

Network: Section 4

HTTP/2 & QUIC

Background & Motivation

Reducing web latency - PLT

- User experience
- Scaling of web platform

Insecure -> Secure

- TLS/TCP

“In practice, once the user has more than 5 Mbps of bandwidth, further improvements deliver minimal increase in the loading speed of the average Web application.....”

- streaming HD video from the Web -> bandwidth-bound
- loading the page hosting the HD video, with all of its assets -> latency-bound

Source:

<https://queue.acm.org/detail.cfm?id=2555617>

Recap: HTTP/1.X -> HTTP/2.0

Originally developed by Google

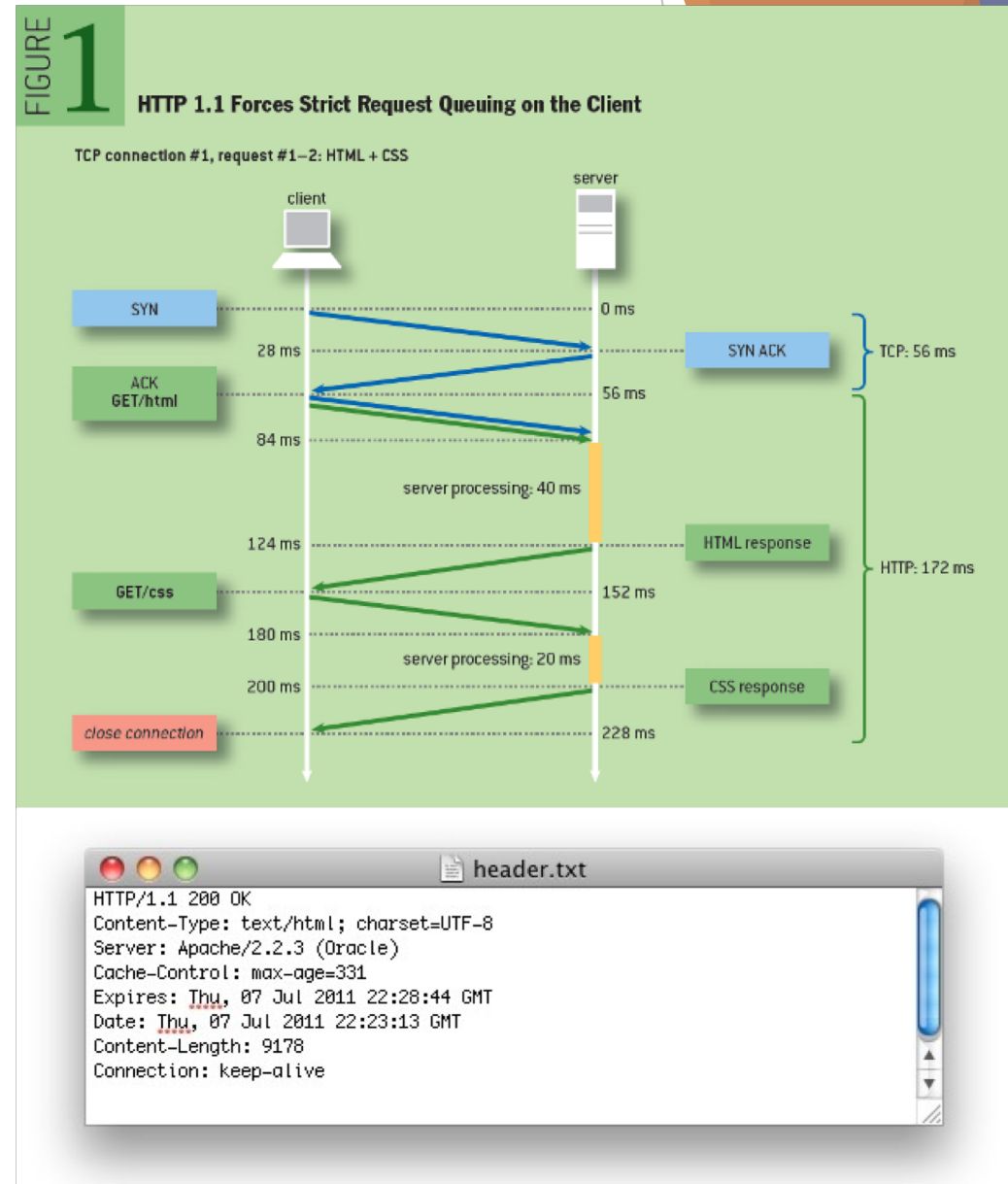
Reference:

