SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

A QUIC Introduction

Lars Eggert
NetApp

Talk outline

1) Internet Transport

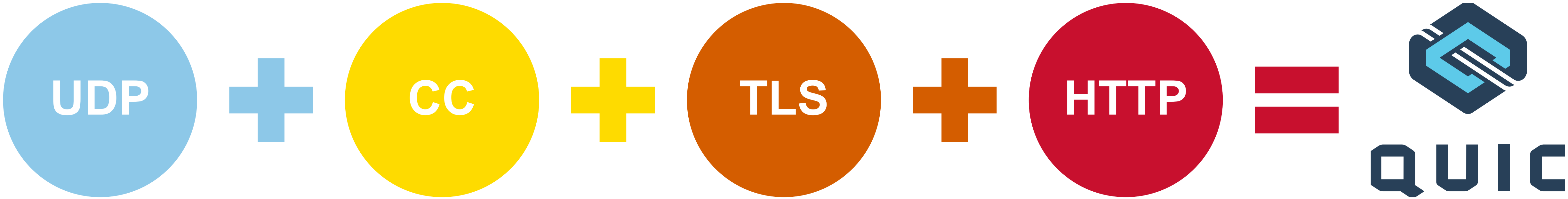
2) Current Challenges

3) QUIC

4) Status & discussion

QUIC: a fast, secure, evolvable transport protocol for the Internet

- **Fast** **better user experience** than TCP/TLS for HTTP/2 and other content
- **Secure** **always-encrypted** end-to-end security, resist pervasive monitoring
- **Evolvable** prevent network from ossifying, deploy new QUIC versions quickly
- **Transport** support all TCP content & more (realtime media, etc.)
  provide better abstractions, avoid known TCP issues



tl;dr

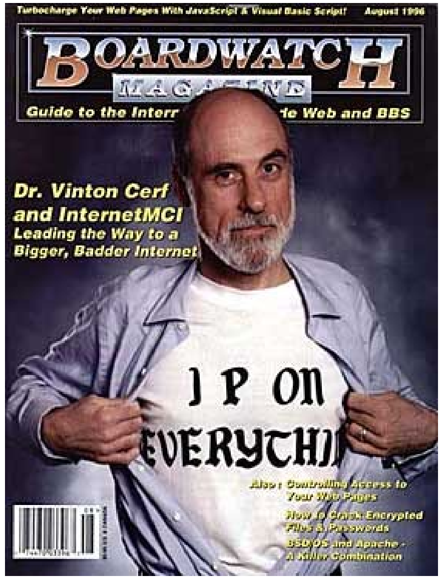
- **The web will move to QUIC first**, and then everything else will
  - This year!
- If you do anything with HTTP, TCP or just networks,
  QUIC should be on your radar now

Internet transport

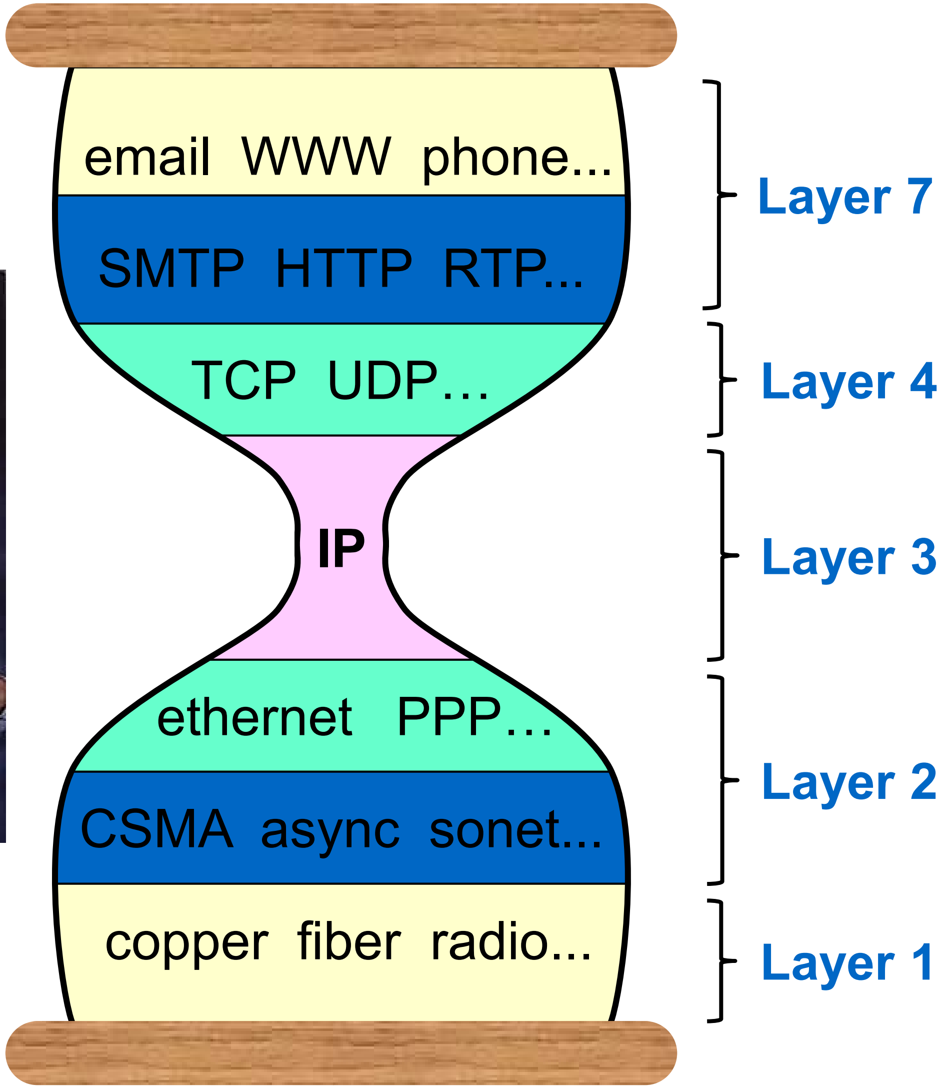
The Internet hourglass

Classical version

- Inspired by OSI “seven-layer” model
  - Minus presentation (6) and session (5)
- “IP on everything”
  - All link tech looks the same (approx.)
- **Transport layer** provides communication abstractions to apps
  - Unicast/multicast
  - Multiplexing
  - Streams/messages
  - Reliability (full/partial)
  - Flow/congestion control
  - …



Boardwatch Magazine, Aug. 1994.

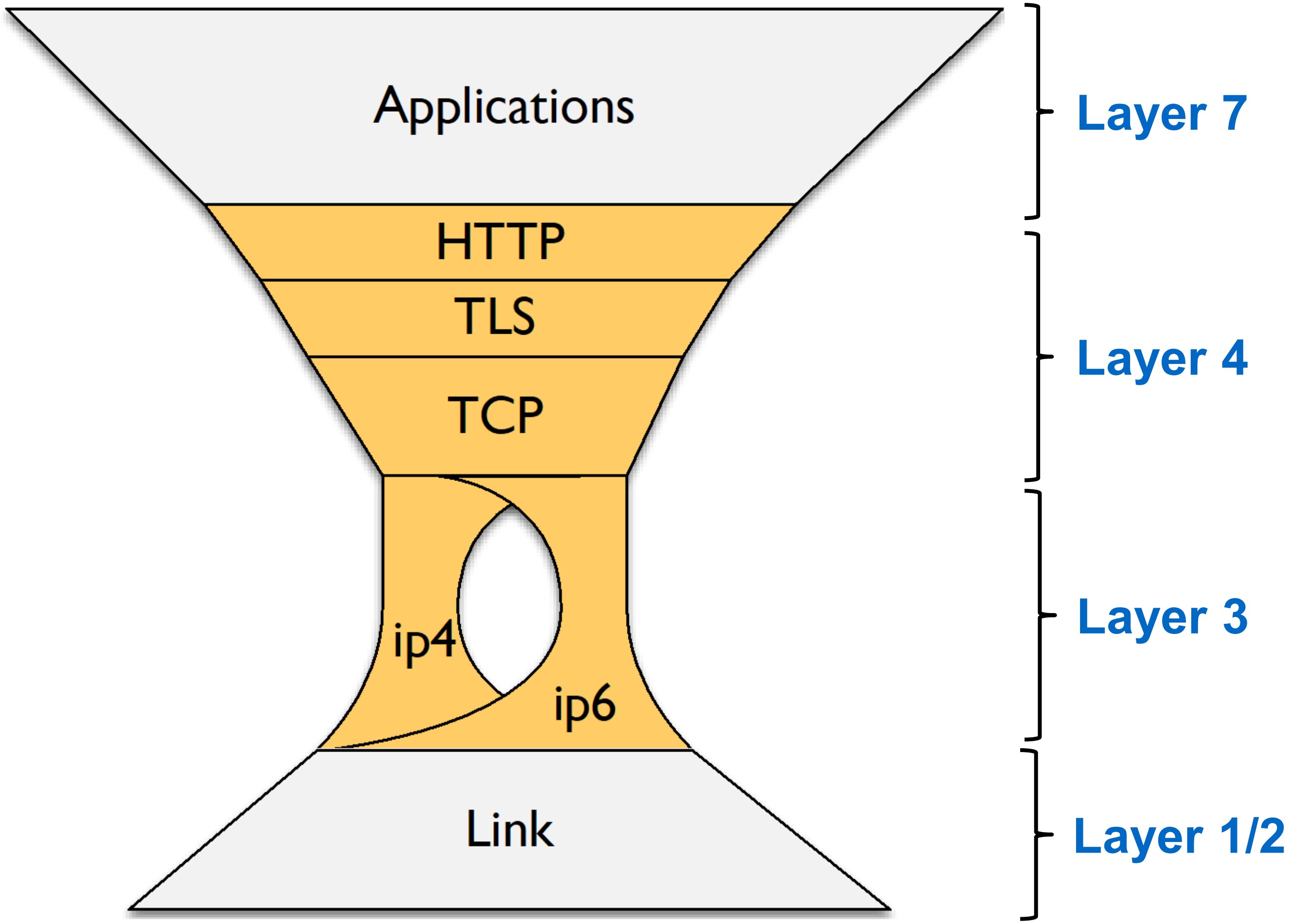


Steve Deering. Watching the Waist of the Protocol Hourglass. Keynote, IEEE ICNP 1998, Austin, TX, USA. http://www.ieee-icnp.org/1998/Keynote.ppt

The Internet hourglass

2015 version (ca.)

