February 2, 2025

QUIC

vs.

Middleboxes

Lars Eggert, lars@eggert.org, FOSDEM 2025

Mozilla

QUIC: a fast, secure, evolvable transport

Fast.

Better user experience than TCP/TLS for HTTP/2 and other content.

Evolvable.

Prevent network from ossifying, deploy new QUIC versions quickly.

Secure.

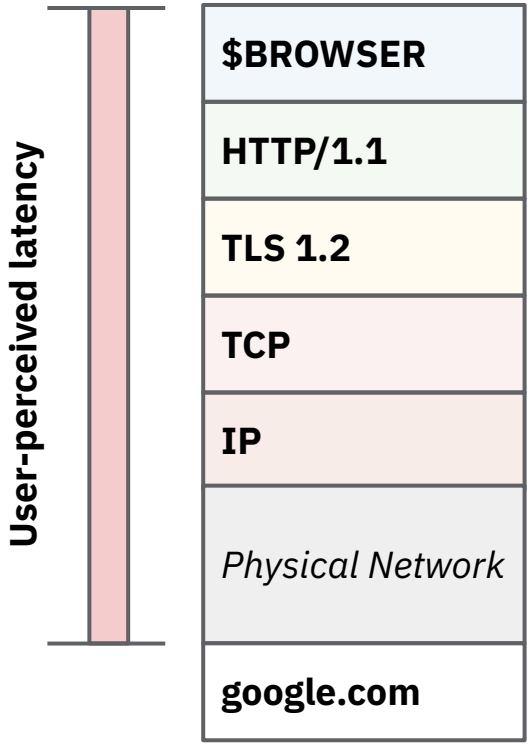
Always-encrypted end-to-end security, resist pervasive monitoring.

Transport.

Support all TCP content & more (realtime media, etc.)
Provide better abstractions, avoid known TCP issues.

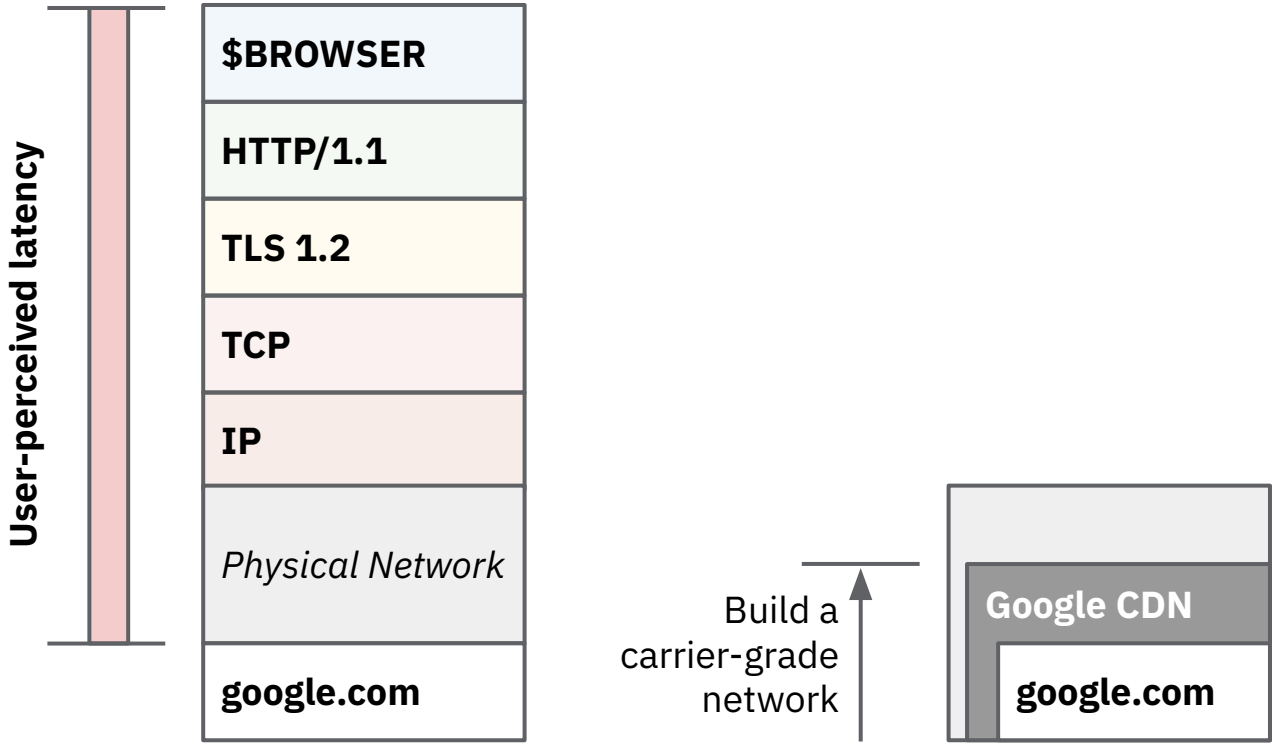
How do you make the web faster?

