



# QUIC

## Redefining Internet Transport

**Presenter:** Jana Iyengar



# QUIC

Reinventing? Internet Transport

Reinventing?

**Presenter:**

Jana Iyengar



# QUIC

Redesigning Internet Transport

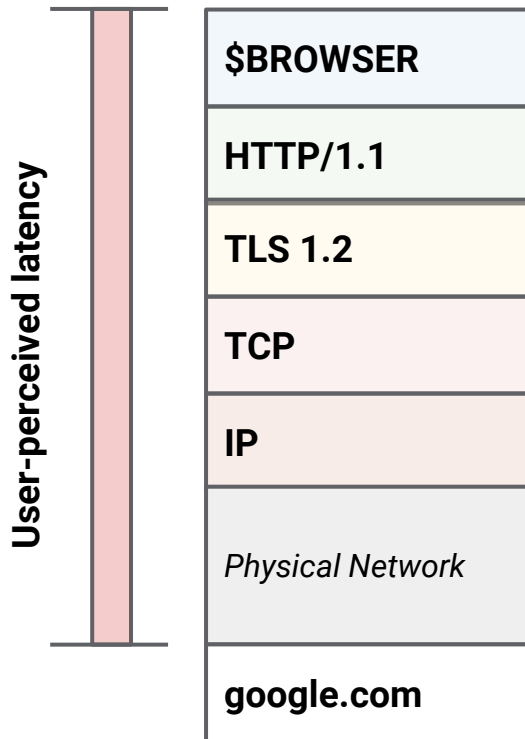
**Doing**

**Right!**

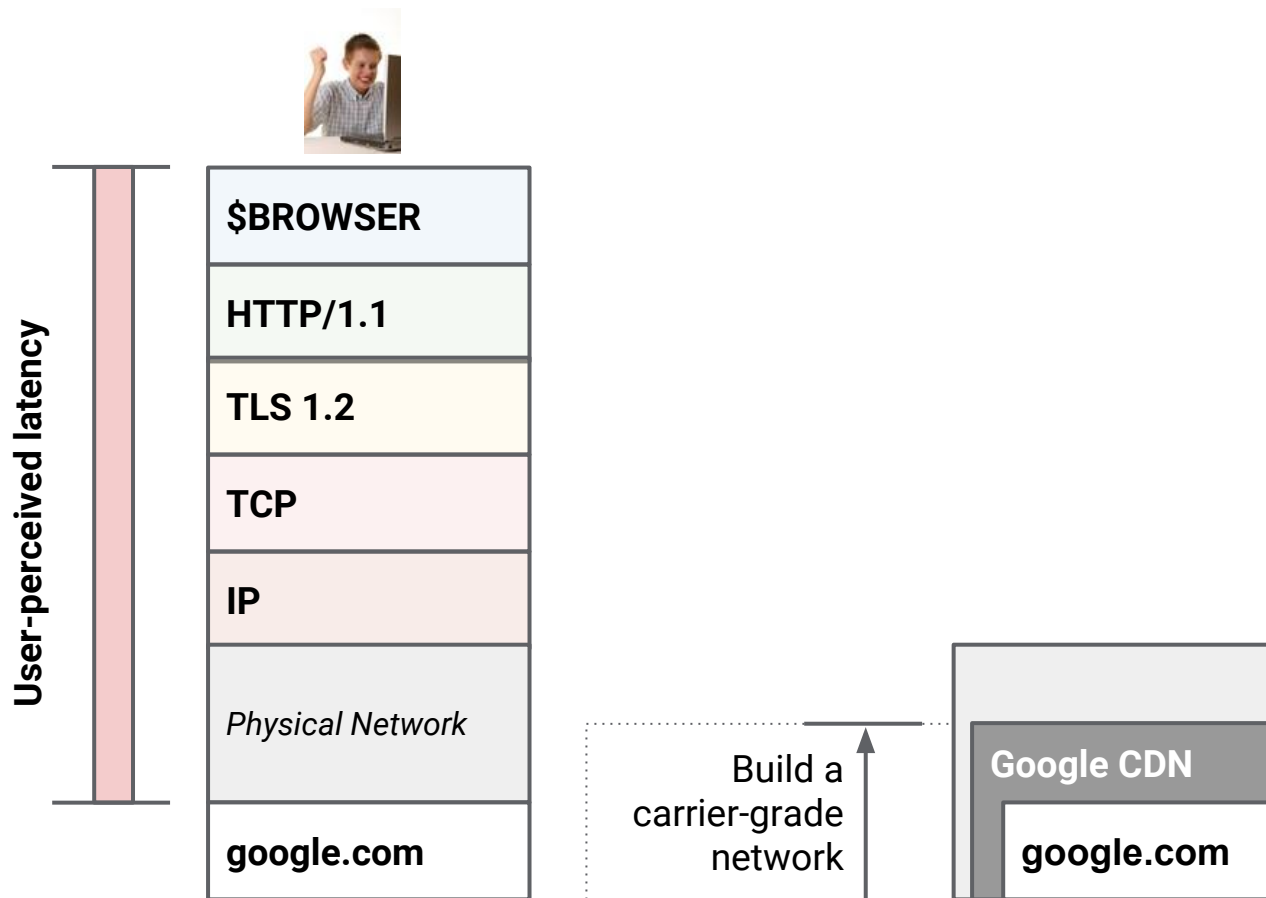
**Presenter:**

Jana Iyengar

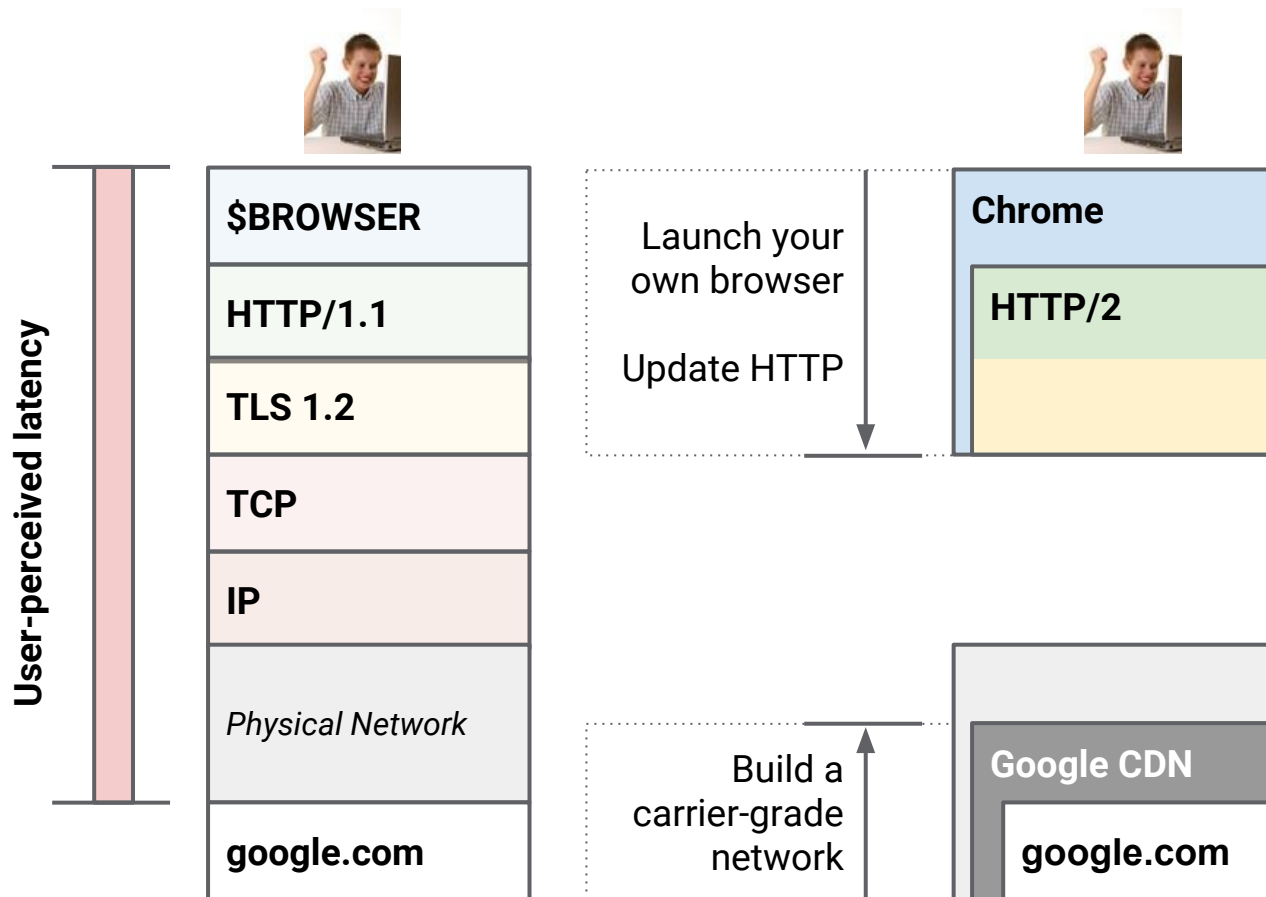
# How do you make the web faster?



