CS-340 Introduction to Computer Networking

Lecture 18: QUIC and Course Review

Steve Tarzia

Last Lecture: Mobility

- IP addressing was designed for a static world.
- **Mobile IP** gives host a permanent **home address**.
  - Registers with **Foreign Agent** which registers with **Home Agent**.
  - Home agent encapsulates received traffic in **IP tunnel**, forwards traffic to foreign agent at **care-of address**.
- Smartphone push notifications also use **location registration**.
- **Handoff**: set up the 2nd channel, transfer connection, then close 1st.

Let’s Review some Networking Themes.

Decentralization

- There is very little *centrally-controlled* infrastructure for the Internet.
  - ICANN just controls the distribution of IP address and domain names.
- Internet standards are developed by the IETF through an open process, leading to RFCs.
- Internet's physical links added *ad-hoc* by pairs of AS's choosing to *peer*.
- DNS: 13 root server IP addresses, then top level domains (TLDs).
  - DNS servers at the edge of the network *cache* results.
- BGP uses Distance-Vector algorithm to compute shortest path.
- TCP congestion control mitigates *core* congestion from observations at the *edge*.

Fault tolerance

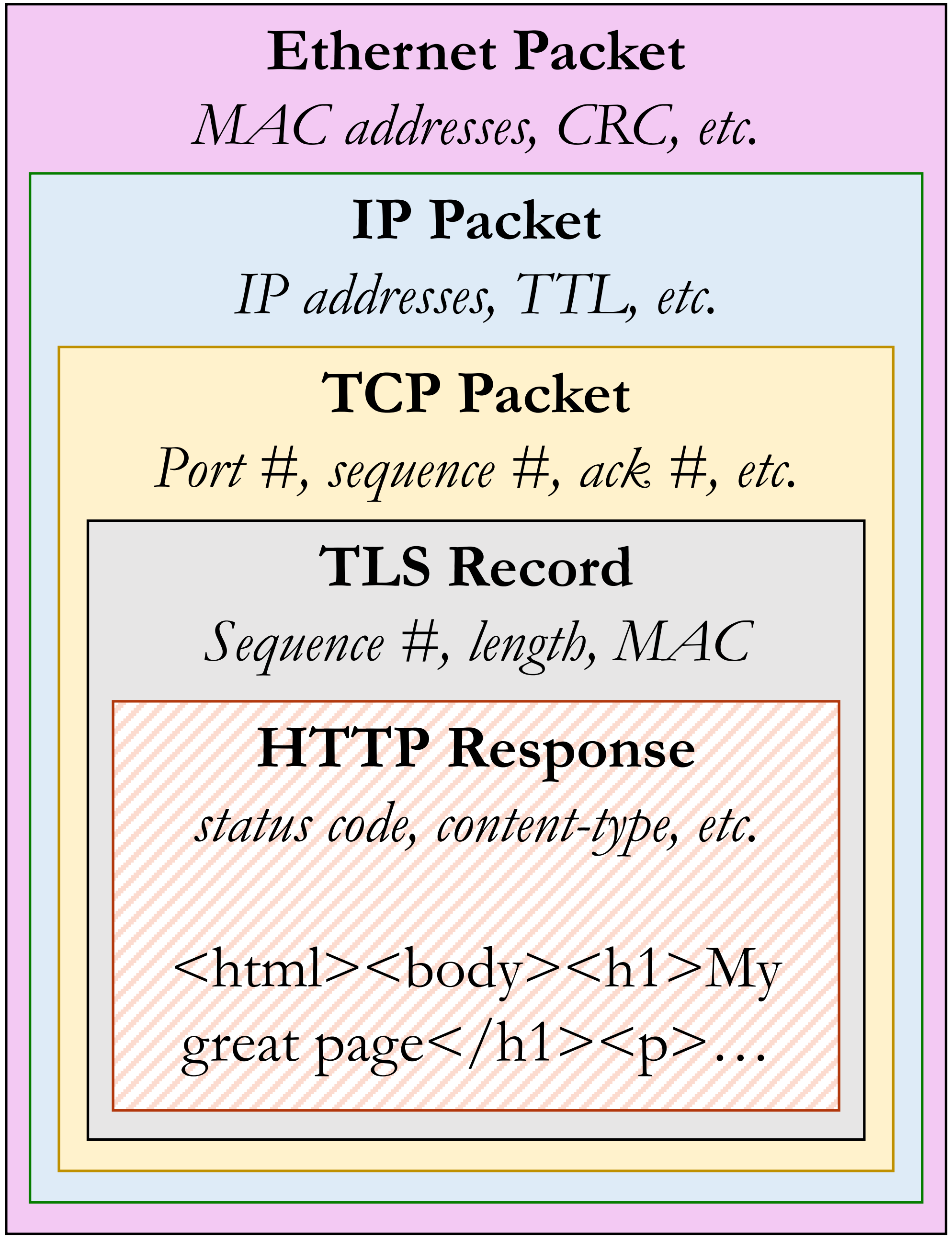
- Assume that links and routers are unreliable:
  - Bits can be flipped, • Packets can be dropped
  - This software/protocol-level assumption simplifies hardware design
- Bit error checking is included in Ethernet, IPv4, TCP/UDP headers.
- BGP allows routes to change in response to broken links.
- TCP provides delivery confirmation, retransmission, and ordering.
- TLS provides message integrity (with digital signatures).
- HTTP response code can indicate an error
  - (eg., 404 Not Found, 500 Internal Server Error)

Empiricism

- The design of networking protocols is mostly **experimental.**
- Networking researchers build systems and test them at scale.
- There was only a little bit of theory, proofs, and math in this class. Eg:
    - Convergence of BGP to the shortest path in the long run.
    - Fair share allocated to each TCP Reno connection.
    - Formulas for peak performance of link-layer protocols.
- Internet traffic has complex patterns that are difficult to model and theoretically optimize.