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



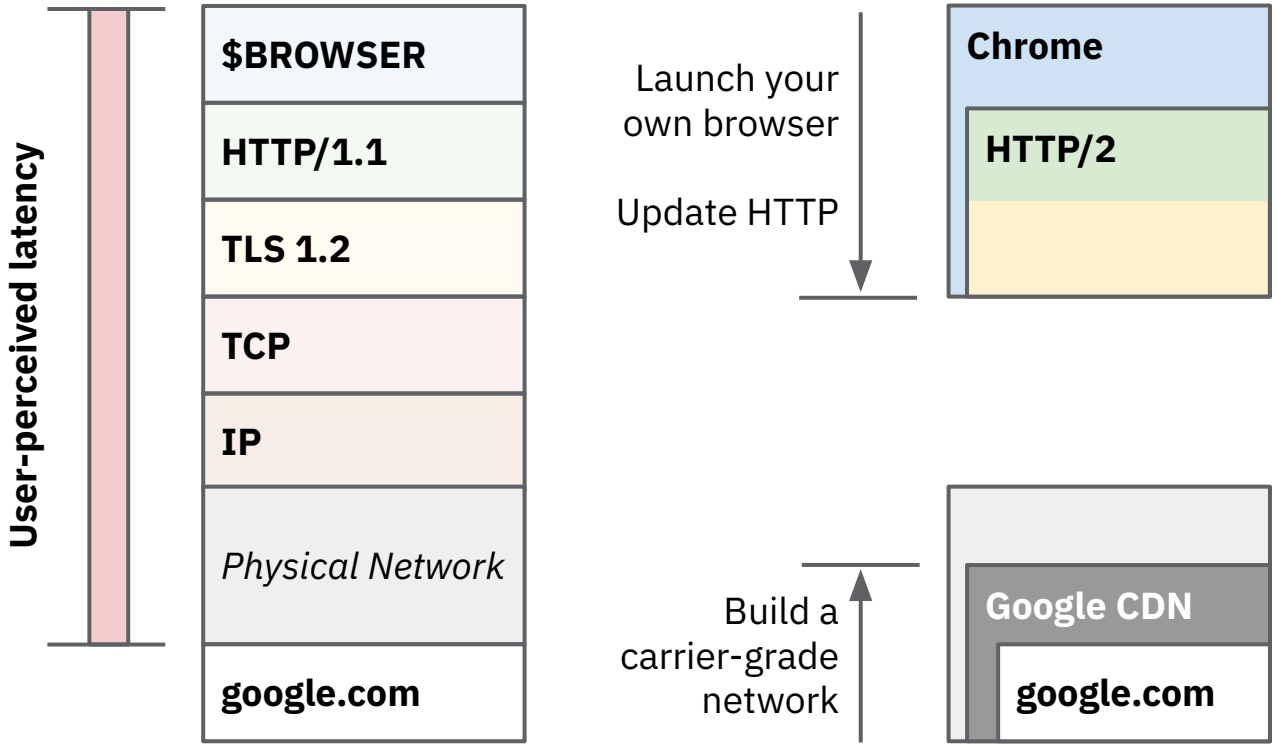
How do you make the web faster?