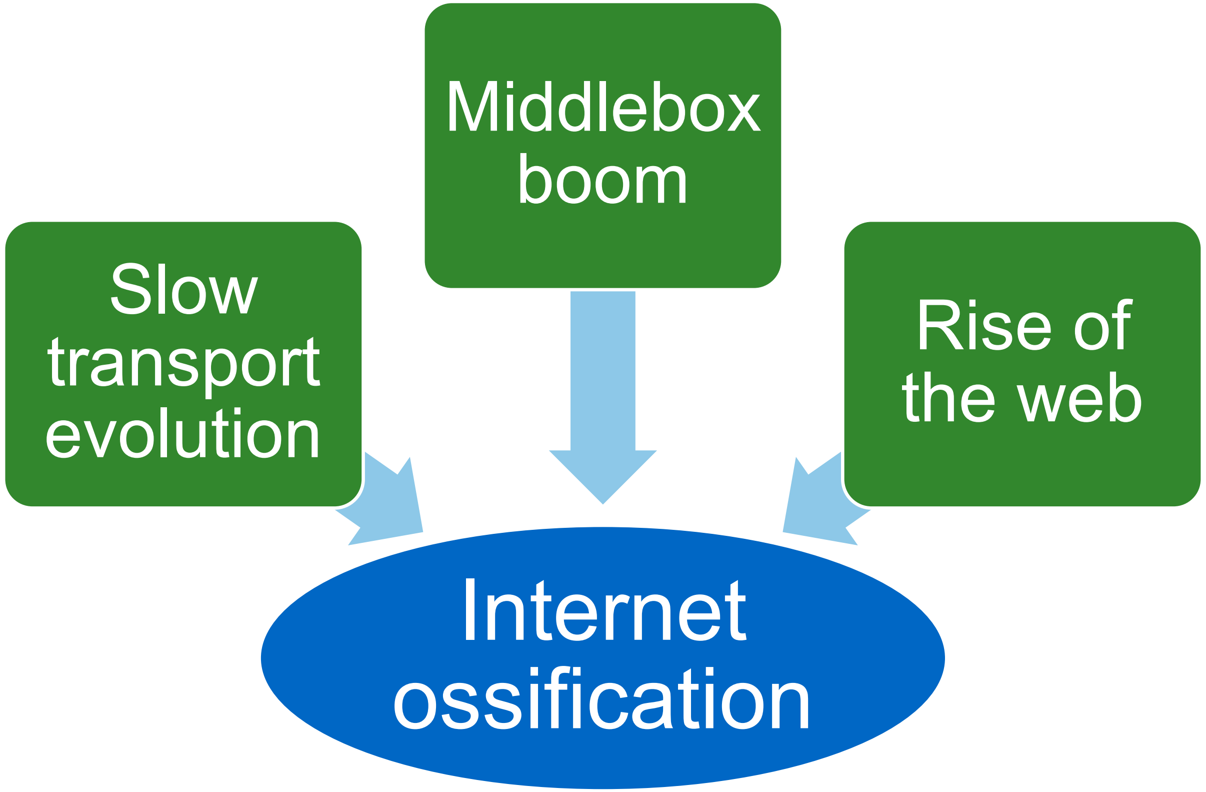
- The waist has split: **IPv4** and **IPv6**
- **TCP** is drowning out UDP
- **HTTP** and **TLS** are *de facto* part of transport
- Consequence: **web apps** on IPv4/6



B. Trammell and J. Hildebrand, "Evolving Transport in the Internet," in *IEEE Internet Computing*, vol. 18, no. 5, pp. 60-64, Sept.-Oct. 2014.

What happened?

- **Transport slow to evolve** (esp. TCP)
  - Fundamentally difficult problem
- **Network made assumptions** about what (TCP) traffic looked like & how it behaved
- Tried to “help” and “manage”
  - TCP “accelerators” & firewalls, DPI, NAT, etc.
- **The web happened**
  - Almost all content on HTTP(S)
  - Easier/cheaper to develop for & deploy on
  - Amplified by mobile & cloud
  - Baked-in client/server assumption



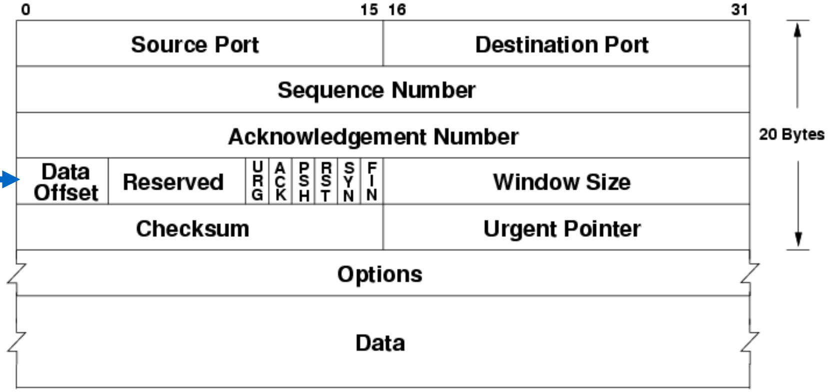
Example ossifications

IP	•Send from/to anywhere anytime	vs. enforced directionality & timeliness
IP	•Many protocols on top of IP	vs. packets dropped unless TCP or UDP
IP	•End-to-end addressing	vs. network assumes it can rewrite addresses/ports
IP	•Use IP options to signal	vs. options not used (dropped) on WAN
*	•Bits have meaning only inside a layer	vs. network can (should!) touch bits across a packet
TCP	•Network is stateless	vs. network assumes it can track entire connection
TCP	•Data has meaning to app only	vs. network can rewrite or insert

TCP challenges

TCP is not aging well

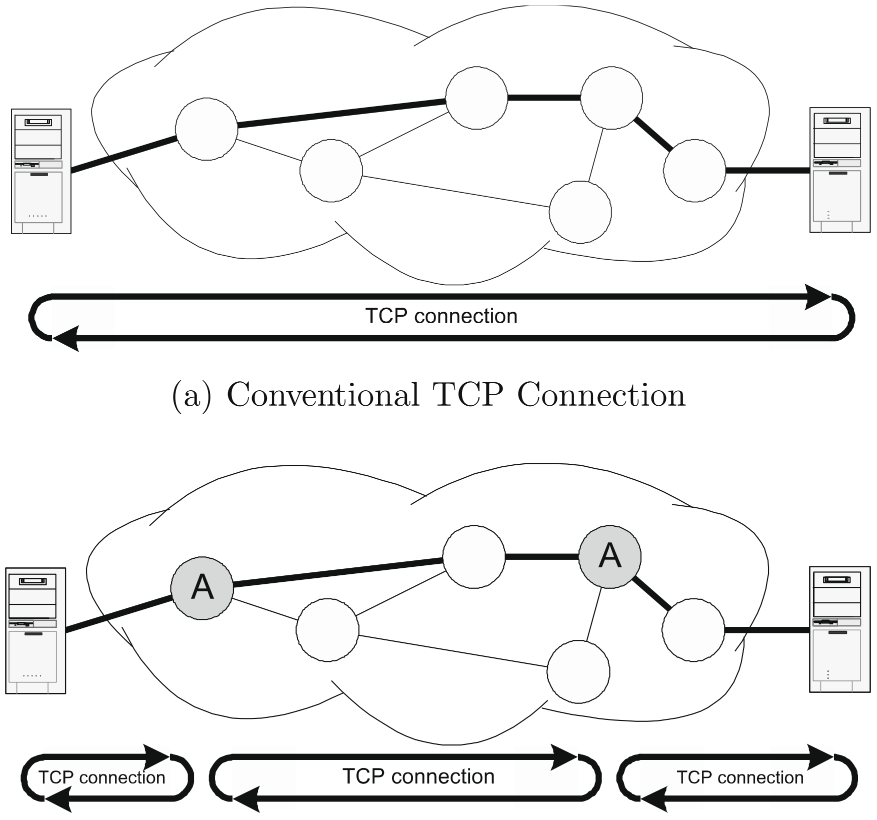
- **We’re hitting hard limits** (e.g., TCP option space)
  - 40B total (15 * 4B - 20)
  - Used: SACK-OK (2), timestamp (10), window Scale (3), MSS (4)
  - Multipath needs 12, Fast-Open 6-18…
- **Incredibly difficult to evolve**, c.f. Multipath TCP
  - New TCP must look like old TCP, otherwise it gets dropped
  - TCP is already very complicated
- **Slow upgrade cycles** for new TCP stacks (kernel update required)
  - Better with more frequent update cycles on consumer OS
  - Still high-risk and invasive (reboot)
- **TCP headers not encrypted** or even authenticated – middleboxes can still meddle
  - TCP-MD5 and TCP-AO in practice only used for (some) BGP sessions