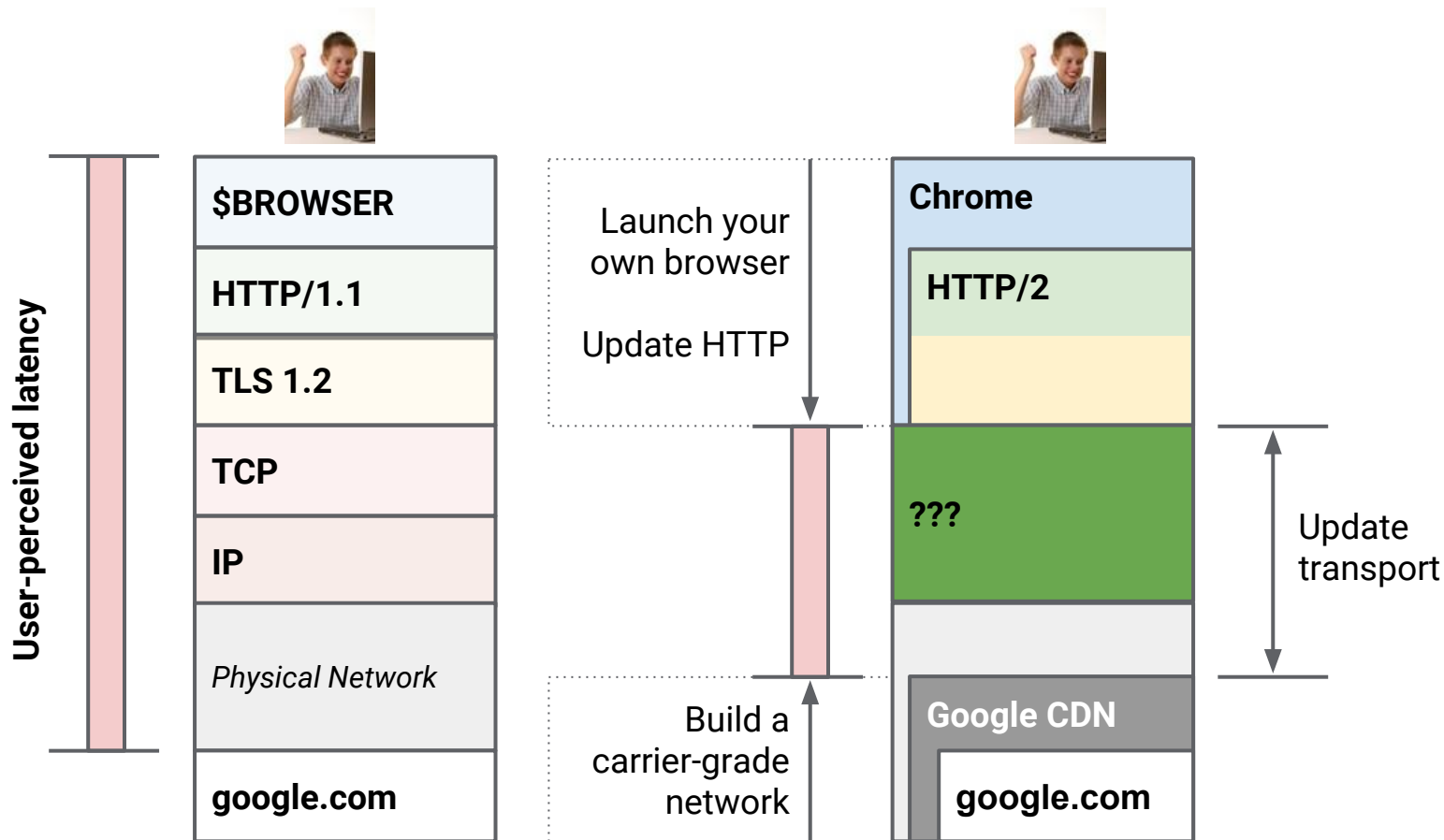
# How do you make the web faster?