<https://queue.acm.org/detail.cfm?id=2555617>

<https://developers.google.com/web/fundamentals/performance/http2/>

HTTP/1.1

- ▶ Reuse TCP connection
 - ▶ HTTP/1.0 - one TCP per resource -> overhead
 - ▶ HTTP/1.1 - up to six TCP per origin
- ▶ Request pipeline
 - ▶ Theoretically, yes; **but failed.**
 - ▶ Community homebrew “optimizations”
 - ▶ Multiple origins -> more parallelism
 - ▶ Bundle files -> less requests
 - ▶ One giant CSS file
 - ▶ One giant JS file
 - ▶ Code everything directly into HTML
 - ▶ Leads to network congestion + poor modularity (caching & page loading)
- ▶ HTTP Header
 - ▶ plain text - each char is 1 byte
 - ▶ newline-delimited

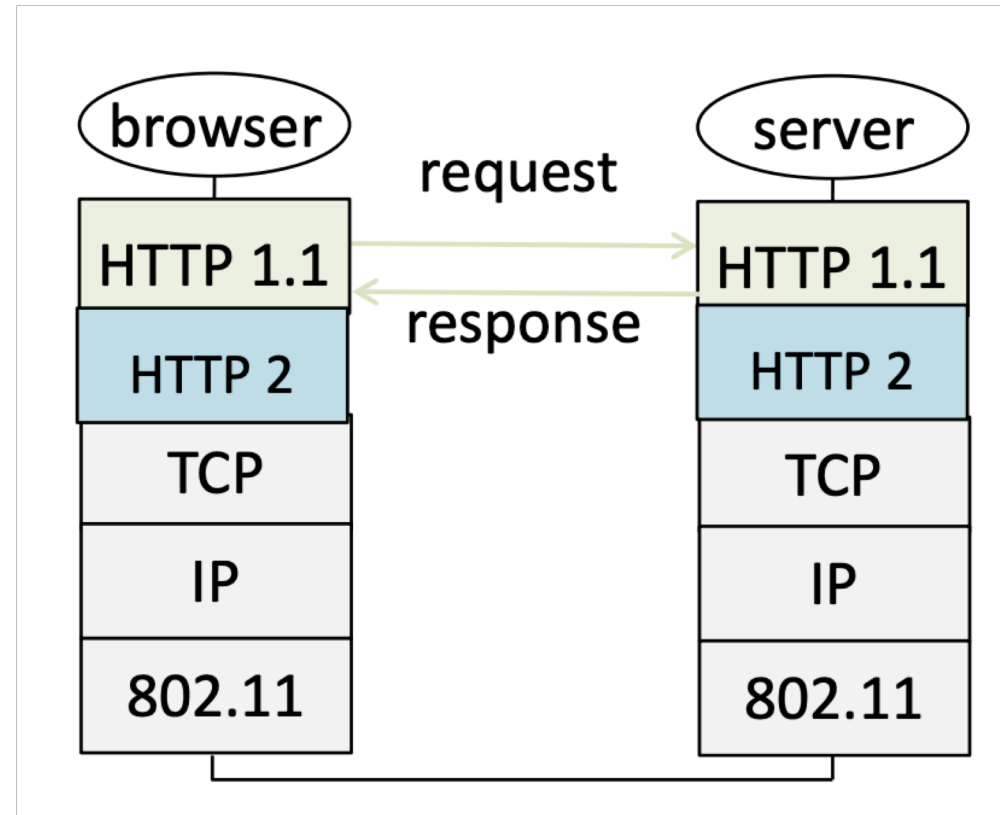


“All problems in computer science can be solved by another level of indirection...”