Separation of concerns

- **Link layer**: shares a physical channel among several transmitters/receivers
- **Network layer**: routes from source to destination, along many hops.
- **Transport layer**:
  - Multiplexing >1 connection per machine
  - Ordering, • Acknowledgement, • Pacing
- **TLS**:
  - Encryption, • Authentication.
- **HTTP layer**:
  - Resource urls, • Response codes,
  - Caching, • Content-types, • Compression



Ethernet Packet

MAC addresses, CRC, etc.

IP Packet

IP addresses, TTL, etc.

TCP Packet

Port #, sequence #, ack #, etc.

TLS Record

Sequence #, length, MAC

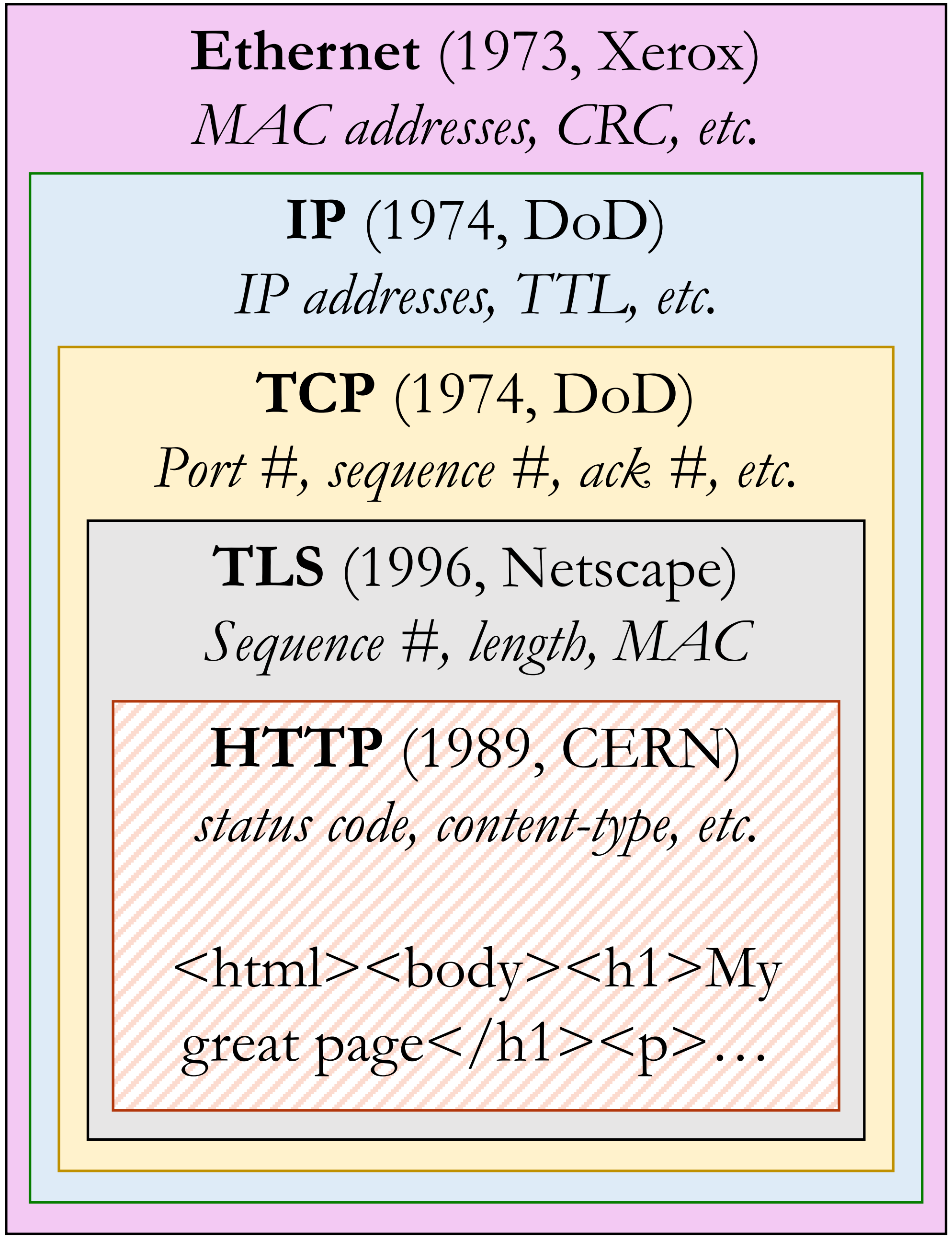
HTTP Response

status code, content-type, etc.