By Ere at Norwegian Wikipedia (Own work) [Public domain], via Wikimedia Commons

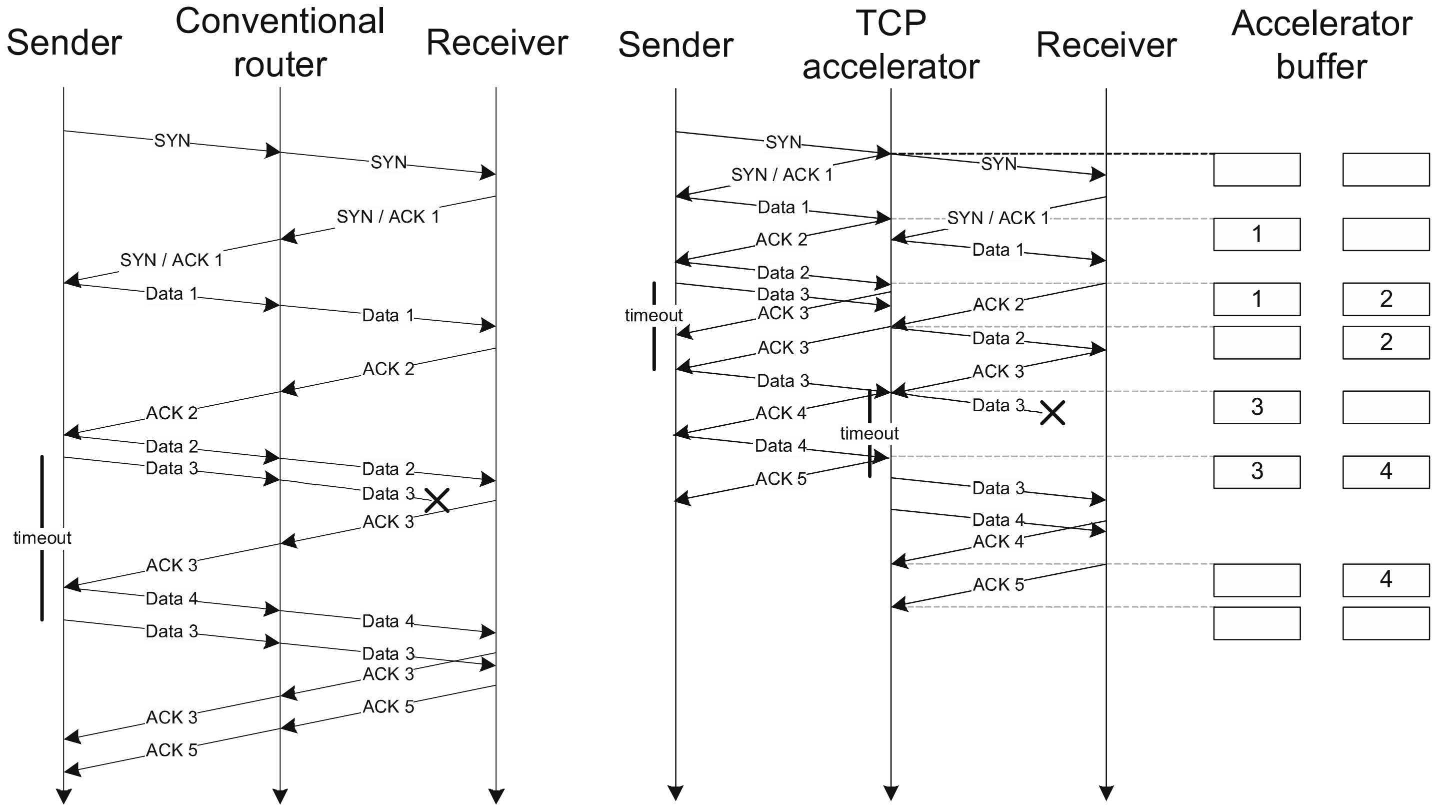
Middleboxes meddle

Example: TCP accelerators



(a) Conventional TCP Connection

(b) Accelerated TCP Connection



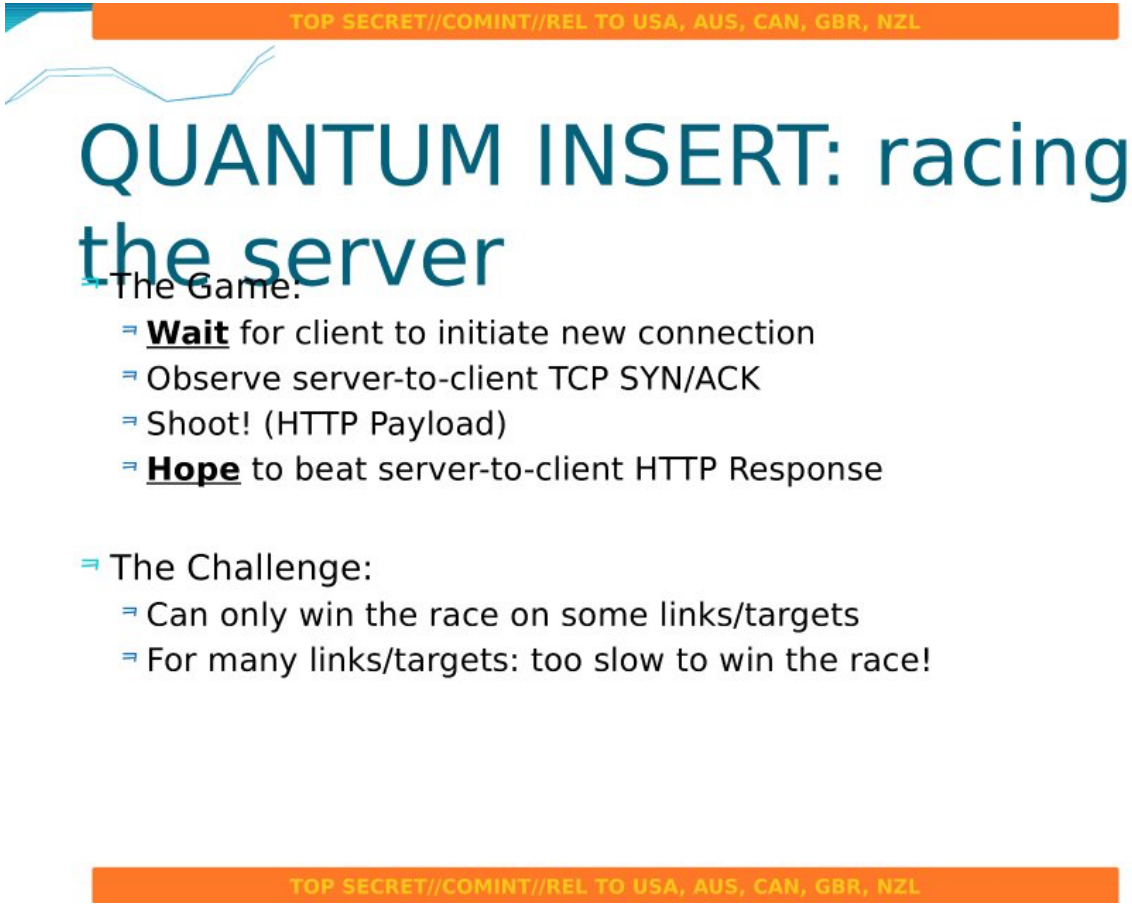
(a) Conventional TCP Connection

(b) Accelerated TCP Connection

Sameer Ladiwala, Ramaswamy Ramaswamy, and Tilman Wolf. Transparent TCP acceleration. Computer Communications, Volume 32, Issue 4, 2009, pages 691-702.

Middleboxes meddle

Example: Nation states attacking end users or services



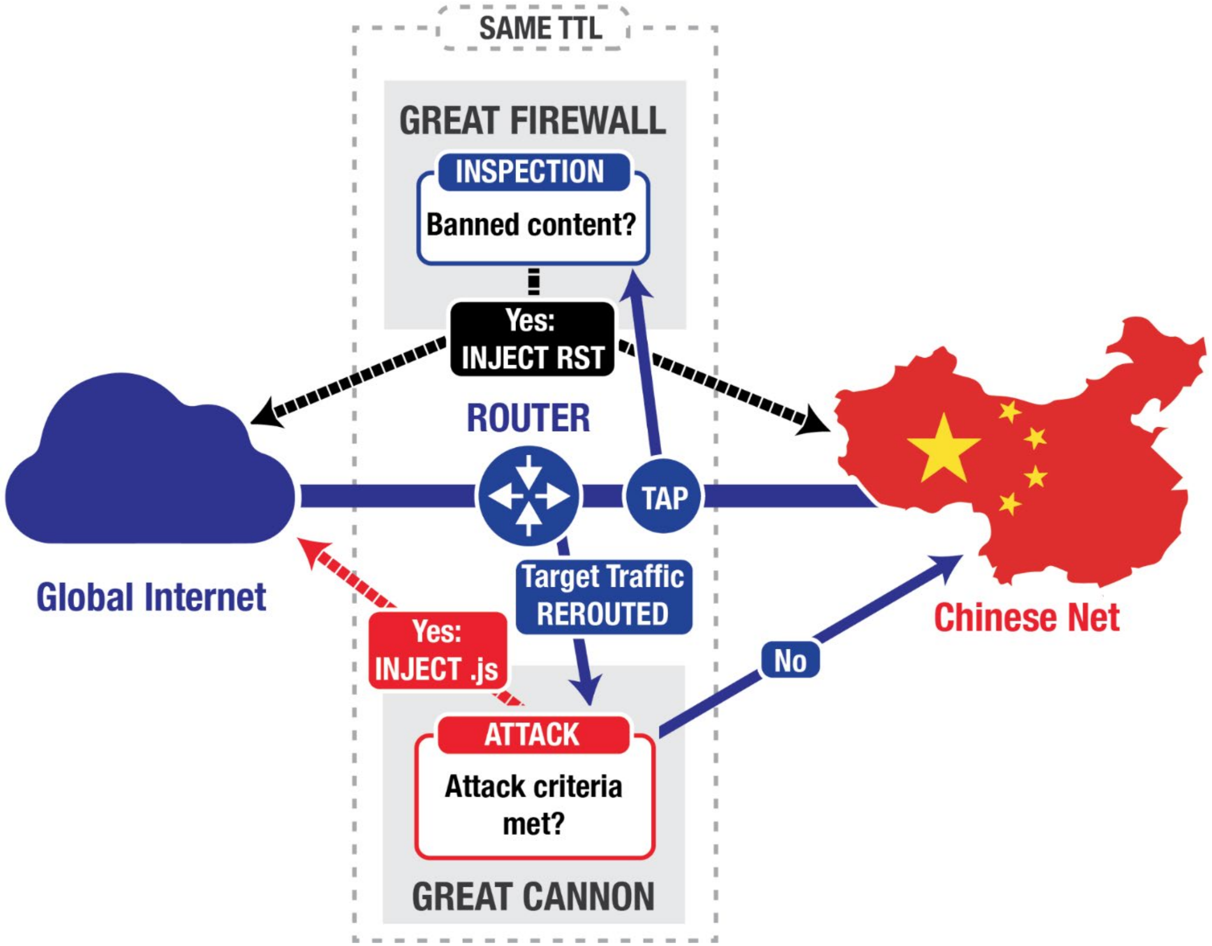
TOP SECRET//COMINT//REL TO USA, AUS, CAN, GBR, NZL

QUANTUM INSERT: racing the server

- The Game:
  - **Wait** for client to initiate new connection
  - Observe server-to-client TCP SYN/ACK
  - Shoot! (HTTP Payload)
  - **Hope** to beat server-to-client HTTP Response
- The Challenge:
  - Can only win the race on some links/targets
  - For many links/targets: too slow to win the race!

TOP SECRET//COMINT//REL TO USA, AUS, CAN, GBR, NZL

QFIRE Pilot Lead. NSA/Technology Directorate. QFIRE pilot report. 2011.



B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, and V. Paxson. An Analysis of China’s “Great Cannon”. 5th USENIX FOCI Workshop, 2015.

Pervasive monitoring is an attack

RFC 7528

- IETF (& wider) community consensus that pervasive monitoring is an attack
- Agreement to mitigate pervasive monitoring
- What does “mitigate” mean?
- To many, ”encrypt as much as possible”