# How do you make the web faster?



# How do you make the web faster?



# 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

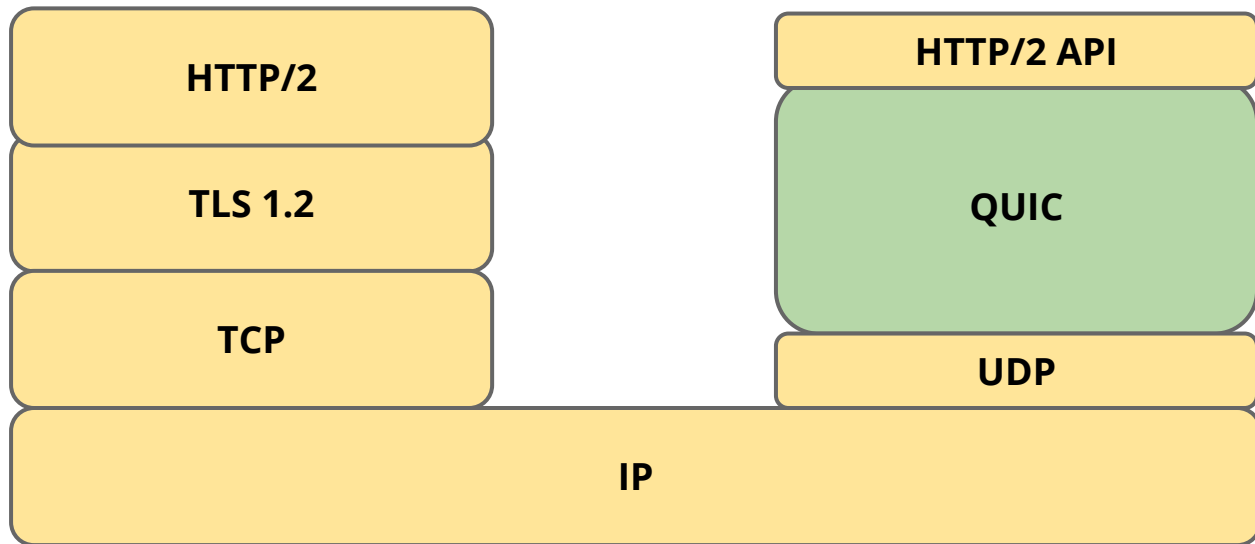
# What is QUIC?

## **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?



# Always encrypted

## Comparable to TLS

Perfect forward secrecy, with more efficient handshake

## IP spoofing protection

Signed proof of address

## Inspired TLS 1.3's 0-RTT handshake

Plan to adopt TLS 1.3 when complete

[more crypto details...](#)

# Connection establishment

## Connection identified by Connection ID

- As opposed to common 5-tuple
- 64 bits
- Chosen randomly by the client
- Enables connection mobility across IP, port

# 0-RTT connection establishment

## TCP

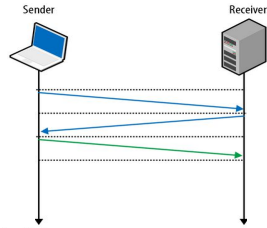


Figure 2-1. Three-way handshake

## TCP + TLS

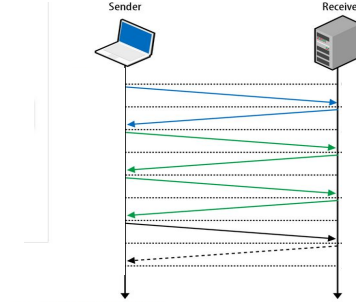
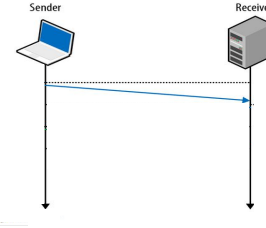