<html><body><h1>My great page</h1><p>…

Why are protocols **layered?**

- **Historical reasons**
  - Lowest-level functionality was developed first, and upper layers were added to build more complex and useful services.
  - Different use-cases and different engineers drove the development of each layer.
  - Various protocols coexisted and competed at each layer.
- **Separation of concerns**
  - Networking is complex problem.
  - Easier to solve by breaking into several independent sub-problems.



Ethernet (1973, Xerox)
MAC addresses, CRC, etc.

IP (1974, DoD)
IP addresses, TTL, etc.

TCP (1974, DoD)
Port #, sequence #, ack #, etc.

TLS (1996, Netscape)
Sequence #, length, MAC

HTTP (1989, CERN)
status code, content-type, etc.

<html><body><h1>My great page</h1><p>…

And now a modern update…

Pioneering the future
Markus Schäfer
Member of the Board of Management of Daimler AG & Mercedes-Benz AG, responsible for Group Research and Mercedes-Benz Cars Development, Procurement and Supplier Quality
Katie Linendoll
Consumer technology expert
Customizing the future
Bettina Fetzer
Vice President Marketing Mercedes-Benz Cars

Layered designs create constraints

- Lower layers cannot be controlled by upper layers (cost of abstraction)
- Some hacks exist, like TCP connection parameters allowing application to enable/disable Nagle's algorithm.
- Vast majority of Internet traffic is HTTP+TLS+TCP+IP
- We can improve performance significantly by combining some of these protocols.
- At right, we see that nowadays most web traffic is **encrypted**.
  - Protocols should be designed to make the common case as efficient as possible.

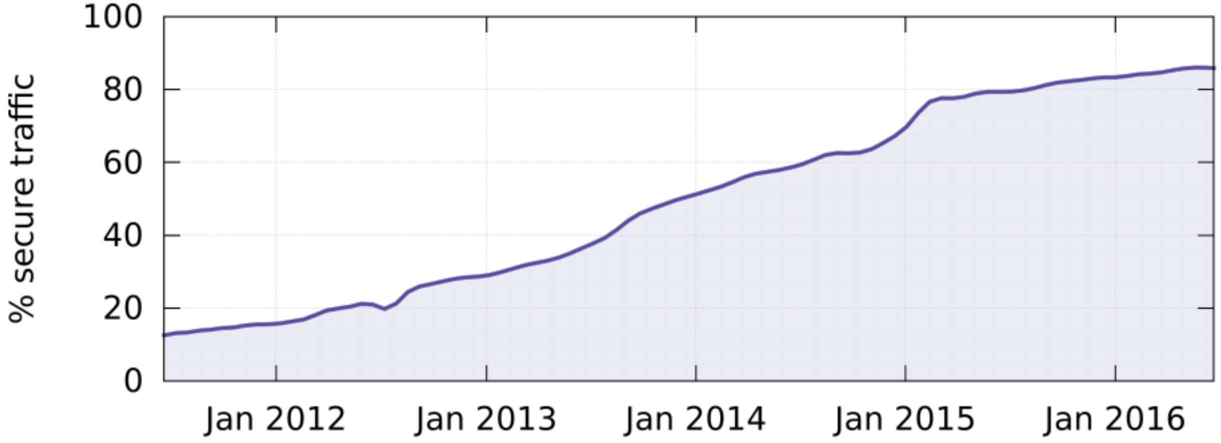


Figure 3: Increase in secure web traffic to Google's front-end servers.

Quick UDP Internet Connections

- Replaces TLS and TCP.
- QUIC was designed alongside a new version of HTTP to work with it.
- Development started at Google in 2012
- Deployed in their Chrome browser, and services like YouTube.