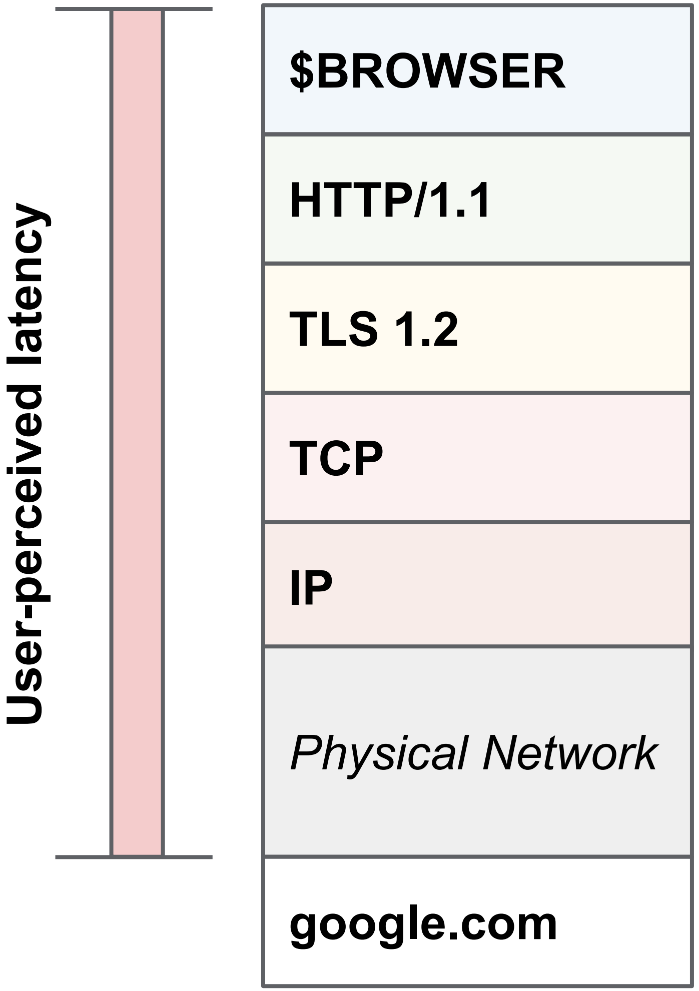
Laura Poitras / Praxis Films. CC BY 3.0

QUIC

Introduction

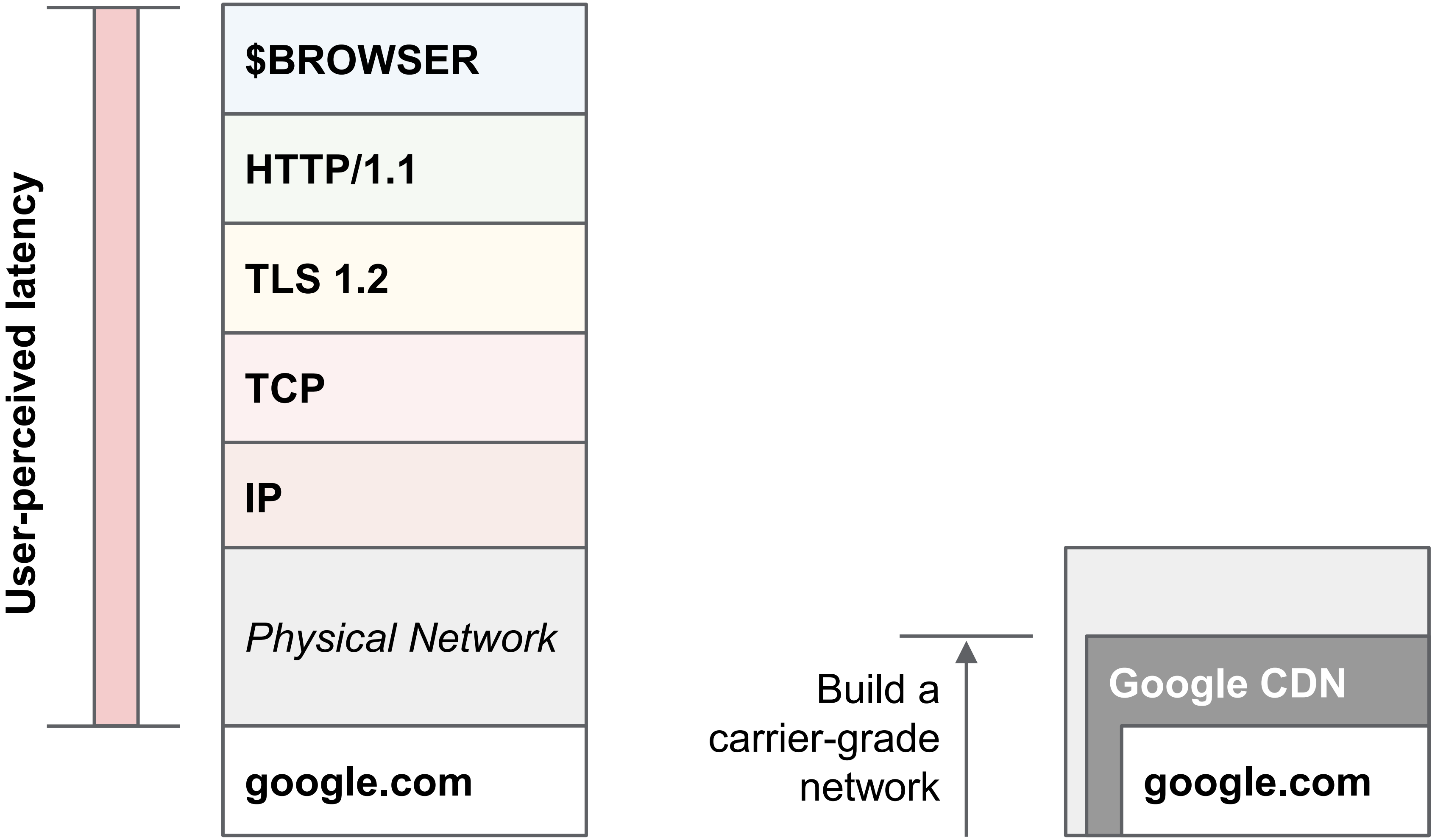
How do you make the web faster?

QUIC - Redefining Internet Transport. J. Iyengar. IETF-93 QUIC BoF presentation, 2015.



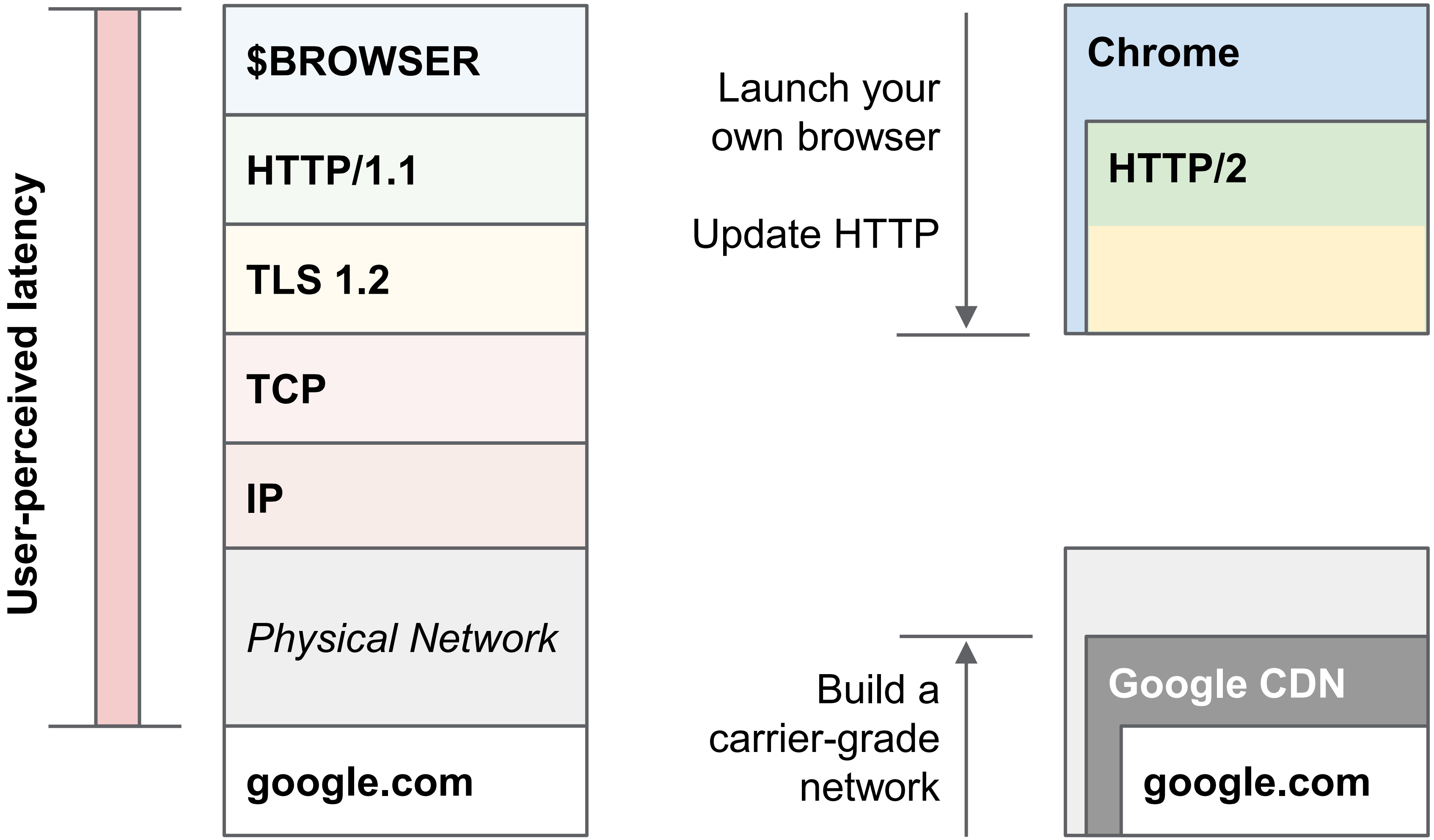
How do you make the web faster?

QUIC - Redefining Internet Transport. J. Iyengar. IETF-93 QUIC BoF presentation, 2015.



How do you make the web faster?

QUIC - Redefining Internet Transport. J. Iyengar. IETF-93 QUIC BoF presentation, 2015.



User-perceived latency

$BROWSER

HTTP/1.1

TLS 1.2

TCP

IP

Physical Network

google.com

Launch your own browser

Update HTTP

Chrome

HTTP/2

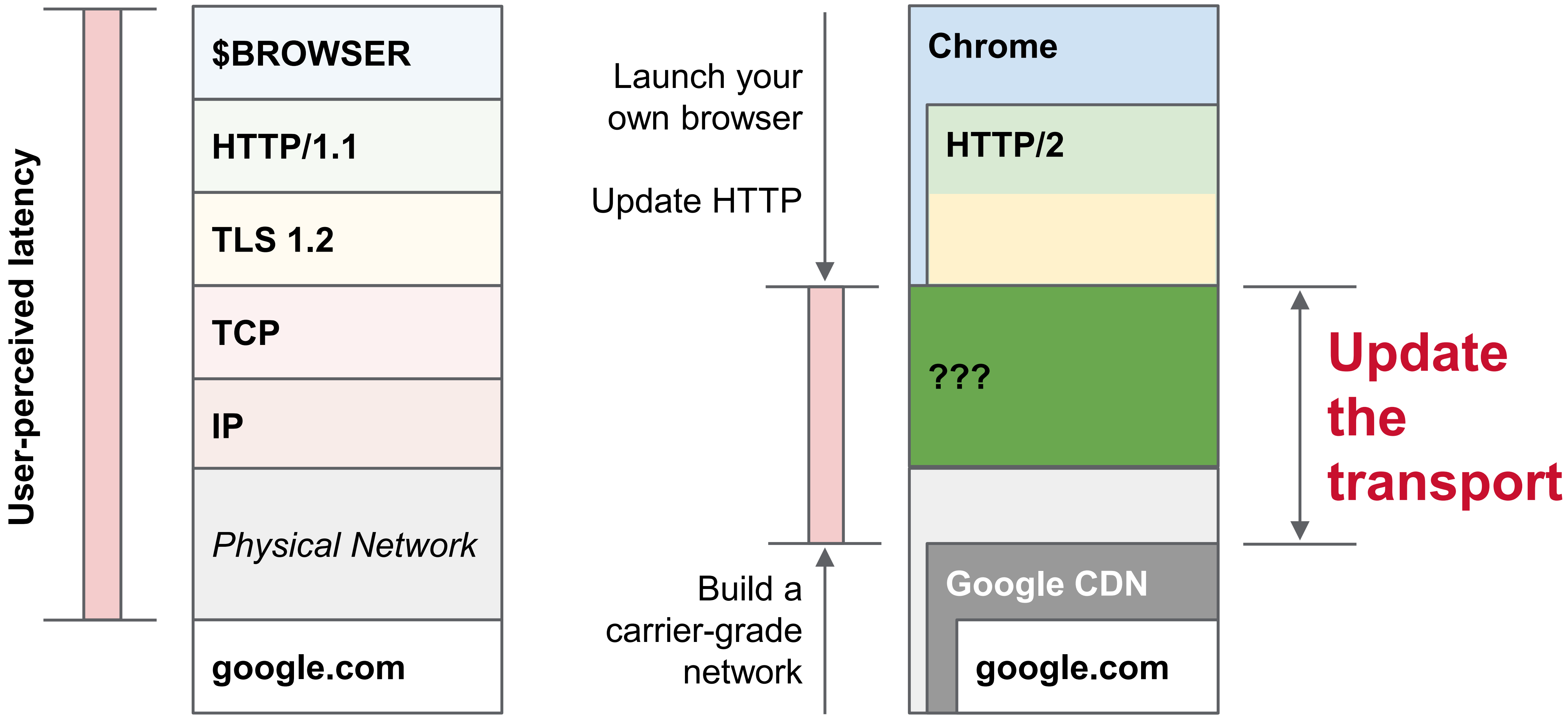
Build a carrier-grade network

Google CDN

google.com

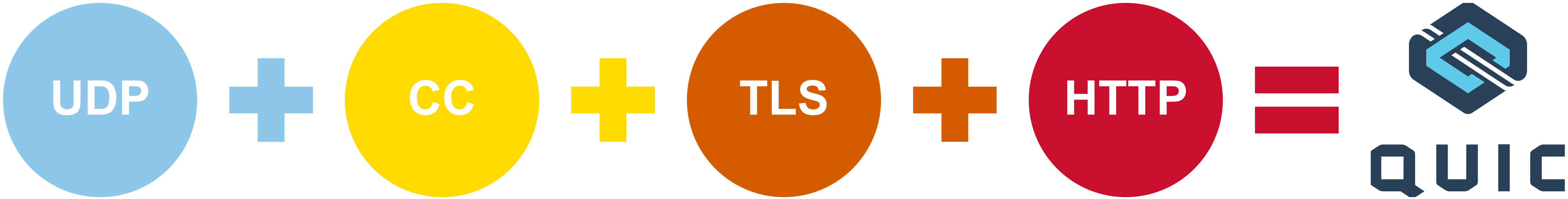
How do you make the web faster?