Figure 4-2. TLS handshake protocol

## QUIC (equivalent to TCP + TLS)



# First-ever connection - 1 RTT

**No cached information available**

**First CHLO is inchoate (empty)**

Simply includes version and server name

**Server responds with REJ**

Includes server config, certs, etc

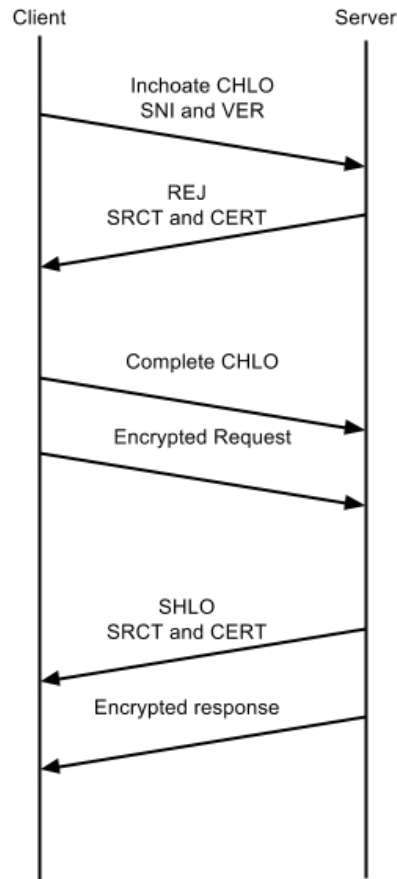
Allows client to make forward progress

**Second CHLO is complete**

Followed by initially encrypted request data

**Server responds with SHLO**

Followed immediately by forward-secure encrypted response data



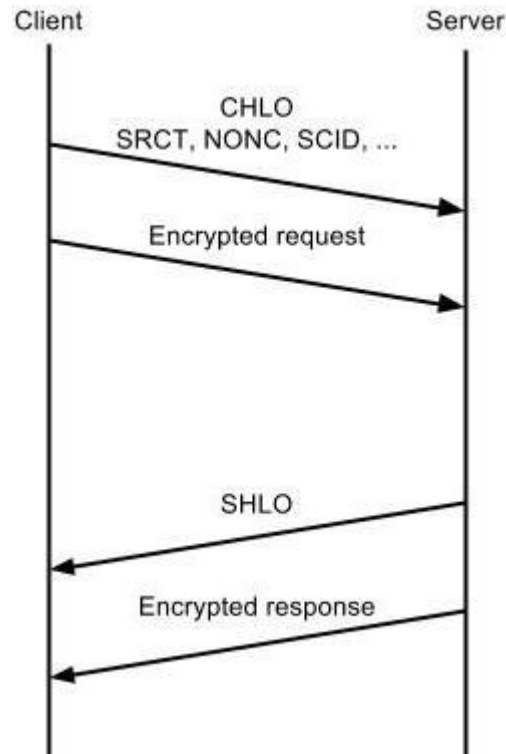
# Subsequent connections - 0 RTT

## First CHLO is complete

Based on information from previous connection  
Followed by initially encrypted data.

## Server responds with SHLO

Followed immediately by forward-secure encrypted data





# Congestion control & reliability

**QUIC builds on decades of experience with TCP**

**Incorporates TCP best practices**

TCP Cubic - fair with TCP

FAACK, TLP, F-RTO, Early Retransmit...

**More flexibility going forward**

Improved congestion feedback, control over acking

**Better signaling than TCP**

# Better signaling than TCP

## **Retransmitted packets consume new sequence number**

- No retransmission ambiguity

- Prevents loss of retransmission from causing RTO

## **More verbose ACK**

- TCP supports up to 3 SACK ranges

- QUIC supports up to 256 NACK ranges

- Per-packet receive times, even with delayed ACKs

## **ACK packets consume a sequence number**

# Effective

---

How quick is QUIC?