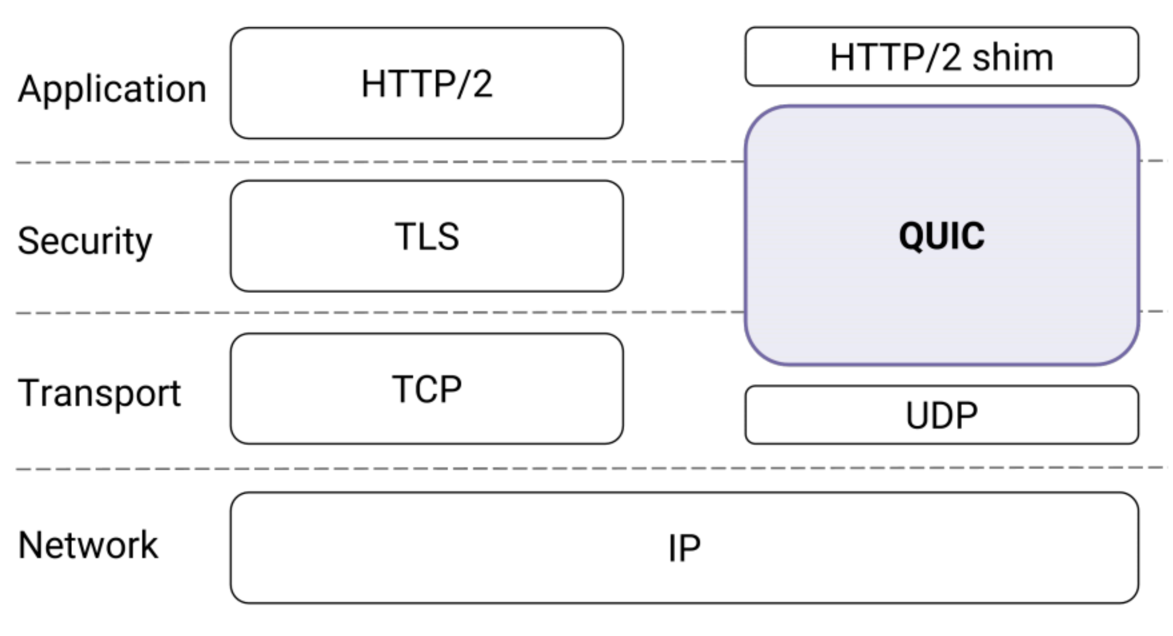


Figure 1: QUIC in the traditional HTTPS stack.

- 5-10% of current Internet traffic is QUIC, as of late 2018.
  - Wireshark trace on your machine will show QUIC traffic if you use Chrome
- HTTP-over-QUIC will be renamed to "**HTTP version 3**"
  - IETF announced this on November 2018 (both names are used nowadays).

- QUIC is based on lessons learned from SPDY and HTTP/2
  - Solves problems caused by abstracting and isolating network layers ….

Why UDP?

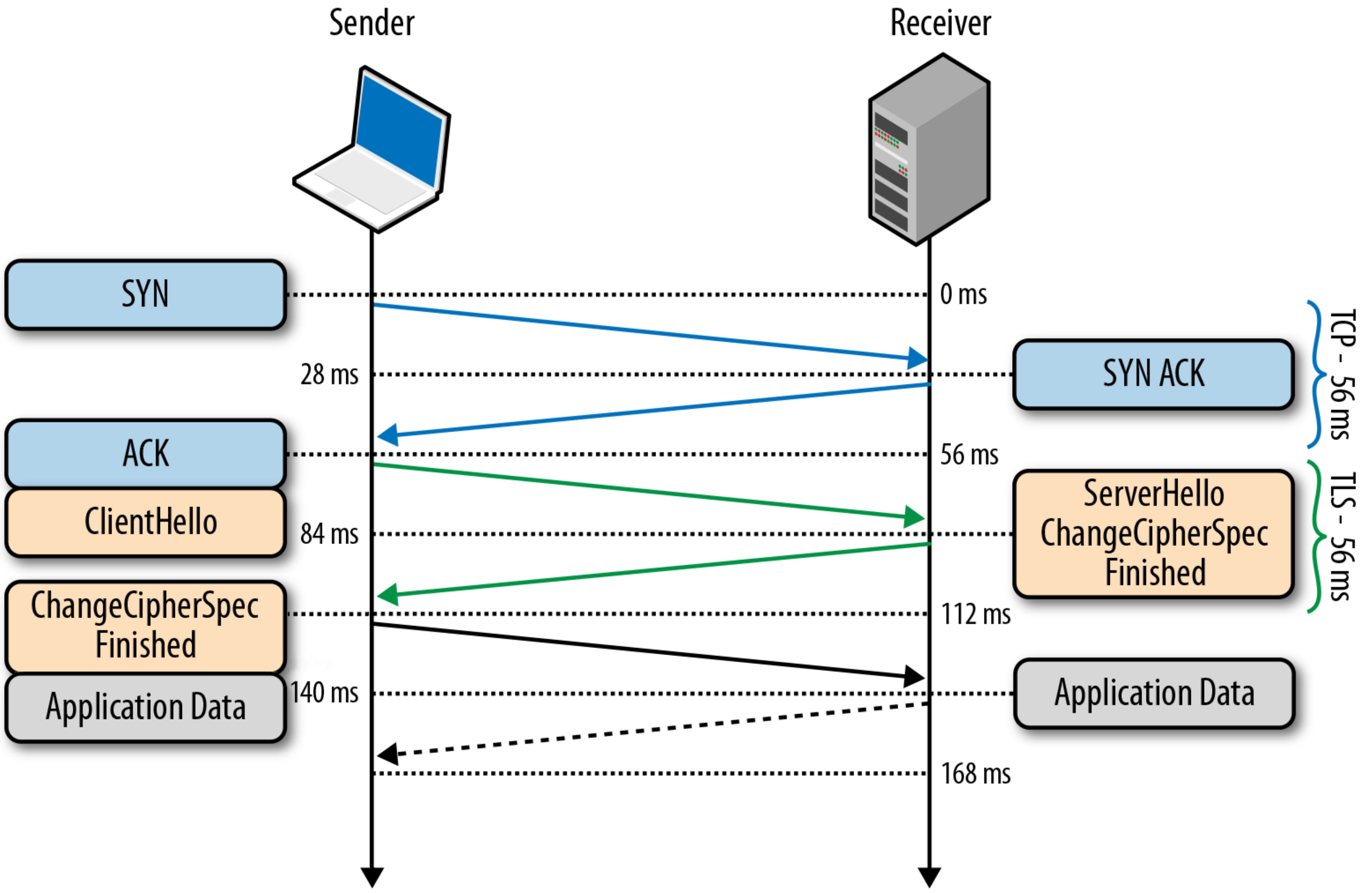
- UDP allows the user-level application to control each packet.