QUIC - Redefining Internet Transport. J. Iyengar. IETF-93 QUIC BoF presentation, 2015.



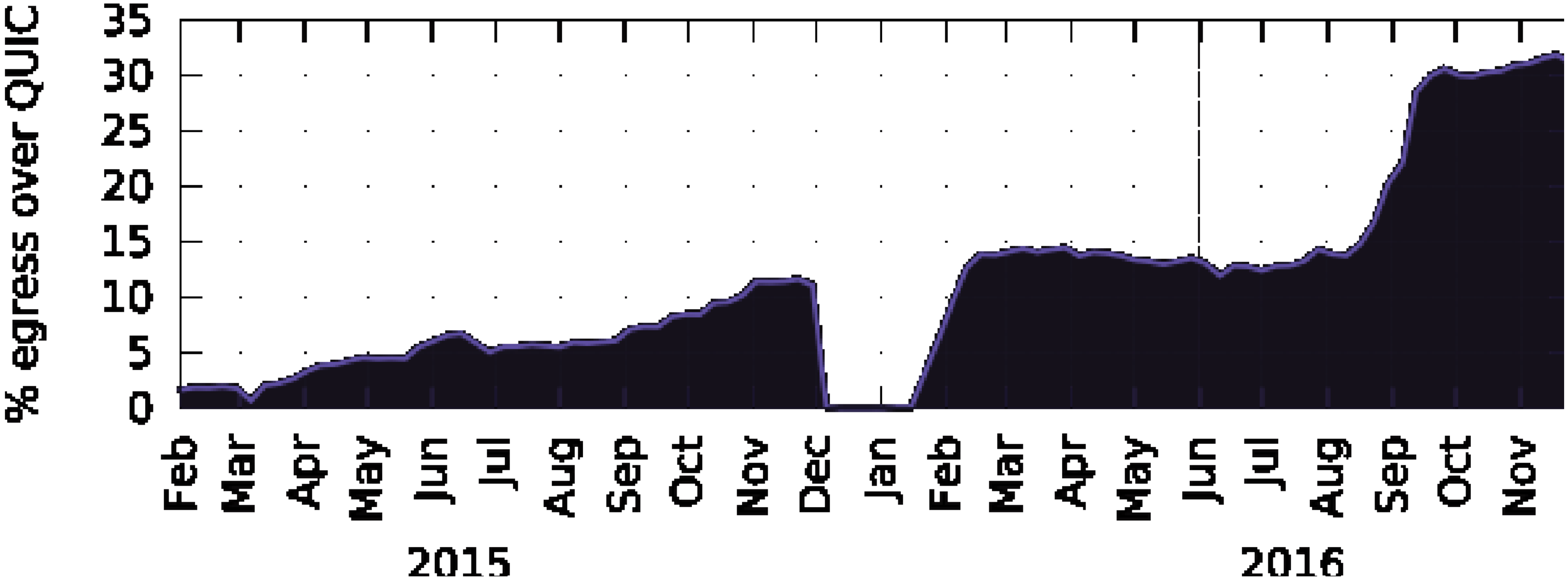
QUIC: a fast, secure, evolvable transport protocol for the Internet

- **Fast** **better user experience** than TCP/TLS for HTTP/2 and other content
- **Secure** **always-encrypted** end-to-end security, resist pervasive monitoring
- **Evolvable** prevent network from ossifying, deploy new QUIC versions quickly
- **Transport** support all TCP content & more (realtime media, etc.)
  provide better abstractions, avoid known TCP issues



QUIC is not *that* new, actually

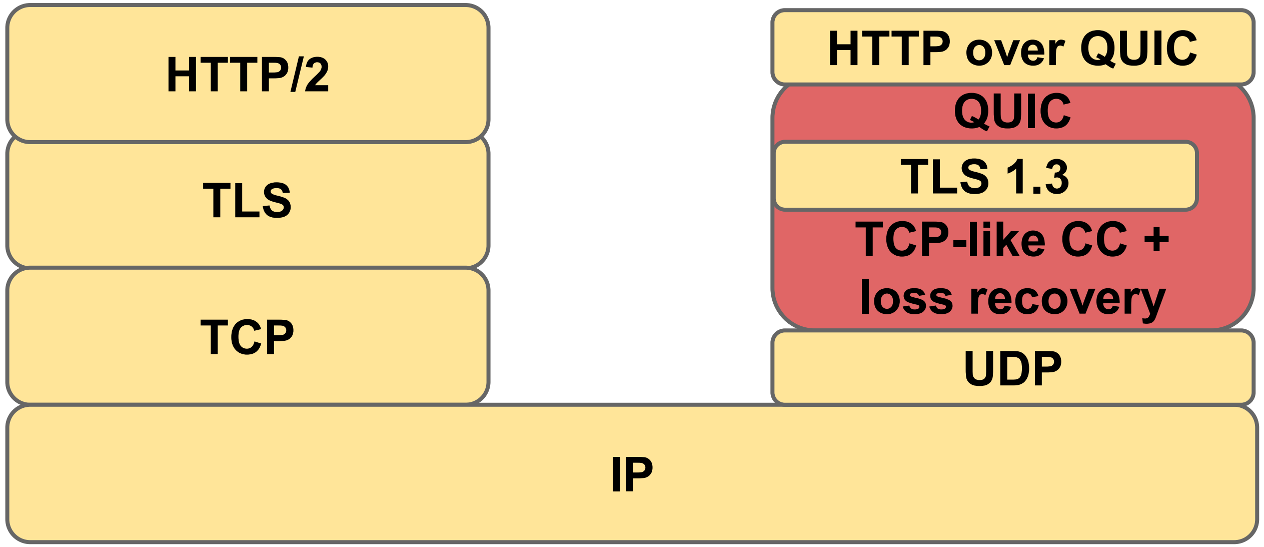
- Originates with Google, deployed between Google services and Chrome since 2014
- As of mid-2017, makes up 35% of Google egress traffic (**~7% of total Internet traffic**)



A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang, and Z. Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment.. ACM SIGCOMM, 2017.

QUIC in the stack

- Integrated transport stack on top of UDP
- Replaces TCP and some part of HTTP; reuses TLS-1.3
- Initial target application: HTTP/2
- Prediction: many others will follow



J. Iyengar. QUIC Tutorial A New Internet Transport/ IETF-98 Tutorial, 2017.

UDP

Why UDP?

- TCP hard to evolve
- Other protocols blocked by middleboxes (SCTP, etc.)
- **UDP is all we have left**
- Not without problems!
  - Many middleboxes ossified on “UDP is for DNS”
  - Enforce short binding timeouts, etc.
  - Short-term issue with hardware NIC offloading
- Also, benefits
  - Can deploy in userspace (no kernel update needed)
  - Can offer alternative transport types (partial reliability, etc.)

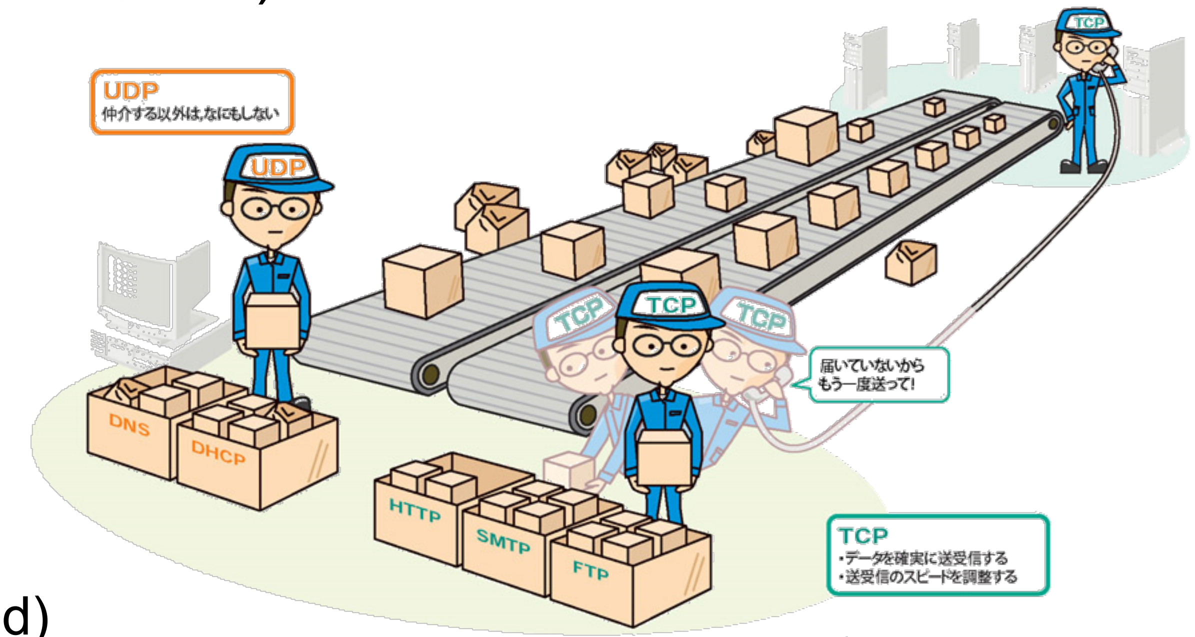

UDP
仲介する以外は,なにもしない
UDP
TCP
DNS
DHCP
TCP
TCP
TCP
届いていないから
もう一度送って!
HTTP
SMTP
FTP
TCP
・データを確実に送受信する
・送受信のスピードを調整する

Image from http://itpro.nikkeibp.co.jp

Why congestion control?

CC

- Functional CC is **absolute requirement** for operation over real networks
  - UDP has no CC
- First approach: **take what works for TCP, apply to QUIC**
- Consequence: need
  - Segment/packet numbers
  - Acknowledgments (ACKs)
  - Round-trip time (RTT) estimators
  - etc.
- Not an area of large innovation at present
  - This will change

