connections. The dashed line shows the handshake latency for only those $QUIC_g$ connections that did not achieve a 0-RTT handshake. Data shown is for Desktop connections, mobile connections look similar.

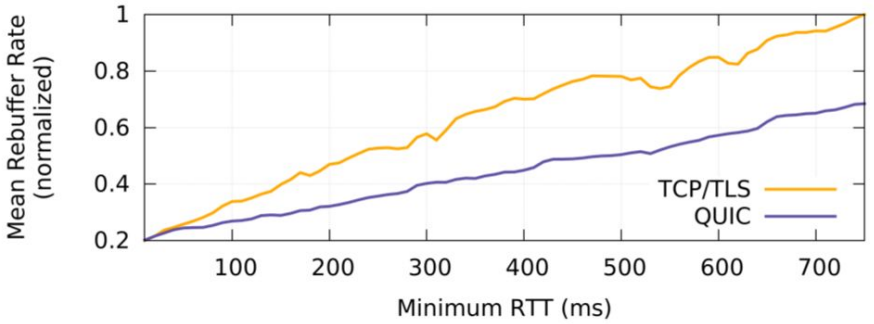
Video Rebuffer Rate

		% 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

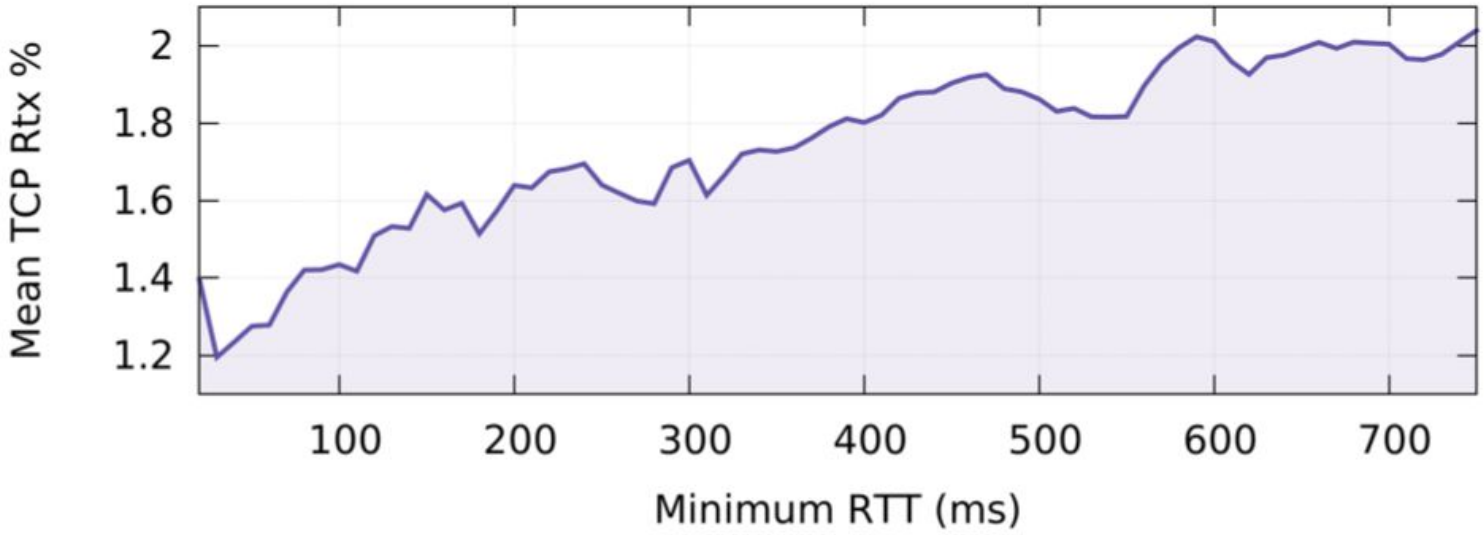
All metrics improve more as RTT increases ...