Like Project 2!

- QUIC implements **reliable transport in an application library**, rather than letting the OS's TCP library handle it.
  - QUIC implements retransmission, handshakes, pacing, etc. on top of UDP.
- This allows Google's Chrome browser to include QUIC without modifying the underlying OS.
- Ideally, it would make sense for QUIC to be implemented as an *alternative* to UDP and TCP, directly on top of the IP layer.
  - However, existing routers, firewalls, NATs don't know about QUIC.
  - *Lesson from IPv6:* incompatibility with old hardware will block adoption.

TCP/TLS Problem #1: Two Handshakes

- TLS cannot start its key exchange until after the TCP handshake
- The speed of light is a hard limitation, so **latency** will become the dominant factor influencing performance of future networks.



QUIC Handshakes

- **Worst case:** combine TCP+TLS handshakes.
  - Send public key and other encryption parameters in the **first packet.**
  - Server "REJ" response provides pub key.
- **Best case: zero-RTT handshake**
  - Remember long-lived public key from previous connection to the same server.
  - Send encrypted request along with the client's pub key.

CHLO = Client Hello
REJ = Rejection

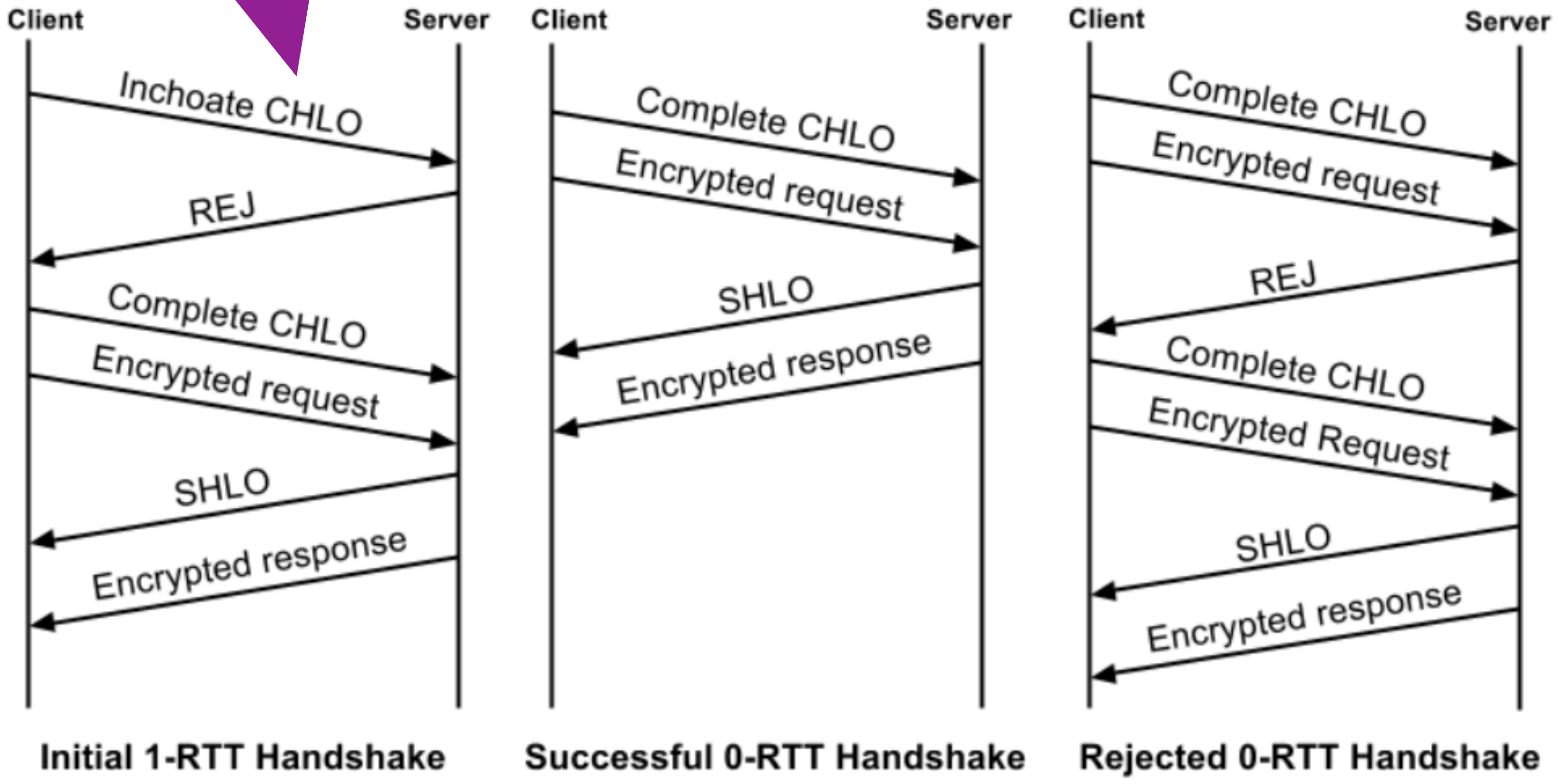


Figure 4: Timeline of QUIC's initial 1-RTT handshake, a subsequent successful 0-RTT handshake, and a failed 0-RTT handshake.