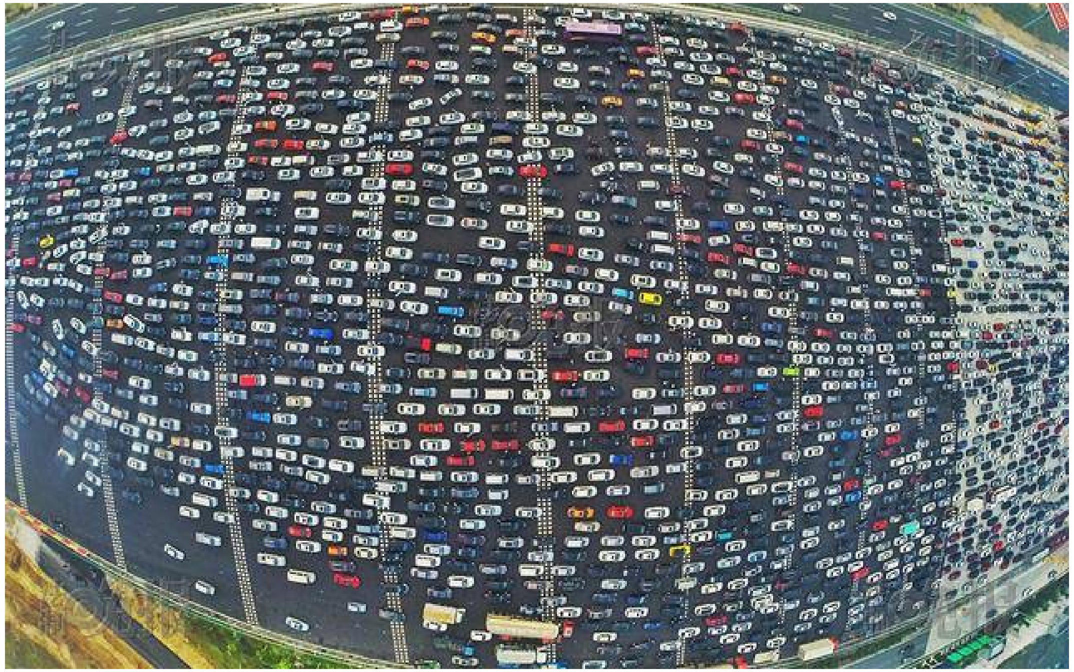
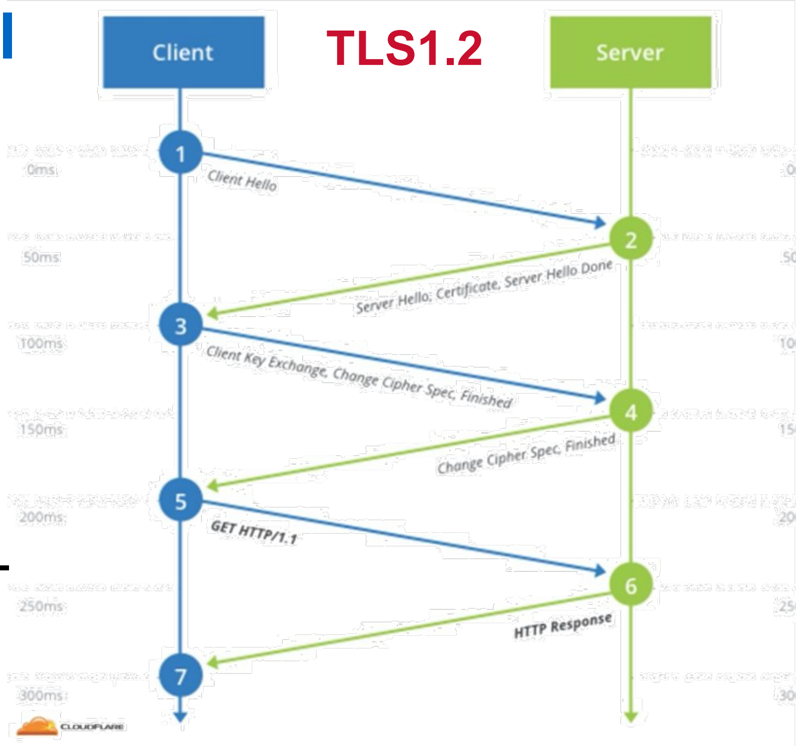


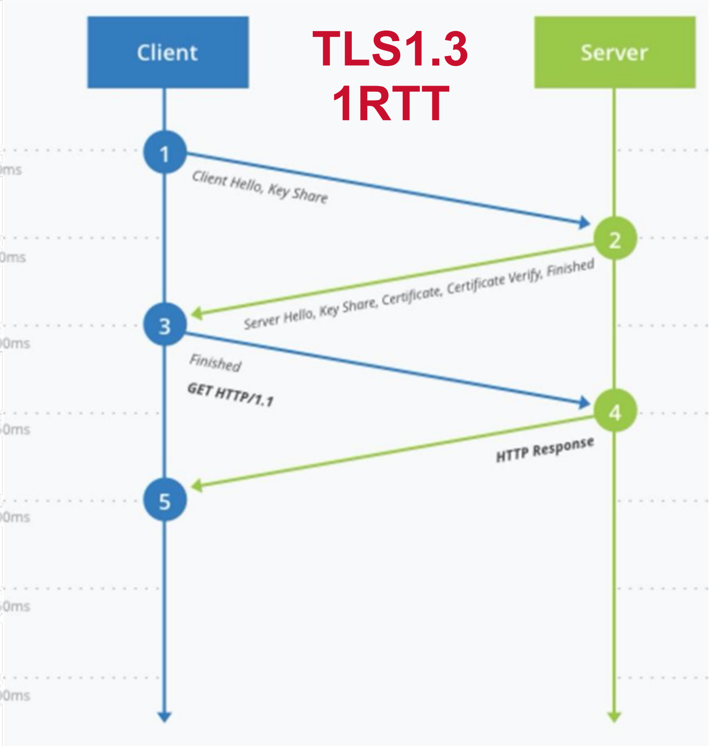
Image from People's Daily, http://people.cn/

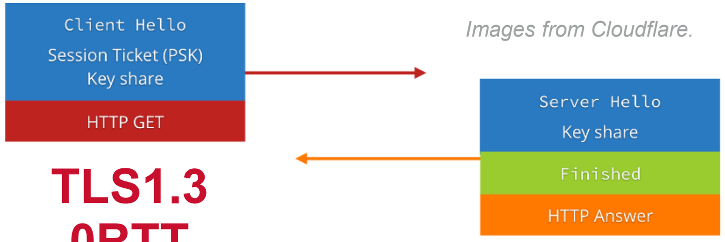
Why transport-layer security (TLS)?

TLS

- **End-to-end security is critical**
  - To protect users
  - To prevent network ossification
- TLS is very widely used
  - Can leverage all community R&D
  - Can leverage the PKI
- **Don’t want custom security** – too much to get wrong
  - Even TLS keeps having issues
  - But TLS 1.3 removes a lot of cruft
- And benefit from new TLS features
  - E.g., 0-RTT handshakes (inspired by gQUIC-crypto)


TLS1.2
Client
Server
0ms
50ms
100ms
150ms
200ms
250ms
300ms
1
Client Hello
2
Server Hello, Certificate, Server Hello Done
3
Client Key Exchange, Change Cipher Spec, Finished
4
Change Cipher Spec, Finished
5
GET HTTP/1.1
6
HTTP Response
7
CLOUDFLARE


TLS1.3
1RTT



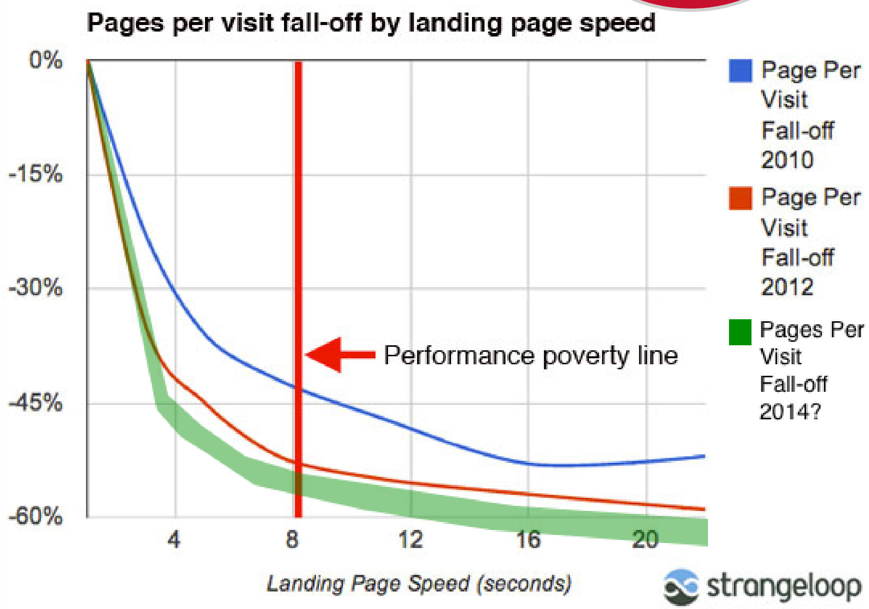
TLS1.3
0RTT

Images from Cloudflare.

Why HTTP?

HTTP

- Because that’s where the **impact** is
  - Web industry incredibly interested in improved UE and security
- Rapid update cycles for browsers, servers, CDNs, etc.
  - Can deploy and update QUIC quickly
- Many other app protocols will follow

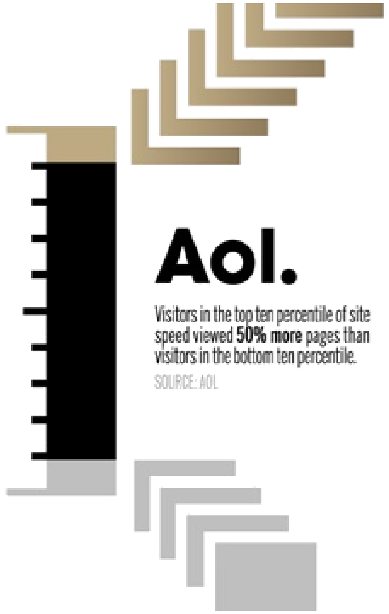


Sped up average page load time from 6 seconds to 1.2 seconds. Results: Increased revenue by 12% and page views by 25%.

SOURCE: Shopzilla

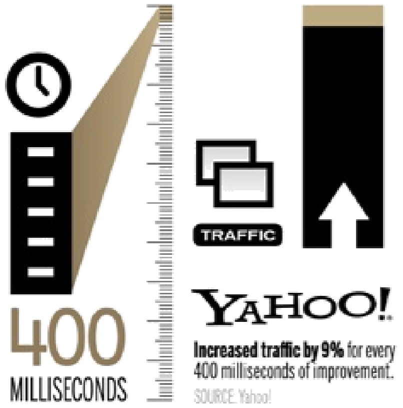
Increased revenue by 1% for every 100 milliseconds of improvement.

SOURCE: Amazon



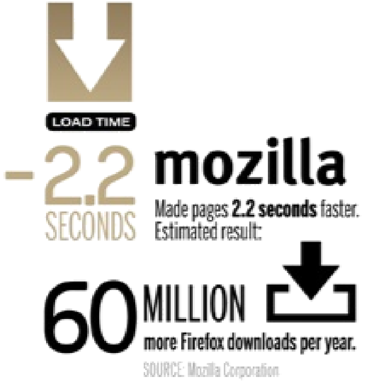
Visitors in the top ten percentile of site speed viewed **50% more** pages than visitors in the bottom ten percentile.

SOURCE: AOL



Increased traffic by 9% for every 400 milliseconds of improvement.

SOURCE: Yahoo!



Made pages **2.2 seconds** faster. Estimated result:

60 MILLION more Firefox downloads per year.

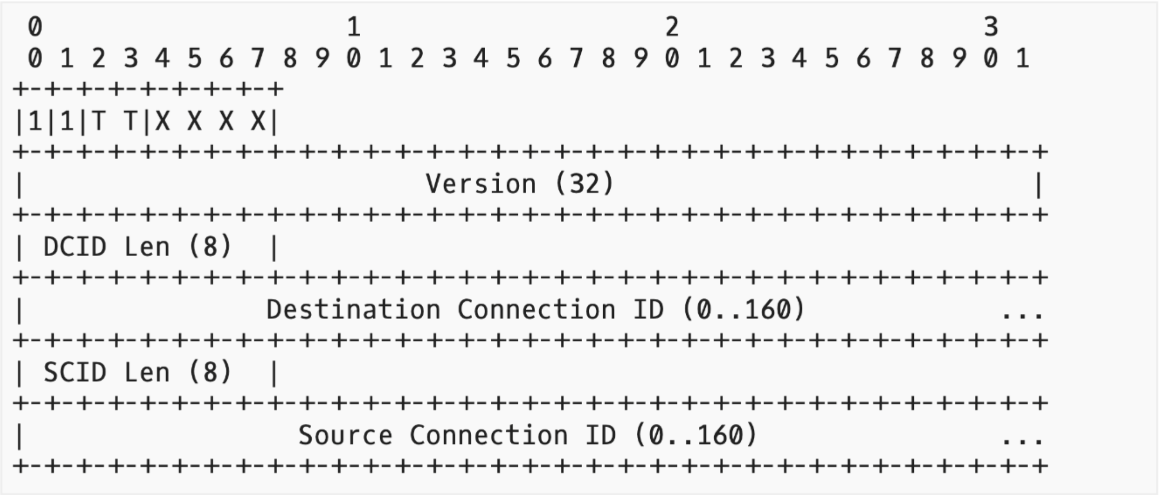
SOURCE: Mozilla Corporation

QUIC

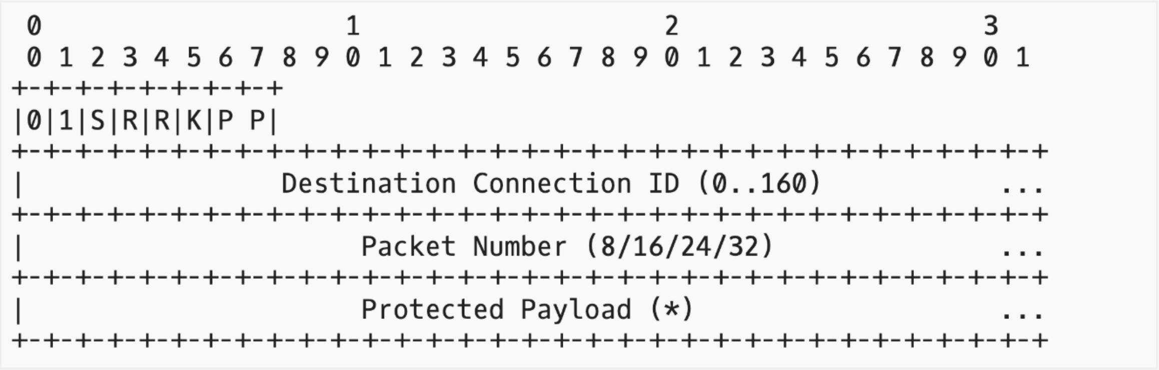
Selected aspects

Minimal network-visible header

- With QUIC, the network sees:
  - Packet **type** (partially obfuscated)
  - QUIC **version** (only in long packet header)
  - Destination **CID**
  - Packet **number** (obfuscated)
- With TCP, also
  - ACK numbers, ECN information
  - Timestamps
  - Windows & scale factors
- Also, entire QUIC header is **authenticated**, i.e., not modifiable



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|1|1|T T|X X X X|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Version (32)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| DCID Len (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (0..160)            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SCID Len (8)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Connection ID (0..160)               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