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



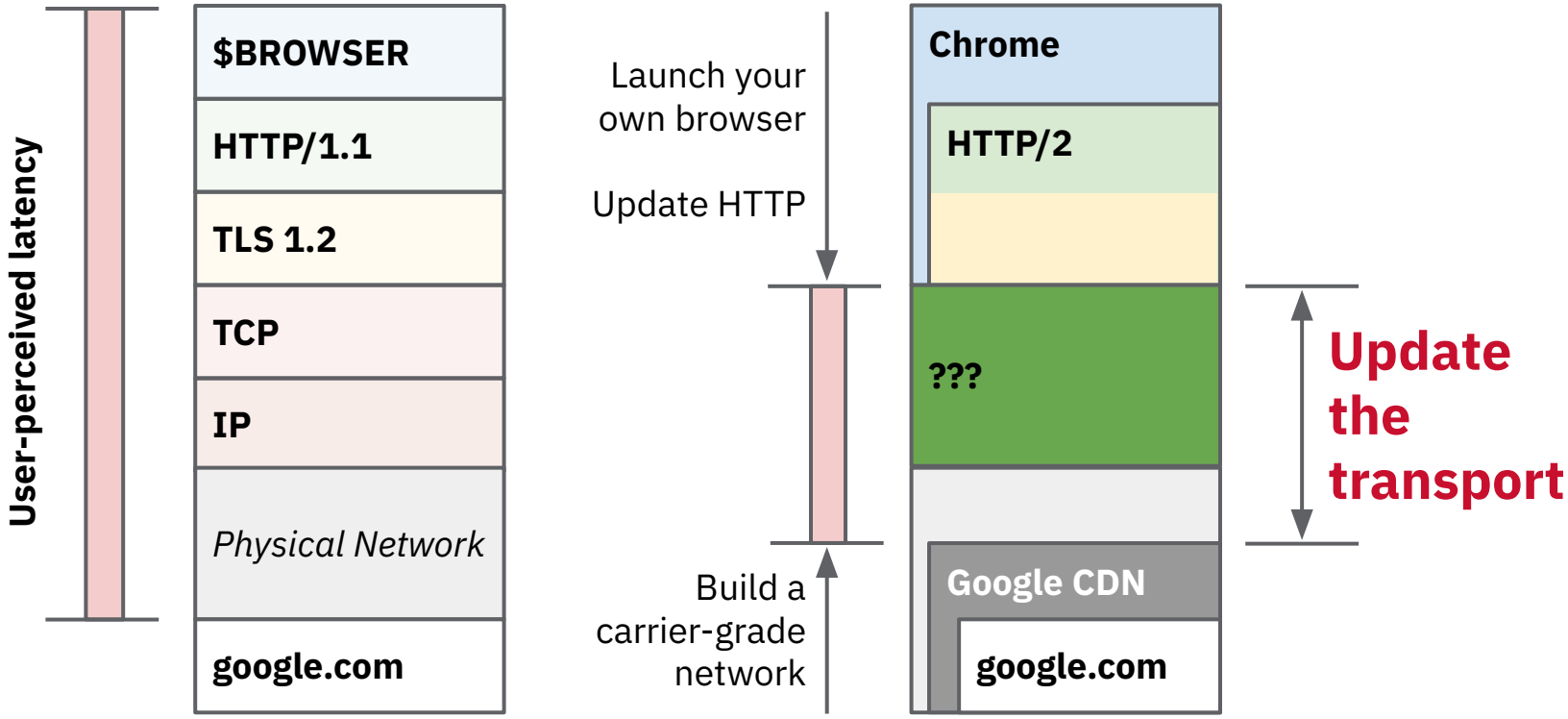
How do you make the web faster?

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



How do you make the web faster?

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



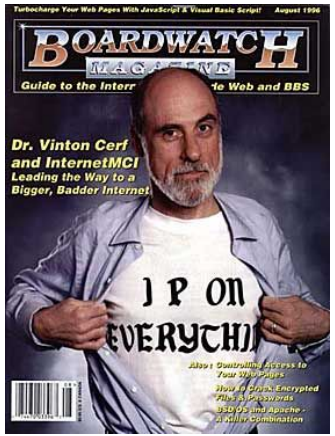
Internet transport

Mozilla

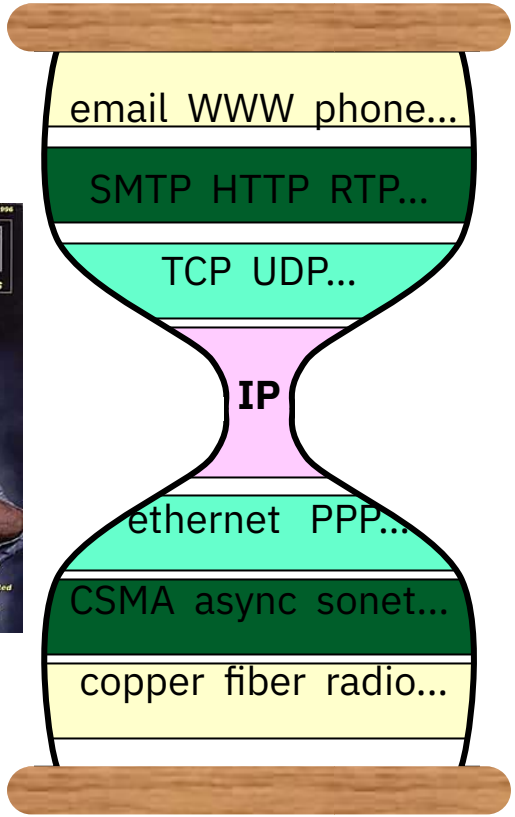
The Internet hourglass

Classical version

- Inspired by OSI “seven-layer” model
- “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.

