QUIC
Next generation multiplexed transport over UDP
How do you make the web faster?

User-perceived latency

$BROWSER

HTTP/1.1

TLS 1.2

TCP

IP

Physical Network

google.com
How do you make the web faster?

User-perceived latency

- $BROWSER
- HTTP/1.1
- TLS 1.2
- TCP
- IP
- Physical Network
- google.com

Build a carrier-grade network

Google CDN

google.com
How do you make the web faster?

- Launch your own browser
- Update HTTP
- Build a carrier-grade network
- Google CDN

$BROWSER
HTTP/1.1
TLS 1.2
TCP
IP
Physical Network
goOGLE.com

Chrome
HTTP/2

User-perceived latency
How do you make the web faster?

- Launch your own browser
- Update HTTP
- Build a carrier-grade network
- Update transport

BROWSER
- HTTP/1.1
- TLS 1.2
- TCP
- IP
- Physical Network
- google.com

Chrome
- HTTP/2
- ???
- Google CDN
- google.com
What is QUIC?
QUIC
Quick UDP Internet Connections

- A reliable, multiplexed transport over UDP
- Always encrypted
- Reduces latency
- Runs in user-space
- Open sourced in Chromium
Where does it fit?

- HTTP/2
- TLS 1.2
- TCP
- UDP
- QUIC
- IP
- HTTP/2 API
QUIC Works Great™ when...

You treat UDP like TCP:
- UDP port 443 open
- No UDP rate-limits
- No worse UDP QoS treatment
- Reasonable stateful FW/NAT timeouts
- 5-tuple traffic load balancing
Congestion control & reliability

QUIC builds on decades of experience with TCP

Incorporates TCP best practices
   TCP Cubic - fair with TCP
   FACK, TLP, F-RTO, Early Retransmit...

Adds signaling improvements that can’t be done to TCP
   Loss detection - retransmission uses a new sequence number

More flexibility going forward
   Improved congestion feedback, control over acking
Zero-RTT connection establishment

TCP

TCP + TLS

QUIC

(equivalent to TCP + TLS)
Always encrypted

Comparable to TLS
   Perfect forward secrecy, with more efficient handshake

IP spoofing protection
   Signed proof of address

Inspired TLS 1.3’s zero RTT handshake
   Plan to adopt TLS 1.3 when complete
Effective

How quick is QUIC?
Measuring performance

**Controlled Experiments**

**Client Side**
- Latency, Bandwidth, Quality of Experience, Errors

**Server Side**
- Latency, Bandwidth, QUIC Success Rate

**Fine Grained Analysis**
- By ASN, Server, OS, Version

**Transparency**
- ISP view on peering.google.com
Performance on Google properties

Faster page loading times
- 5% faster on average
- 1 second faster for web search at 99th-percentile

Improved YouTube Quality of Experience
- 30% fewer rebuffers (video pauses)

More improvements to come
- Bandwidth resumption, forward error correction, etc

Recent Blog Post
Where are the gains from?

Zero-RTT
- Over 50% of the latency improvement (at median and 95th-percentile)

Improved loss recovery
- Over 10x fewer timeout based retransmissions improve tail latency and YouTube video rebuffer rates

Other, smaller benefits
- e.g. head of line blocking, more efficient framing
Deployment timeline

Tested at scale, with millions of users
- Chrome Canary: June, 2013
- Chrome Stable: April, 2014
- Ramping up for Google traffic: January, 2015
Safe
What we’re doing to protect users and networks
Client-side protection

What if UDP is blocked?
● Chrome seamlessly falls back to HTTP/TCP

What if the path MTU is too small?
● QUIC handshake fails, Chrome falls back to TCP

What if a client doesn’t want to use QUIC?
● Chrome flag / administrative policy to disable QUIC
When client-side protection is not enough...

As a last resort, Google disables QUIC to specific ASNs
● This is used as a fallback to protocol features

Why do we disable QUIC delivery?
● Degraded quality of experience measured
● Indications of UDP rate limiting at peak times of day
● End user reports (via chromium.org)
QUIC on your network

Traffic Summary

9.05% of Google traffic to your network is delivered via QUIC

Hourly average, bytes per second

QUIC Traffic Throughout the Day

00:00 04:00 08:00 12:00 16:00 20:00

Readiness Checklist

Your network is successfully serving QUIC traffic! Expect to see a growing proportion of QUIC in the coming months. Check the QUIC FAQ for answers to any questions about the rollout.

Get access at peering.google.com/quicfaq
Debugging Tools: Chrome

**chrome://net-internals**
- Active QUIC sessions
- Captures all events
- Important for filing Chromium bugs
Debugging Tools: Wireshark

Parses

- Protocol: QUIC
- CID: Connection ID
- Seq: Sequence number
- Version: ie: Q024
- Public flags: 1 byte
- Payload: Encrypted
What’s Next?
Future Improvements

- Forward Error Correction
- Connection Mobility
- Multipath
- Congestion Control
Open source implementations

Servers
- Open source test server included in Chromium
- Working to support QUIC in Apache Traffic Server

Clients
- Open source Chromium client library for desktop and mobile
- Google Chrome and some Google Android apps
- Working with other browsers.
QUIC at the IETF

Nov 2013  Initially Presented
Mar 2015  QUIC Crypto
July 2015  Updated presentation
Ongoing   Including Zero-RTT handshake in TLS 1.3
Review: QUIC Summary

- Reliable, multiplexed transport
- Runs over UDP
- Always encrypted
- Lower latency connection establishment
- Optional FEC
- Rapidly evolving user-space implementation
- Open source
Review: Providing Safe Passage

Treat UDP like TCP:
- UDP port 443 open
- No UDP rate-limits
- No differential UDP QoS
- Reasonable stateful FW/NAT timeouts
- Sensible hash-based traffic distribution
ISP Resources for QUIC: peering.google.com/quicfaq

Ian Swett
ianswett@google.com