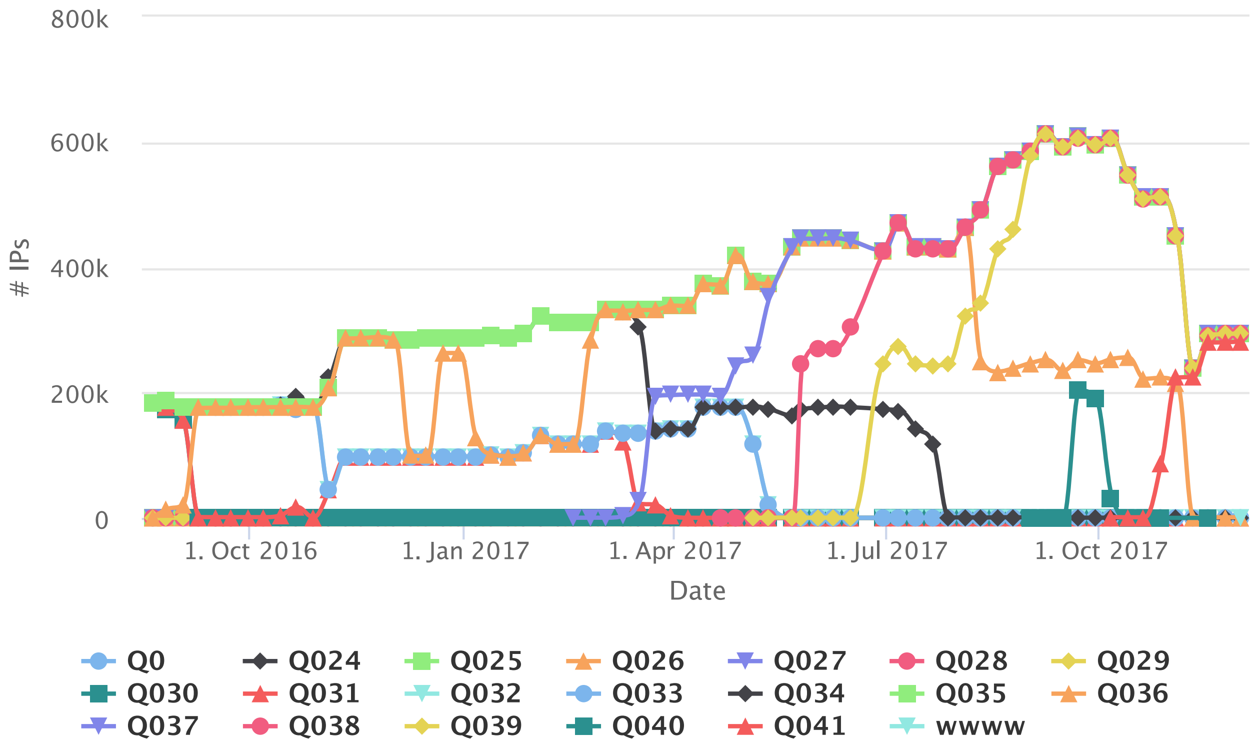


```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|0|1|S|R|R|K|P P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Destination Connection ID (0..160)           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Packet Number (8/16/24/32)              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Protected Payload (*)                   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version negotiation

(Currently under re-design)

- 32-bit version field
  - IP: 8 bits, TCP: 0 bits
- Allows rapid deployment of new versions
  - Plus, vendor-proprietary versions
- Very few **protocol invariants**
  - Location and lengths of version and CIDs in LH
  - Location and lengths of CID in SH (if present)
  - Version negotiation server response
  - Etc. (details under discussion)
- Everything else is version-dependent
  - But must **grease** unused codepoints!



Source: RWTH QUIC Measurements: https://quic.comsys.rwth-aachen.de/

1-RTT vs. 0-RTT handshakes

- **QUIC client can send 0-RTT data in first packets**
  - Using new TLS 1.3 feature
- Except for very first contact between client and server
  - Requires 1-RTT handshake (same latency as TCP w/o TLS)
- **Huge latency win in many cases** (faster than TCP)
  - HTTPS: 7 messages
  - QUIC 1-RTT or TCP: 5 messages
  - QUIC 0-RTT: 2 messages
- Also helps with
  - Tolerating NAT re-bindings
  - Connection migration to different physical interface
- But only for **idempotent** data

Everything else is frames

- Inside the crypto payload,
  QUIC carries a sequence of frames
  - Encrypted = can change between versions
- Frames can come in **any order**
- Frames carry **control data** and **payload data**
- Payload data is carried in **STREAM** frames
  - Most other frames carry control data
- Packet acknowledgment blocks in **ACK** frames

- PADDING
- PING
- **ACK**
- RESET_STREAM
- STOP_SENDING
- CRYPTO
- NEW_TOKEN
- **STREAM**
- MAX_DATA
- MAX_STREAM_DATA
- MAX_STREAMS
- DATA_BLOCKED
- STREAM_DATA_BLOCKED
- STREAMS_BLOCKED
- NEW_CONNECTION_ID
- RETIRE_CONNECTION_ID
- PATH_CHALLENGE
- PATH_RESPONSE
- CONNECTION_CLOSE
- HANDSHAKE_DONE

Stream multiplexing

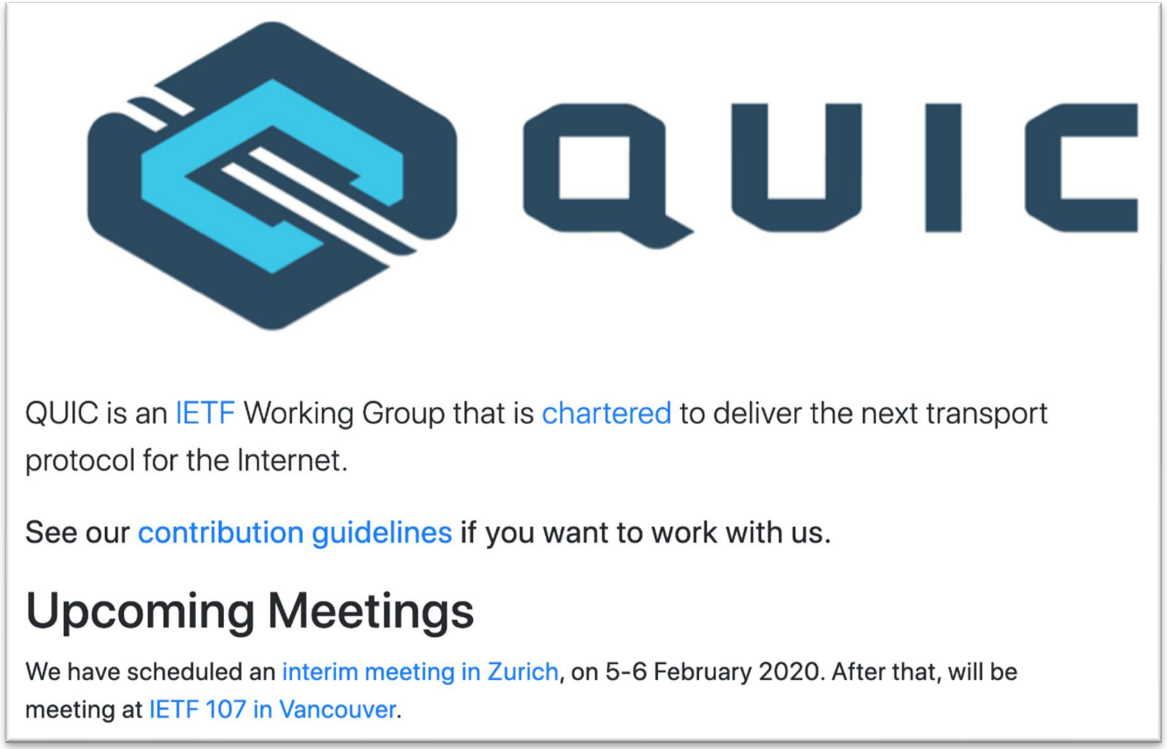
- A QUIC **connection** multiplexes potentially many **streams**
  - Congestion control happens at the connection level
  - Connections are also flow controlled
- **Streams**
  - Carry units of application data
  - Can be uni- or bidirectional
  - Can be opened by client or server
  - Are flow controlled
  - Currently, always reliably transmitted (partial reliability coming soon)
- Number of open streams is negotiated over time (as are stream windows)
- Stream prioritization is up to application

Current status & discussions

QUIC and the IETF

- **QUIC is being standardized in the IETF**
  - QUIC is already very different from Google QUIC
- Est. delivery date: Sep 2020
- 20+ known implementation efforts:

CLOUDFLARE
TUM
facebook
NGINX
Google
Microsoft
traffic server™
NetApp
moz://a
fastly
f5
LITESPEED



QUIC is an IETF Working Group that is chartered to deliver the next transport protocol for the Internet.

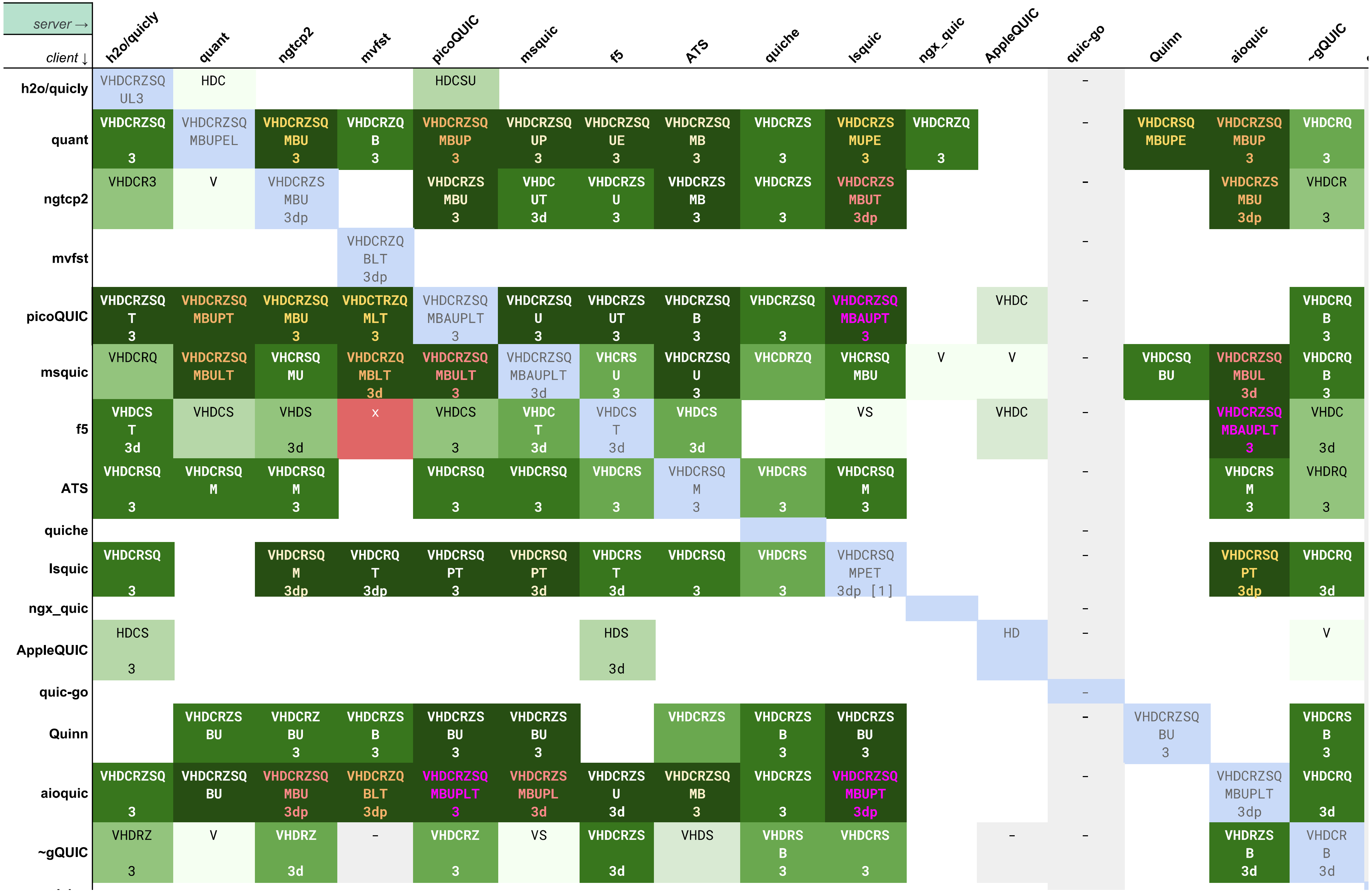
See our contribution guidelines if you want to work with us.

Upcoming Meetings

We have scheduled an interim meeting in Zurich, on 5-6 February 2020. After that, will be meeting at IETF 107 in Vancouver.

- **https://quicwg.github.io/**
- **https://quicdev.slack.com**

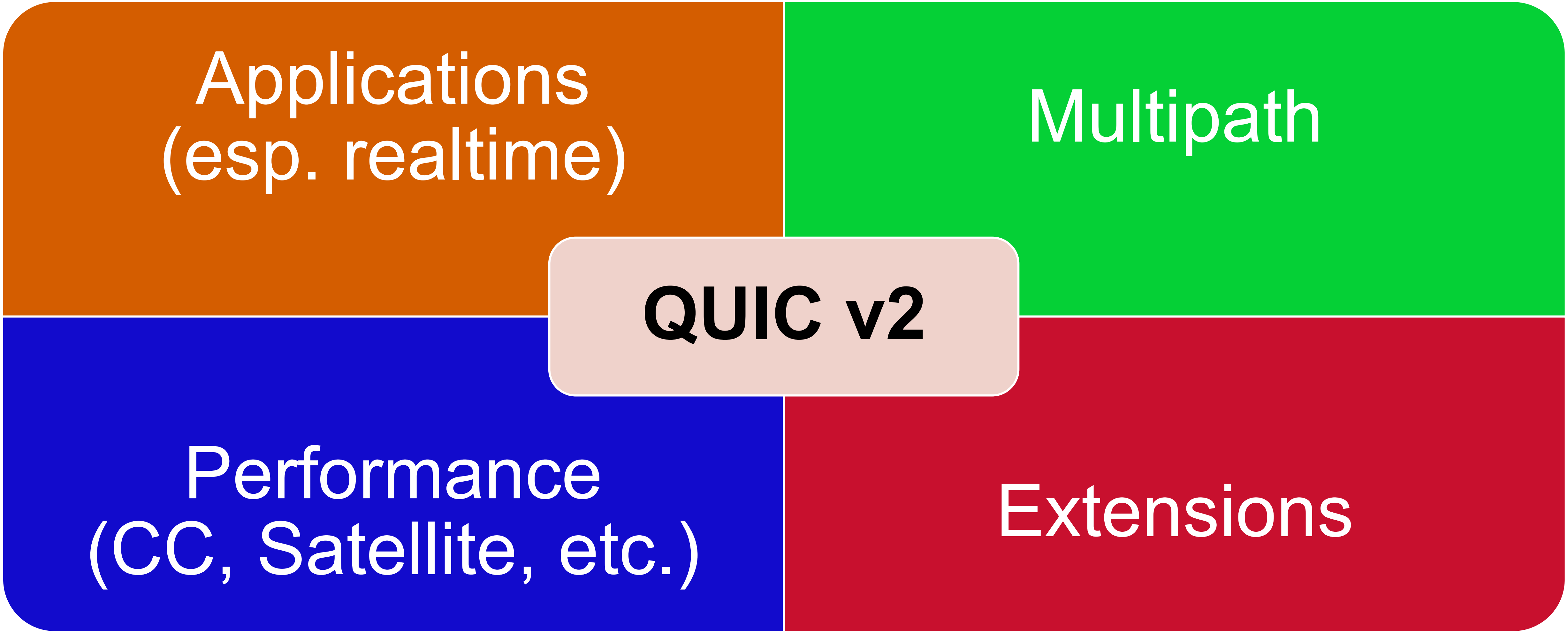
Interop status



server → client ↓	h2o/quicly	quant	ngtcp2	mvfst	picoQUIC	msquic	f5	ATS	quiche	lsquic	ngx_quic	AppleQUIC	quic-go	Quinn	aioquic	~gQUIC
h2o/quicly	VHDCRZSQ UL3	HDC			HDCSU								-			
quant	VHDCRZSQ 3	VHDCRZSQ MBUPEL	VHDCRZSQ MBU 3	VHDCRZQ B 3	VHDCRZSQ MBUP 3	VHDCRZSQ UP 3	VHDCRZSQ UE 3	VHDCRZSQ MB 3	VHDCRZS 3	VHDCRZS MUPE 3	VHDCRZQ 3		-	VHDCRSQ MBUPE	VHDCRZSQ MBUP 3	VHDCRQ 3
ngtcp2	VHDCR3	V	VHDCRZS MBU 3dp		VHDCRZS MBU 3	VHDC UT 3d	VHDCRZS U 3	VHDCRZS MB 3	VHDCRZS 3	VHDCRZS MBUT 3dp			-		VHDCRZS MBU 3dp	VHDCR 3
mvfst				VHDCRZQ BLT 3dp									-			
picoQUIC	VHDCRZSQ T 3	VHDCRZSQ MBUPT	VHDCRZSQ MBU 3	VHDCTRZQ MLT 3	VHDCRZSQ MBAUPLT 3	VHDCRZSQ U 3	VHDCRZS UT 3	VHDCRZSQ B 3	VHDCRZSQ 3	VHDCRZSQ MBAUPT 3		VHDC	-			VHDCRQ B 3
msquic	VHDCRQ	VHDCRZSQ MBULT	VHCRSQ MU	VHDCRZQ MBLT 3d	VHDCRZSQ MBULT 3	VHDCRZSQ MBAUPLT 3d	VHCRS U 3	VHDCRZSQ U 3	VHCDRZQ	VHCRSQ MBU	V	V	-	VHDCSQ BU	VHDCRZSQ MBUL 3d	VHDCRQ B 3
f5	VHDCS T 3d	VHDCS	VHDS 3d	x	VHDCS 3	VHDC T 3d	VHDCS T 3d	VHDCS 3d		VS		VHDC	-		VHDCRZSQ MBAUPLT 3	VHDC 3d
ATS	VHDCRSQ 3	VHDCRSQ M	VHDCRSQ M 3		VHDCRSQ 3	VHDCRSQ 3	VHDCRS 3	VHDCRSQ M 3	VHDCRS 3	VHDCRSQ M 3			-		VHDCRS M 3	VHDRQ 3
quiche													-			
lsquic	VHDCRSQ 3		VHDCRSQ M 3dp	VHDCRQ T 3dp	VHDCRSQ PT 3	VHDCRSQ PT 3d	VHDCRS T 3d	VHDCRSQ 3	VHDCRS 3	VHDCRSQ MPET 3dp [1]			-		VHDCRSQ PT 3dp	VHDCRQ 3d
ngx_quic													-			
AppleQUIC	HDCS 3						HDS 3d					HD	-			V
quic-go													-			
Quinn		VHDCRZS BU	VHDCRZ BU 3	VHDCRZS B 3	VHDCRZS BU 3	VHDCRZS BU 3		VHDCRZS	VHDCRZS B 3	VHDCRZS BU 3			-	VHDCRZSQ BU 3		VHDCRS B 3
aioquic	VHDCRZSQ 3	VHDCRZSQ BU	VHDCRZSQ MBU 3dp	VHDCRZQ BLT 3dp	VHDCRZSQ MBUPLT 3	VHDCRZS MBUPL 3d	VHDCRZS U 3d	VHDCRZSQ MB 3	VHDCRZS 3	VHDCRZSQ MBUPT 3dp			-		VHDCRZSQ MBUPLT 3dp	VHDCRQ 3d
~gQUIC	VHDRZ 3	V	VHDRZ 3d	-	VHDCRZ 3	VS	VHDCRZS 3d	VHDS	VHDRS B 3	VHDCRS 3		-	-		VHDRZS B 3d	VHDCR B 3d

https://docs.google.com/spreadsheets/d/1D0tW89vOoaScs3IY9RGC0UesWGAwE6xyLk0l4JtvTVg/edit#gid=117825384

Beyond QUIC v1



Applications (esp. realtime)

Multipath

QUIC v2

Performance (CC, Satellite, etc.)

Extensions

Before I go...

How to participate?

QUIC

- QUIC WG is open to all
  - Use the mailing list
  - Discuss issues/PRs on GitHub
  - Participate in meetings
- **https://quicwg.org/** will get you started
- You can talk to us first, too
- “Note Well” – disclose IPR

IETF

- IETF is open to all
- 3x meetings/year, next:
  - Vancouver, March
  - **Madrid, July**
  - Bangkok, November
- **Grants** for academics:
  - ACM/IRTF ANRW workshop (travel grants, only students)
  - IRTF Chair discretionary fund (need strong reason)

GitHub

- **https://quicwg.org/** links to a list of implementations
- Many are open source and live on GitHub
- Contact maintainers and start issues/PRs

Please take a moment to rate this session.

Your feedback matters to us.