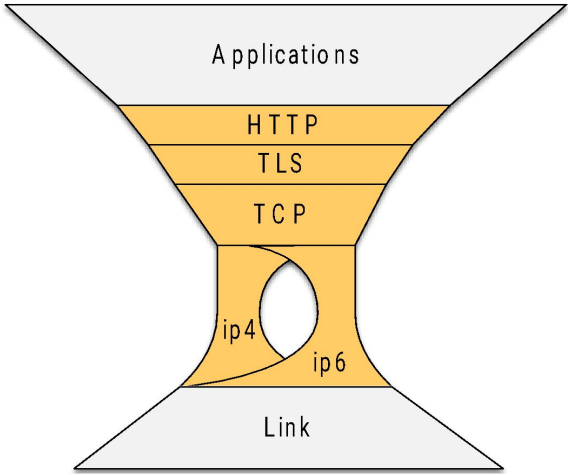


Steve Deering. Watching the Waist of the Protocol Hourglass. Keynote, IEEE ICNP 1998, Austin, TX, USA. http://www.ieee-icnp.org/1998/Keynote.ppt

The Internet hourglass

2015 version (ca.)

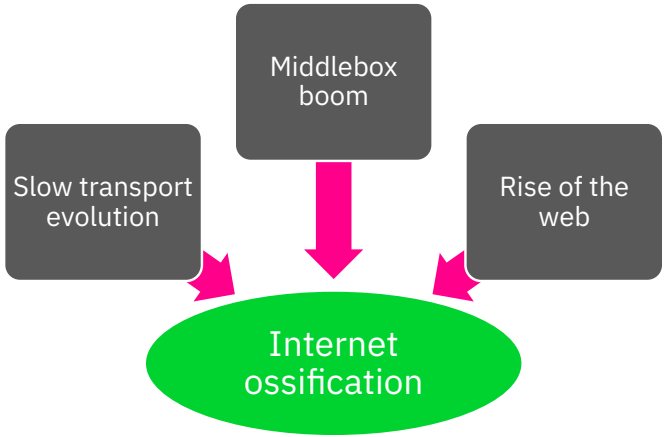
- The waist has split: **IPv4** and **IPv6**
- **TCP** is drowning out UDP
- **HTTP** and **TLS** are *de facto* part of transport
- Consequence: **web apps** on IPv4/6



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



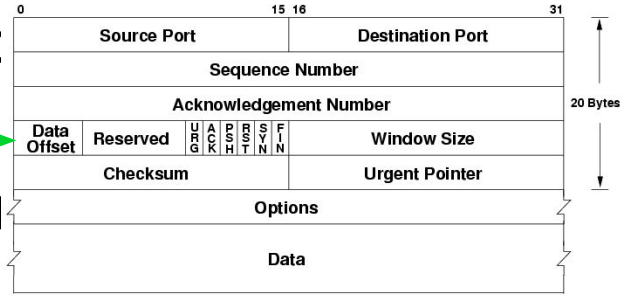
Example ossifications

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

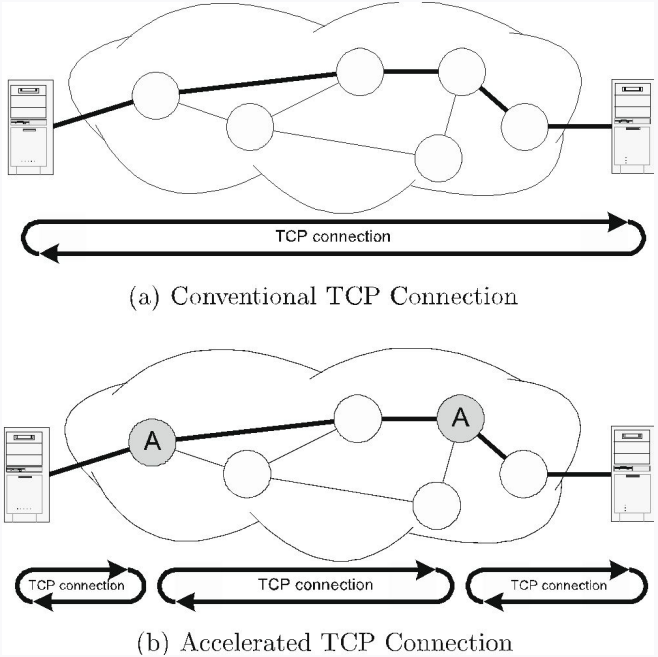
TCP is not aging well

- **We're hitting hard limits** (e.g., TCP optic
  - 40B total (15 * 4B - 20)
  - Used: SACK-OK (2), timestamp (10), wi
    (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)