# Measuring performance



## Controlled Experiments

### Client Side

Latency, Bandwidth, Quality of Experience, Errors

### Server Side

Latency, Bandwidth, QUIC Success Rate

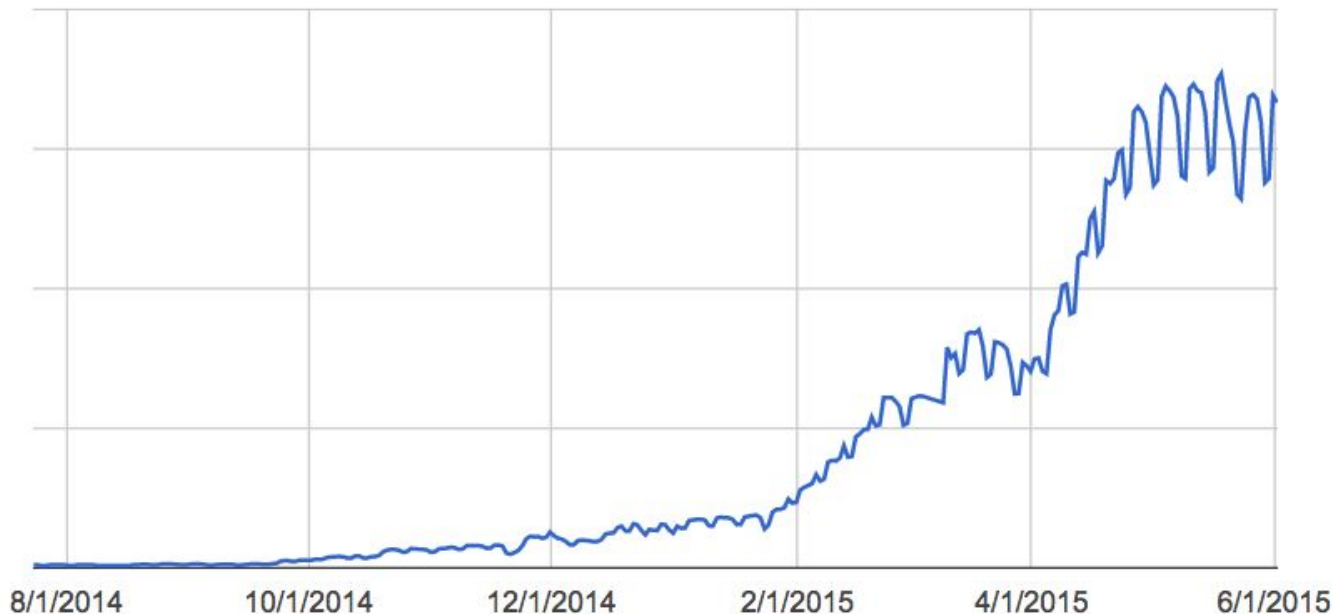
### Fine Grained Analysis

By ASN, Server, OS, Version

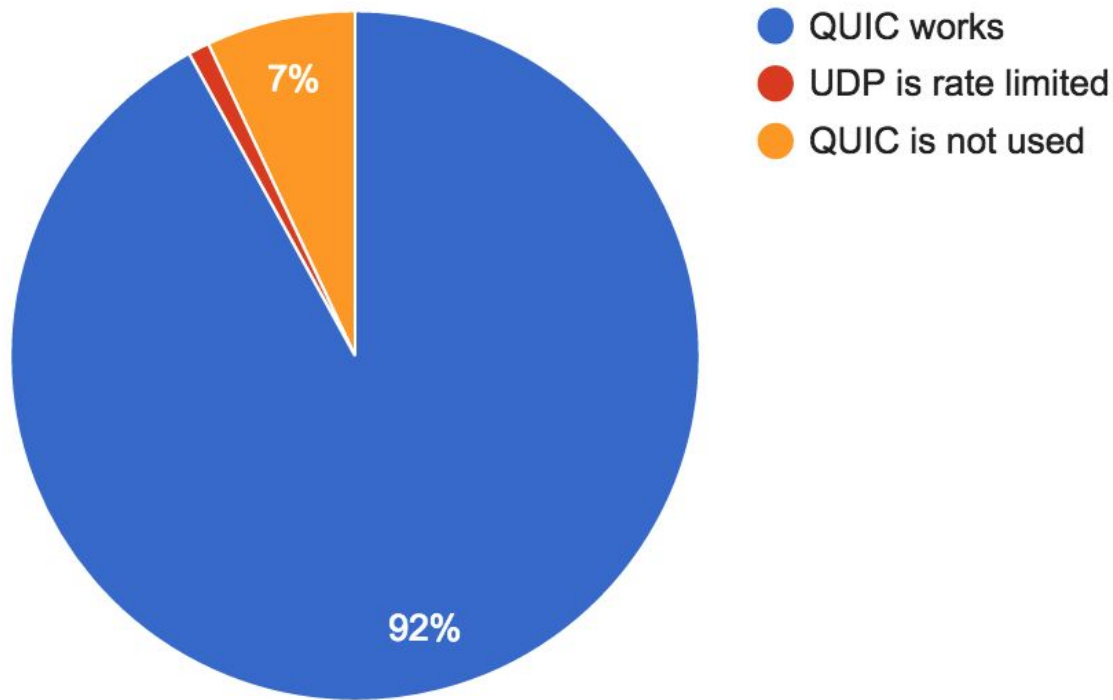
# Deployment timeline

## Tested at scale, with millions of users

- Chrome Canary: June, 2013
- Chrome Stable: April, 2014
- Ramped up for Google traffic in 2015



## QUIC: Does it work?



QUIC handshakes fail when RTTs are greater than 2.5 seconds or when UDP is blocked

# 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)

[Recent Blog Post](#)

# Where are the gains from?

## **0-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



# 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](https://chromium.org))

# Debugging Tools: Chrome

## chrome://net-internals

- Active QUIC sessions
- Captures all events
- Important for filing Chromium bugs

The screenshot shows the Chrome DevTools interface with the 'chrome://net-internals/#events' page open. The 'Events' tab is selected, and a filter is applied: 'type:QUIC\_SESSION is:active'. A table lists 8 of 1327 events. The event at index 3796, for 'www.youtube.com', is selected. The right pane displays the details for this event, showing a QUIC session packet sent to the host 'www.youtube.com'.

ID	Source Type	Description
3767	QUIC_SESSION	i1.ytimg.com