HTTP Problem #2: Multiple streams are expensive

- Web browsers make many requests to load a page:
  - Initial HTML page, plus many CSS, Javascript, AJAX, and Image requests, all of which might be retrieved from the same server.
- Each request needs:
  - its own TCP+TLS connection (with significant handshake setup **latency**),
  - or to **wait** for an existing request to finish, in order to reuse its connection.
- SPDY and HTTP/2 solved this problem by multiplexing many HTTP connections on a single TCP/TLS socket:
  - Allows a single socket to handle many HTTP requests **in parallel**.
  - Adds a **stream id** to distinguish multiple streams in a single socket.
  - HTTP/2 is used in about 50% of web traffic.

HTTP/1.1

From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

- Multiple TCP/TLS handshakes are required
- Client must read response before making requests for related content.

HTTP/2

From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

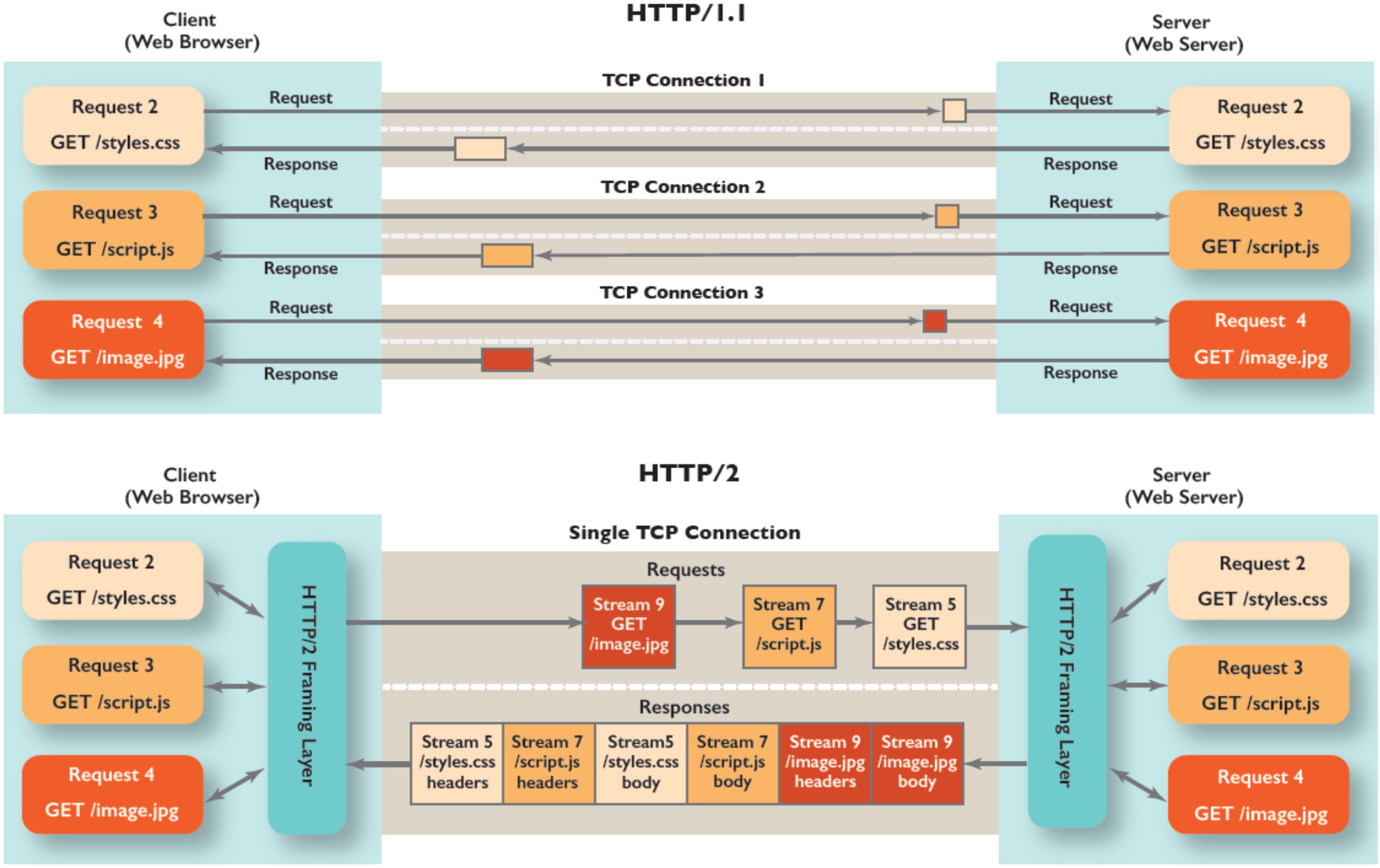
- Only one TCP/TLS connetction/handshake is required for multiple HTTP requests.
- Latency of later requests is reduced, due to connection reuse.
- Fewer TCP port numbers are reserved on the client side

HTTP/2 with Server Push

From HTTP/2 IN ACTION by BARRY POLLARD, Copyright 2018.