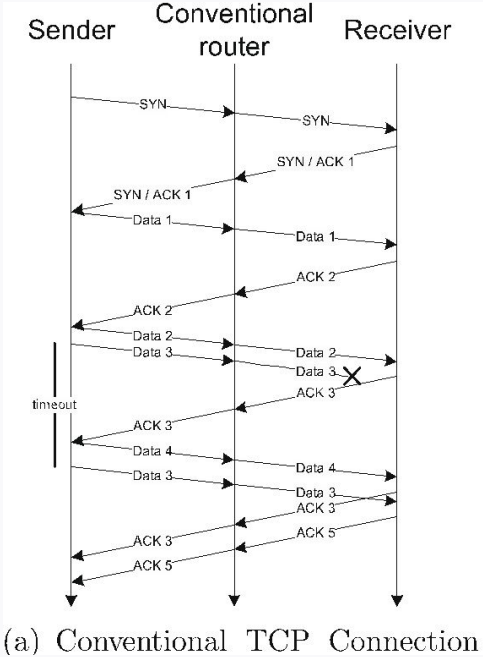
By Ere at Norwegian Wikipedia (Own work) [Public domain], via Wikimedia Commons

Middleboxes meddle
e.g., “TCP accelerators”

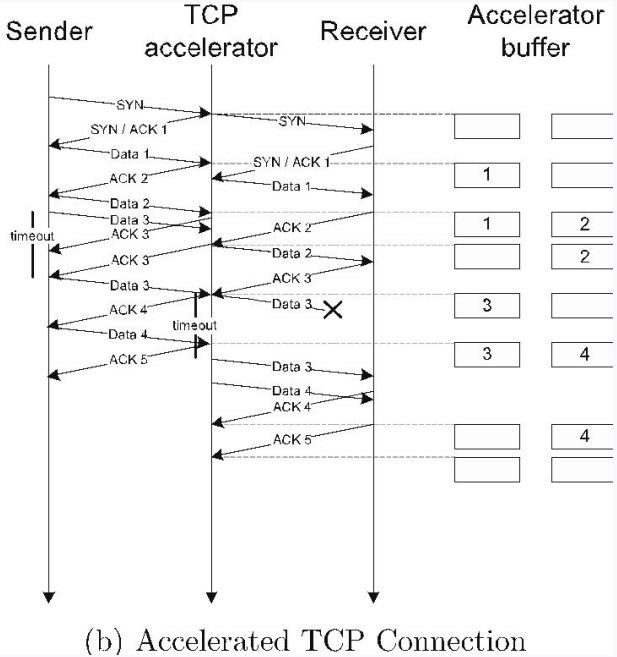


(a) Conventional TCP Connection

(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
e.g., nation states as attackers

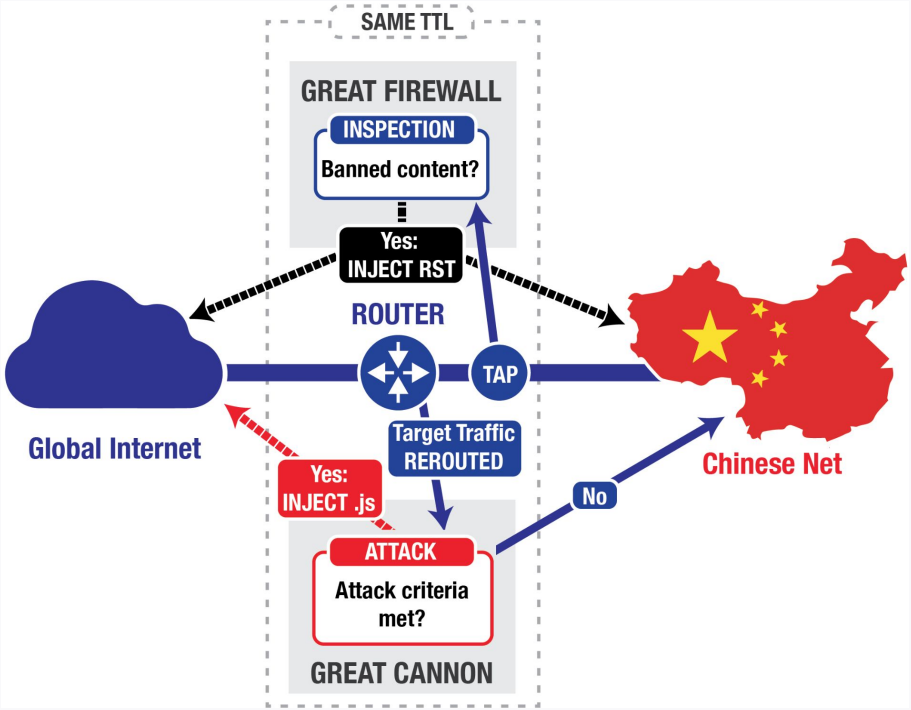
TOP SECRET//COMINT//REL TO USA, AUS, CAN, GBR, NZL

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.