3771	QUIC_SESSION	s.ytimg.com
3773	QUIC_SESSION	csi.gstatic.com
3786	QUIC_SESSION	www.google-analytics.com
3796	QUIC_SESSION	www.youtube.com
3800	QUIC_SESSION	www.gstatic.com
3825	QUIC_SESSION	s2.googleusercontent.com
3884	QUIC_SESSION	pagead2.googleadsyndication.com

**www.youtube.com**  
Start Time: 2013-06-27 11:51:52.832

```
t=1372359112832 [st= 0] +QUIC_SESSION [dt=?]
--> host = "www.youtube.com"
t=1372359112834 [st= 2] QUIC_SESSION_STREAM_FRAME_SENT
--> fin = false
--> length = 512
--> offset = "0"
--> stream_id = 1
t=1372359112834 [st= 2] QUIC_SESSION_PACKET_SENT
--> encryption_level = 0
--> packet_sequence_number = "1"
--> size = 564
t=1372359112835 [st= 3] QUIC_HTTP_STREAM_SEND_REQUEST_HEADERS
--> :host: www.youtube.com
--> :method: GET
--> :path: /user/googlechrome
--> :scheme: http
--> :version: HTTP/1.1
--> accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
--> accept-encoding: gzip,deflate,sdch
--> accept-language: en-US,en;q=0.8
--> cache-control: max-age=0
--> cookie: [280 bytes were stripped]
--> user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2; rv:3.6) AppleWebKit/537.53 (KHTML, like Gecko) Chrome/26.0.1321.109 Safari/537.53
t=1372359112835 [st= 3] QUIC_SESSION_STREAM_FRAME_SENT
--> fin = true
--> length = 568
--> offset = "0"
```

## Debugging Tools: Wireshark

# Parses

- Protocol: QUIC
- CID: Connection ID
- Seq: Sequence number
- Version: ie: Q024
- Public flags: 1 byte
- Payload: Encrypted