- Server predicts that client will request certain documents next.
- Multiple responses are sent along with the initial request and cached locally.
- Client will use the predictively cached responses if necessary.

HTTP/1.1 vs HTTP/2 (multiple streams)



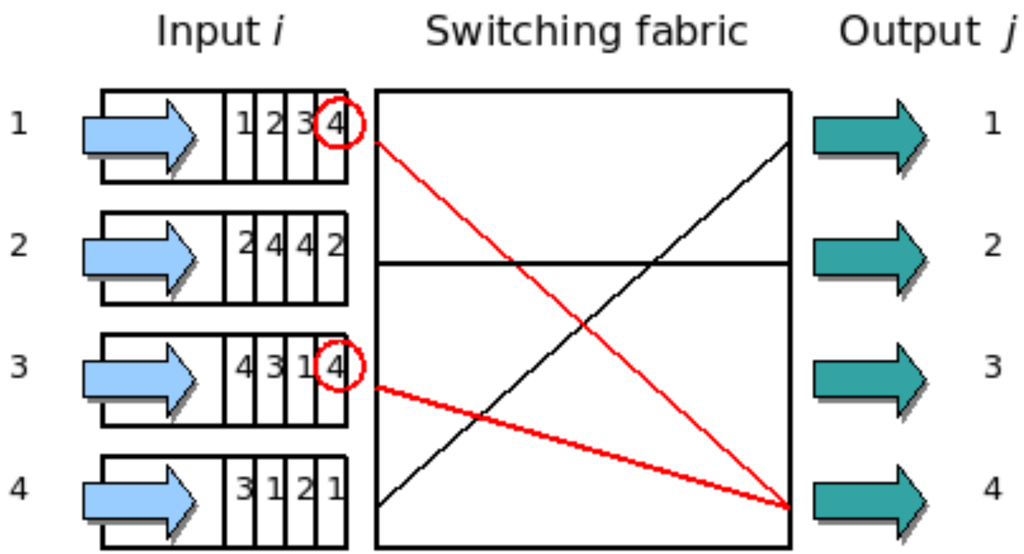
HTTP/2 over TCP is prone to **head-of-line blocking**

- Remember that TCP is a stream-oriented protocol.
- If a packet is dropped, then later packets in the stream can be buffered by the receiver, but they cannot be delivered up the to application layer until the dropped packet is retransmitted and received.
  - In other words, the application must receive data in order (in a stream).
- HTTP/2 transmits many HTTP requests' data in a single TCP stream.
- The dropped packet may involve only one (or a subset) of the HTTP streams, so it's not necessary to block them all. TCP forces this!

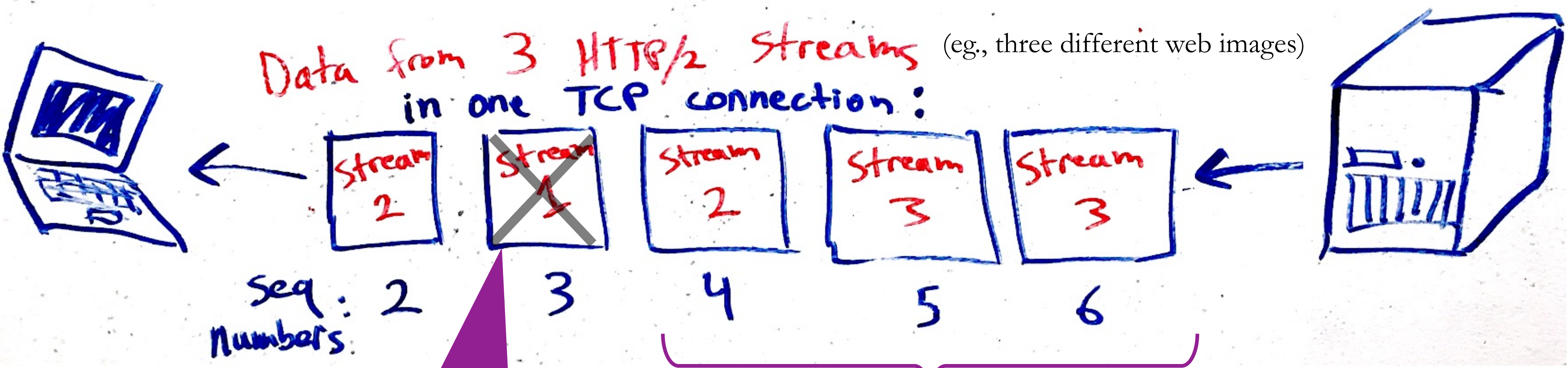
Head-of-line blocking *(in general)*

- HOL blocking is when an item at the head of a queue unnecessarily blocks items behind it.
- At right, two input ports are trying to use Output 4.
- If Input 3 is chosen to proceed, then Input 1 will have to wait.
- Thus, the next item in Input 1's queue (destined to the idle Output 3) is stuck waiting for no good reason.
- The fundamental problem is that we're using a FIFO queue for items with no ordering dependence. We should pop whichever item is ready.

Example in a switch or router:



HTTP/2 head-of-line blocking illustration



(eg., three different web images)

TCP segment 3 is lost in transit.

OS will not deliver data in later segments to application until after segment 3 is retransmitted.