RFC 7528 Pervasive monitoring is an attack

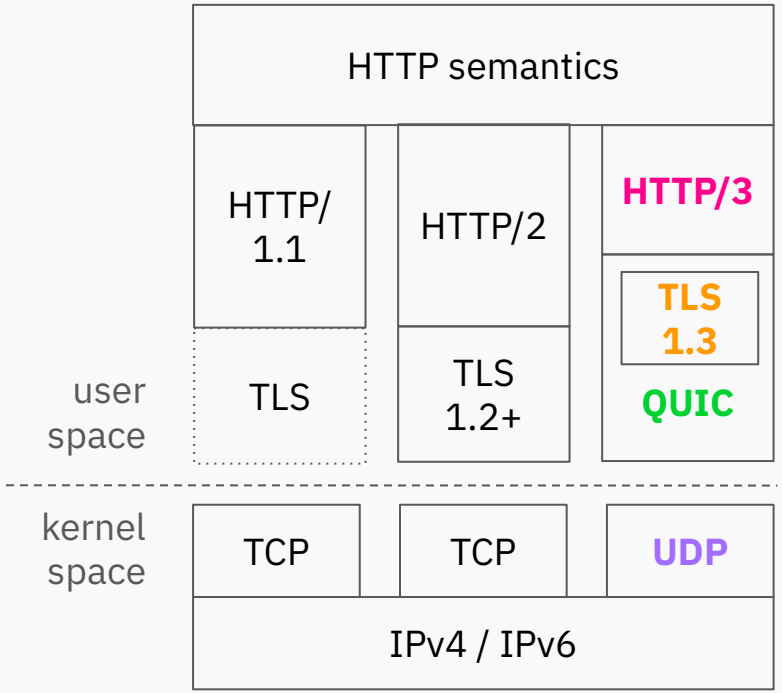
- IETF (& wider) community consensus that pervasive monitoring is an attack
- Agreement to mitigate pervasive monitoring
- What does “mitigate” mean?
- To many, ”encrypt as much as possible”

- **But what else could we do?**

QUIC

Mozilla

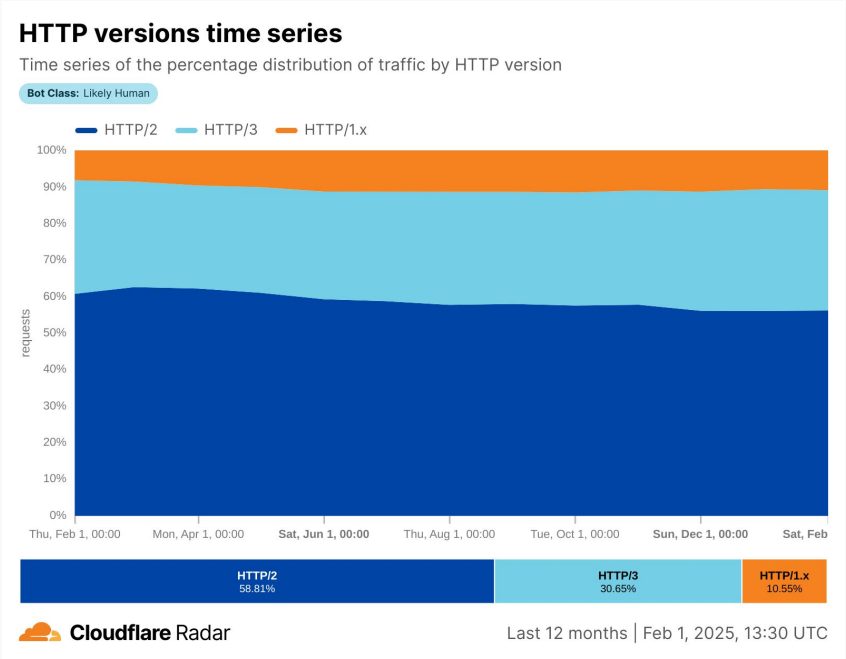
QUIC



2013 Experiment at Google

2016 IETF WG started

2021 RFCs 8999-9002



Why UDP?

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

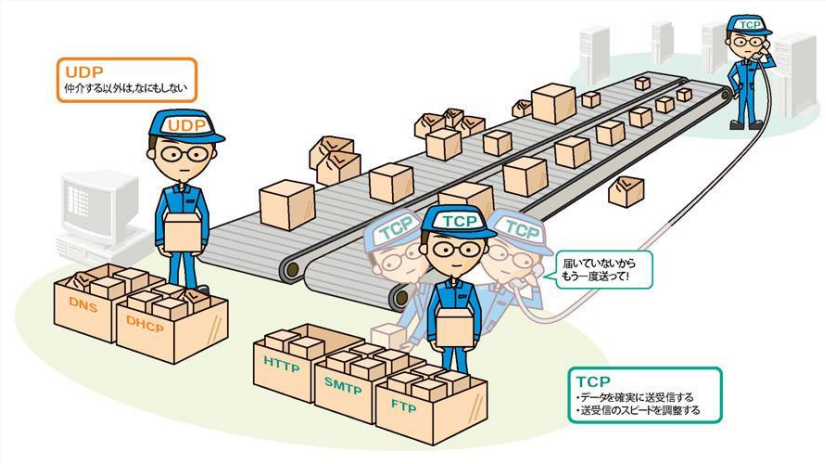


Image from http://itpro.nikkeibp.co.jp

Why congestion control? (Duh)