No.	Time	Source	Destination	Protocol	Length	Info
985	14.027869000	173.194.46.73	10.1.10.14	QUIC	1392	CID: 3182875774876983667, Seq: 1
986	14.028834000	10.1.10.14	173.194.46.73	QUIC	1392	CID: 3182875774876983667, Seq: 2
989	14.065914000	173.194.46.73	10.1.10.14	QUIC	1392	CID: 3182875774876983667, Seq: 2
990	14.066812000	10.1.10.14	173.194.46.73	QUIC	79	CID: 3182875774876983667, Seq: 3
991	14.194009000	10.1.10.14	173.194.46.73	QUIC	1392	CID: 3182875774876983667, Seq: 4
992	14.194164000	10.1.10.14	173.194.46.73	QUIC	350	CID: 3182875774876983667, Seq: 5
993	14.231536000	173.194.46.73	10.1.10.14	QUIC	85	CID: 3182875774876983667, Seq: 3
994	14.258228000	173.194.46.73	10.1.10.14	QUIC	353	CID: 3182875774876983667, Seq: 4
995	14.268285000	2601:6:2c01:9300:69a8:92607:f8b0:4004:a::12	2601:6:2c01:9300:69a8:92607:f8b0:4004:a::12	QUIC	1412	CID: 2735399198252988334, Seq: 1
997	14.270807000	10.1.10.14	216.58.216.238	QUIC	1392	CID: 2060901289831796684, Seq: 1
998	14.273189000	10.1.10.14	173.194.46.76	QUIC	1392	CID: 16164325528471686122, Seq: 1
999	14.277601000	10.1.10.14	173.194.46.73	QUIC	1392	CID: 9176532438181928584, Seq: 1
1000	14.278560000	10.1.10.14	173.194.46.73	QUIC	1392	CID: 9176532438181928584, Seq: 2
1001	14.278618000	10.1.10.14	173.194.46.73	QUIC	515	CID: 9176532438181928584, Seq: 3
1002	14.284072000	10.1.10.14	173.194.46.73	QUIC	82	CID: 3182875774876983667, Seq: 6
1003	14.295209000	2607:f8b0:4004:a::12	2601:6:2c01:9300:69a8:92607:f8b0:4004:a::12	QUIC	1412	CID: 2735399198252988334, Seq: 1
1004	14.296658000	2601:6:2c01:9300:69a8:92607:f8b0:4004:a::12	2601:6:2c01:9300:69a8:92607:f8b0:4004:a::12	QUIC	99	CID: 2735399198252988334, Seq: 2
1005	14.309132000	216.58.216.238	10.1.10.14	QUIC	1392	CID: 2060901289831796684, Seq: 1
1006	14.312428000	173.194.46.76	10.1.10.14	QUIC	1392	CID: 16164325528471686122, Seq: 1

▶ Frame 981: 1392 bytes on wire (11136 bits), 1392 bytes captured (11136 bits) on interface 0 (outbound)  
 ▶ Ethernet II, Src: Apple\_bc:da:74 (78:31:c1:bc:da:74), Dst: Netgear\_bf:79:04 (c4:04:15:bf:79:04)  
 ▶ Internet Protocol Version 4, Src: 10.1.10.14 (10.1.10.14), Dst: 173.194.46.73 (173.194.46.73)  
 ▶ User Datagram Protocol, Src Port: 51863 (51863), Dst Port: 80 (80)  
 ▼ QUIC (Quick UDP Internet Connections)  
   ▶ Public Flags: 0x0d  
     CID: 3182875774876983667  
     Version: Q024  
     Sequence: 1  
     Payload: 9f8da5bbb0e0724d965b22dc01a001000443484c4f130000...

# What's Next?



# Future Improvements

- Forward Error Correction
- Connection Mobility
- Multipath
- More congestion control experiments

# Open source implementations

## Servers

- Open source test server included in Chromium
- Working with other server vendors

## 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	BarBoF <b>NOW!</b>
Ongoing	Subsuming QUIC's 0-RTT handshake in TLS1.3

# Review: QUIC Summary

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



# QUIC

**Source:** QUIC in Chromium

**Page:** www.chromium.org/quic

**Public Mailing list:** proto-quic@chromium.org

**IETF draft:** draft-tsvwg-quic-protocol-01