- Data in segments 4,5,6 is for HTTP/2 responses unrelated to the lost segment.
- There's no need to wait, but the abstraction provided by TCP forces the wait.

QUIC prevents head-of-line blocking

- By coordinating transport and application layers, QUIC can allow HTTP streams unaffected by a packet loss to continue while delaying those that must be delayed.
- QUIC data is sent by UDP, so it’s **not** buffered by the OS to be delivered “in the right order.”

- In other words, QUIC uses one handshake to support an arbitrary number of *independent* data streams.

Problem #3: HTTP headers are inefficient

- Human-readable headers in HTTP waste space.
  - Eg., "Content-Length:" is 15 bytes = 120 bits!
  - Compare to DNS, which is very space-efficient.
- Gzip compression of HTTP *body* is possible using a Content-Encoding header, but headers are always plain ASCII text.
- Some headers (specifying client capabilities) are repeated in each new request to the server (because HTTP is stateless).

QUIC solution:

- Compress headers, using a scheme called QPACK.
- *Downside:* Lose human-readablilty. Must debug w/a tool like Wireshark.

Problem #4: Mobility

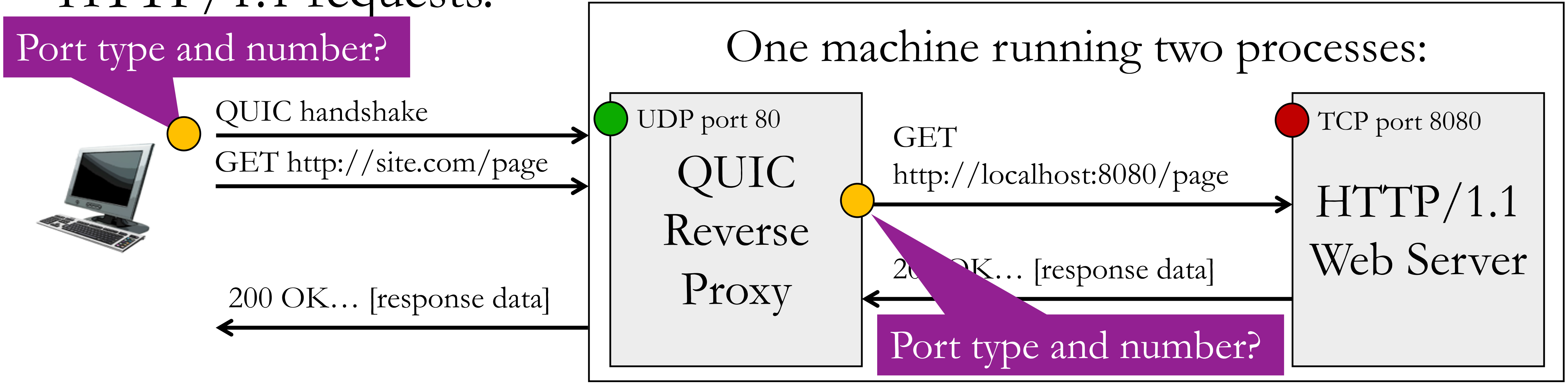
- Mobile radios are often shut off, disconnecting TCP sessions.
- Mobile device may re-join the network with a different IP address.

QUIC uses unique connection IDs:

- When a device moves (eg., from cellular to WiFi), it can resume a connection (eg., streaming a YouTube video) by using the same QUIC connection ID on a different IP address (and UDP port).
- Ie., a connection can continue even if client IP address changes.

Compatibility with HTTP/1.1 and TCP

- Web servers need not change to support QUIC or HTTP/2.
- Can put a reverse-proxy or load-balancer **in front** of basic webserver.
  - This is already commonly done to add TLS to web connections.
  - For example, Nginx supports HTTP/2, unstable release supports QUIC.
- An HTTP-over-QUIC stream can be logically translated into HTTP/1.1 requests.



References

- https://dl.acm.org/citation.cfm?id=3098842
- https://www.zdnet.com/article/http-over-quic-to-be-renamed-http3/
- https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/edit
- https://tools.ietf.org/html/draft-ietf-quic-transport-13
- https://tools.ietf.org/html/draft-ietf-quic-http-13
- https://tools.ietf.org/html/draft-ietf-quic-qpack-01
- https://freecontent.manning.com/animation-http-1-1-vs-http-2-vs-http-2-with-push/

Recap

- The networking stack is layered for historical reasons and for simplicity
- But all abstractions have limitations, including TCP.
- QUIC moves transport to the application layer to solve many modern problems, using modern assumptions about networks:
  - Loading a website requires many HTTP requests.
  - RTTs are dominant factor in network performance.
  - Devices move.
  - Encryption is standard, not optional!

Follow-up courses

- CS-397/497 Wireless Protocols for the Internet of Things
- CS-397/497 Internet-scale Experimentation
- CS-450 Internet Security
- CS-396 Intro to Cryptography
- CS-310 Scalable Software Architectures
- CS-345 Distributed Systems
- COMP_ENG-364 Internet of Things

To learn about the latest networking research:

- Check out articles published at recent SIGCOMM conferences:
  - http://conferences.sigcomm.org/sigcomm/2020/program.html