QUIC Experiments
QUIC
Quick UDP Internet Connections

- A reliable, multiplexed transport over UDP
- Always encrypted
- Reduces latency
- Runs in user-space
- Open sourced in Chromium
What is QUIC?

New transport designed to reduce web latency

- TCP + TLS + SPDY over UDP
- Faster connection establishment than TLS/TCP
  - 0-RTT usually, 1-RTT sometimes
- Deals better with packet loss than TCP
- Has Stream-level and Connection-level Flow Control
- FEC recovery
- Multipath

*except for HTTP/2 headers, which should be fixed as well.
Where does it fit?

HTTP/2
TLS 1.2
TCP
IP

HTTP/2 API
QUIC
UDP
Experiments

Conducted at scale between Chrome and Google servers (including GGC)

Most experiments are enabled with tags in the COPT field of the CHLO. See [crypto_protocol.h](#) in Chromium for a list of tags.
0-RTT’s impact

- About 75% connections are 0-RTT connections
- Accounts for between 50 to 80% of the median latency improvements
- 0-RTT helps more when Chrome’s pre-connect isn’t able to predict the host
- No significant effect on other transport stats
Connection Pooling

- QUIC’s connection pooling is equivalent to HTTP/2’s
- Improves latency about 10% vs disabling it
- No latency metrics were worse
- Could be improved with better connection pooling via Alt-Svc
Packet Pacing

- Similar to Linux kernel’s fq qdisc
- Pacing does
  - improve tail page load latency
  - reduce retransmits ~25%
- Pacing does not
  - change median page load latency
  - change YouTube QoE
**IW10 vs IW32**

- QUIC defaults to 32, similar to HTTP/2 default
- 30% of QUIC’s "time to playback" gains for YouTube due to IW32
- IW10 had equal or slightly worse latency, even at the 95%
- IW10 decreased retransmit rate slightly
  - *IW10 without pacing had higher retransmit rate than IW32 with pacing*
- (Invoked with IW10 connection option. IW03, IW20, and IW50 also available)
Reno vs Cubic

- QUIC defaults to Cubic, similar to Linux
- Latency across all services is extremely similar between Reno and Cubic
- QoE is extremely similar between Reno and Cubic
- Retransmits are ~20% lower with Reno than Cubic
- (Reno available with the RENO connection option)

Why not default to Reno?

We’re thinking about it...
1 vs 2 Connection Emulation

- QUIC defaults to 2-connection emulation
- 2-connection shows large improvements in YouTube QoE
- 1- vs 2-connection has a negligible effect on median page load latency
  - 2-connection shows slight improvement in tail latency
- Retransmits are 20% higher with 2-connection
Tail Loss Probe

- QUIC defaults to 2 TLPs before RTO
- Disabling TLP has no effect on median latency
- TLP improves 95% latency almost 1%
- TLP Improves YouTube rebuffer rate almost 1%
- Disabling TLP reduces retransmits 5%
Time based loss detection

- QUIC defaults to FACK with a fixed dupack threshold of 3
- Time-based loss detection waits ¼ RTT after the first NACK for the packet to be lost
- Shows no significant improvements vs FACK on user-facing networks