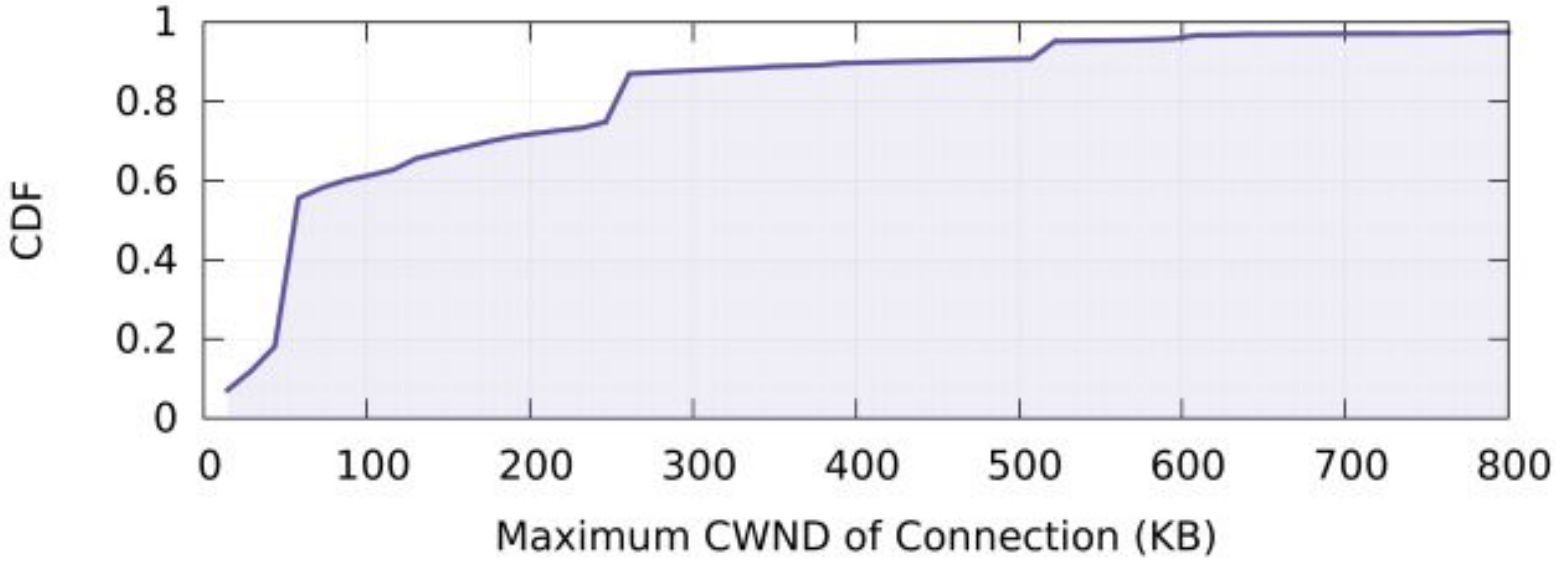


Network loss rate increases with RTT



TCP receive window limit

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



QUIC improvement by country

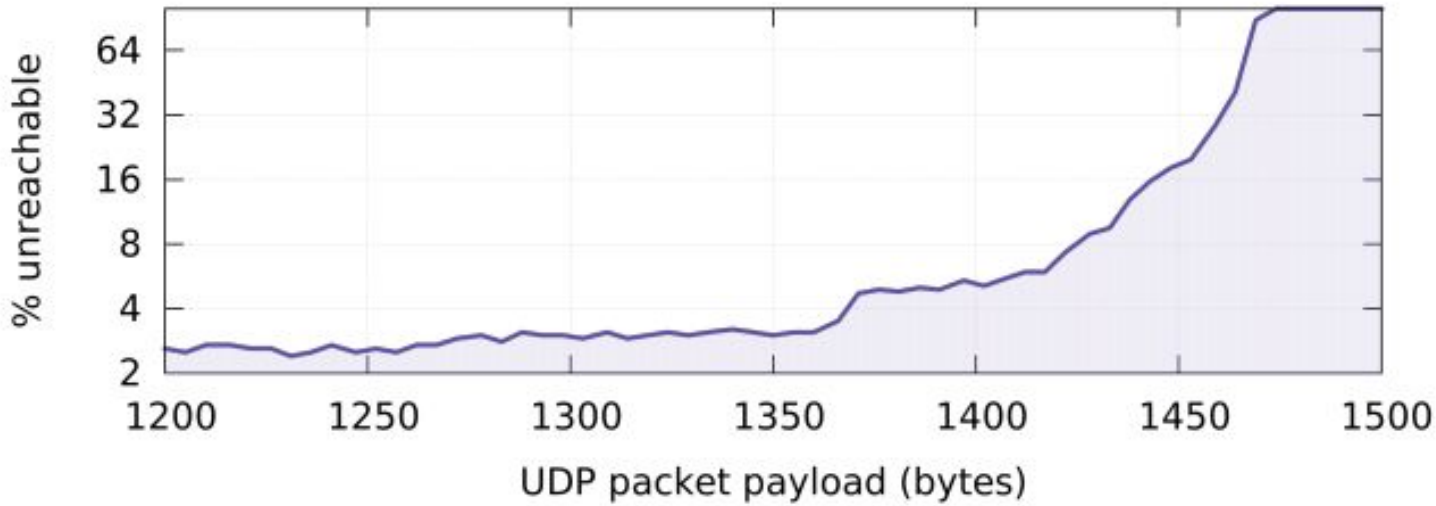
			% Reduction in Search Latency		% Reduction in Rebuffer Rate	
Country	Mean Min RTT (ms)	Mean TCP Rtx %	Desktop	Mobile	Desktop	Mobile
South Korea	38	1	1.3	1.1	0.0	10.1
USA	50	2	3.4	2.0	4.1	12.9
India	188	8	13.2	5.5	22.1	20.2

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



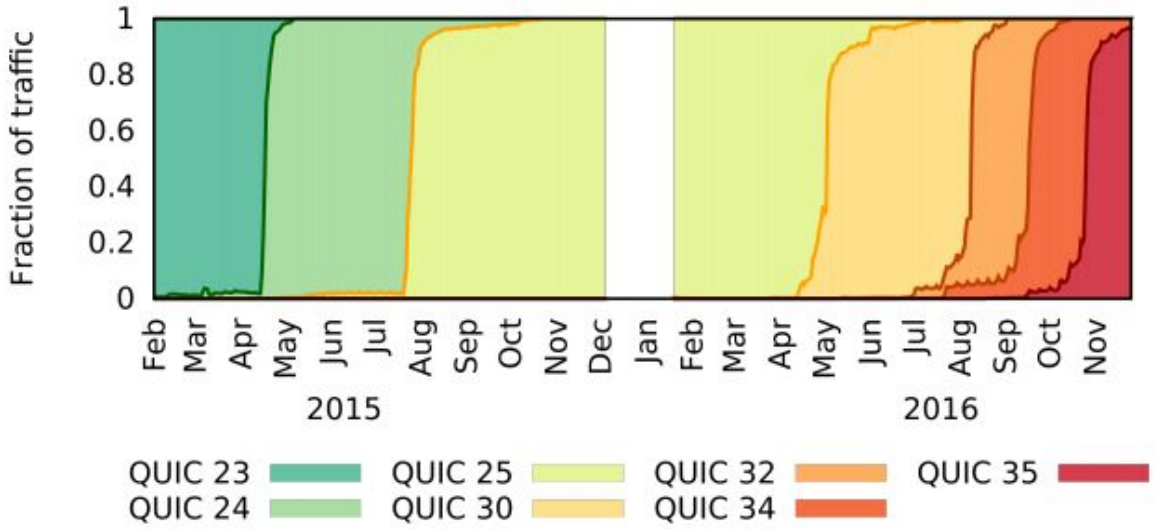
Experiments and Experiences: FEC in QUIC

- 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: Userspace development

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



Experiments and Experiences: Network Ossification

- Middlebox ossification
  - Vendor ossified *first byte* of QUIC packets (flags byte)
  - … since it seemed to be the same on all QUIC packets
  - Broke QUIC deployment when a flag was flipped

  Encryption is the only protection against network ossification

- Userspace development
  - Has better practices and tools than kernel
  - Better integration with tracing and logging infrastructure