- 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

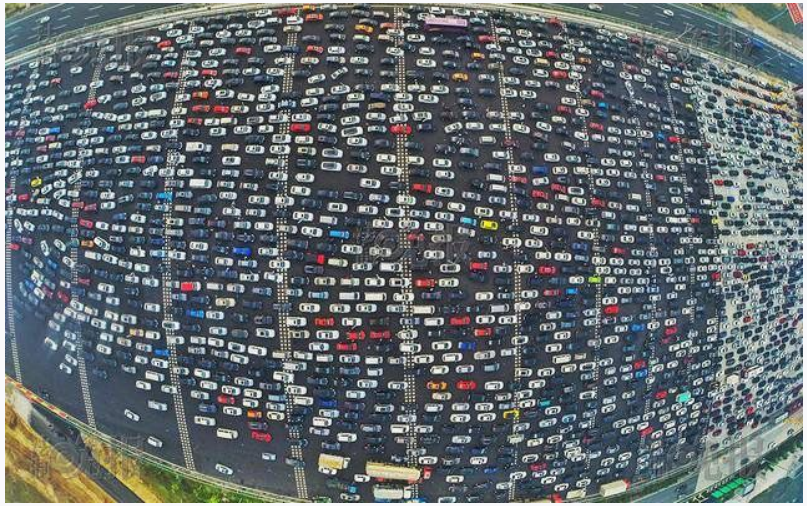
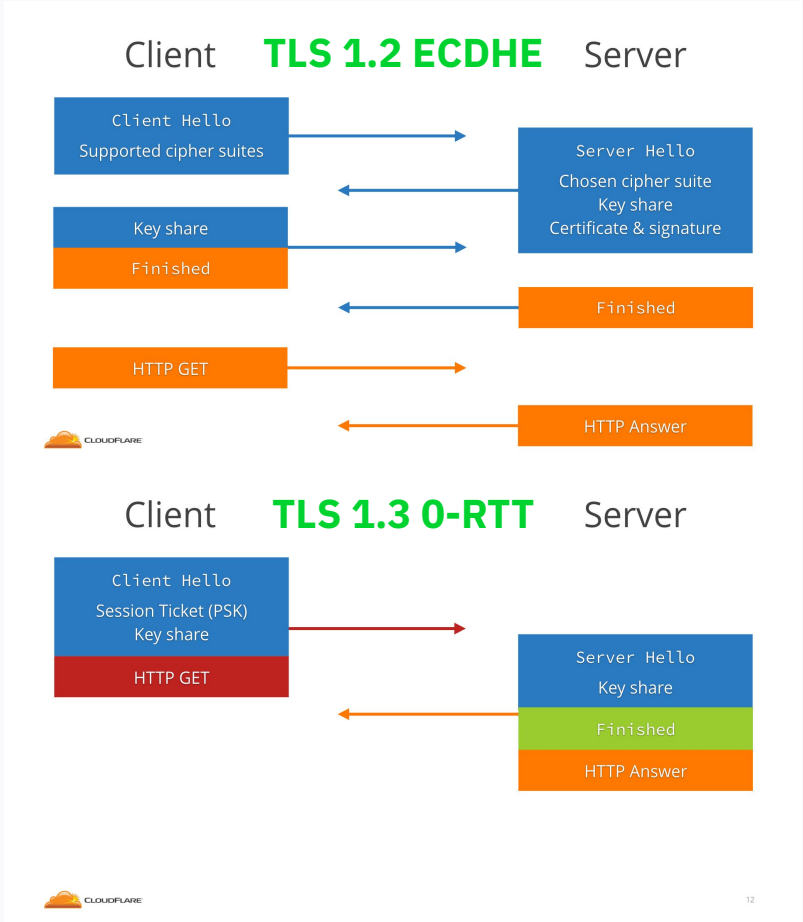


Image from People's Daily, http://people.cn/

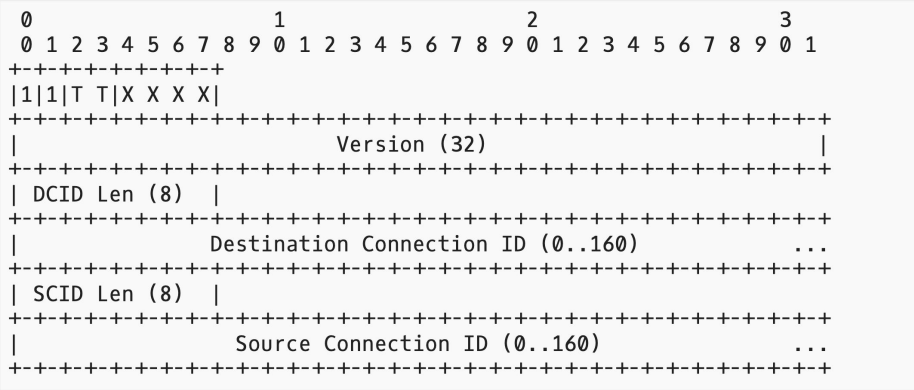
Why TLS? (Duh)

- **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 adds new features (0-RTT!)



Minimal network-visible header

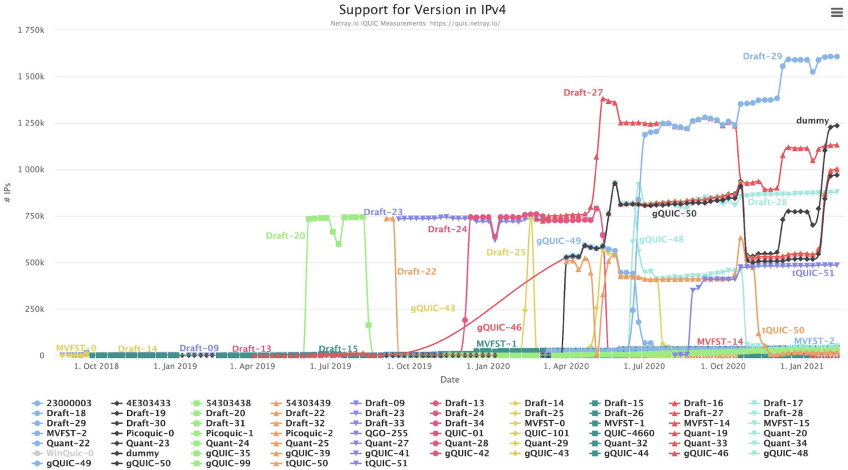
- With QUIC, the network sees:
  - Packet **type** (partially obfuscated)
  - QUIC **version** (only in long packet header)
  - Destination **CID**
  - Packet **number** (obfuscated)
- With TCP, also
  - ACK numbers, ECN information
  - Timestamps
  - Windows & scale factors



- Also, entire QUIC header is **authenticated**, i.e., not modifiable

Version negotiation

- 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 & length of version & CIDs in LH
  - Location & length of CID in SH
  - Version negotiation server response
- 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

Thank you

Lars Eggert, lars@eggert.org
lars@social.secret-wg.org

Lars Eggert

Firefox Networking

Browser meets network

- **HTTP:** H1, H2, H3
- **Transports:** TCP, TLS, UDP, QUIC
- **Naming:** DNS, DoH
- **Proxying:** SOCKS, MASQUE, OHTTP
- **Features:** ECH, WebSocket, WebTransport, PQC
- **Related:** Disk cache, cookies, WebRTC, WebPKI, TRR, Happy Eyeballs

Also: Security, privacy, operational aspects, performance, telemetry and OS interfaces for all of these.

DoH

- Improve Trusted Recursive Resolver (TRR) program
- Fallback via *Discovery of Designated resolvers*
- Rollout on Android
- Open questions
  - Performance impact of DoH
  - Global deployment
  - Optimizations (e.g., QUIC 0-RTT)

Getting off getaddrinfo

- Want: asynchronous resolution of multiple RRs
  - esp. relevant for Happy Eyeballs v3
- Want: HTTPS records on all platforms
  - Thus more QUIC
- Want: Bypass various OS bugs
- **Open questions**
  - Operating specific optimizations

Happy Eyeballs v3

- v1 in Firefox too easily falls back to H1/H2, less QUIC/H3
- New in v3
  - **Explicit design goal: prefer IPv6 and QUIC**
  - Support for HTTPS resource records
  - Discovery of alt. endpoints, protocols (H3), ECH configs
- **Open questions**
  - Impact on connection establishment latency
  - Constant tuning (e.g., *Connection Attempt Delay*)

Encrypted Client Hello

- TLS SNI is the last bit of plaintext
- ECH can encrypt that – but!
  - Low ECH adoption (only Cloudflare, basically)
  - Easy to block based on outer SNI
  - Needs DoH (or other encryption) to get ECH config
- **Open questions**
  - How do we get more ECH?
  - How do we get more DoH? (Or what instead of DoH?)