--- David Wheeler

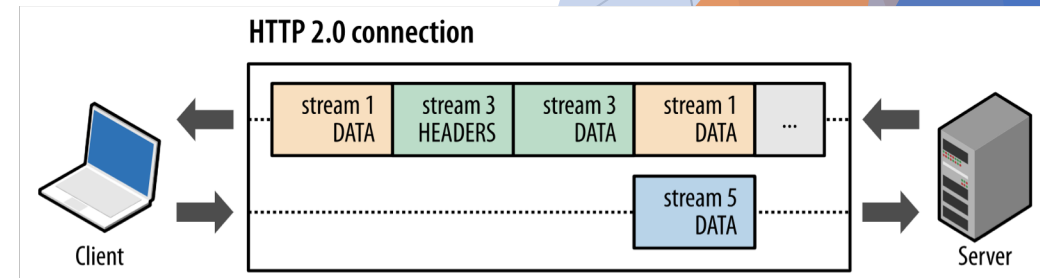
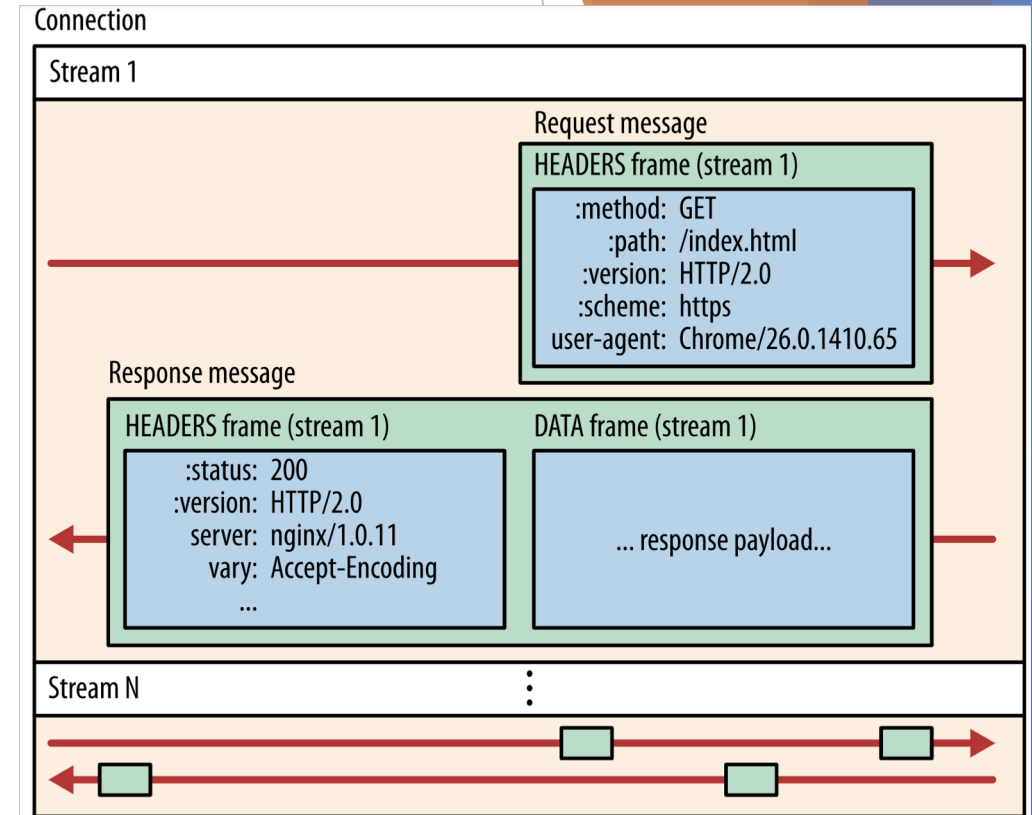
HTTP/2.0

- ▶ Another layer of abstraction over HTTP/1.1
- ▶ Multiplexing
- ▶ Prioritization
- ▶ Header compression
- ▶ Server push



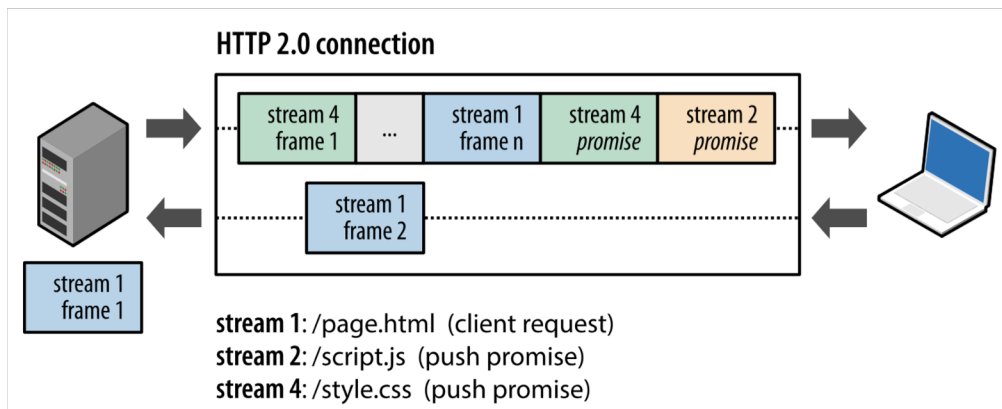
HTTP/2.0: Multiplexing, Prioritization

- Streams inside one TCP connection
 - HTTP message -> HEADER frame + DATA frame
 - One stream for one HTTP request + response
 - Multiple streams inside one TCP connection
 - streams can have different priorities
 - Frames from different streams may be interleaved and then reassembled via the embedded stream identifier in the header of each frame.



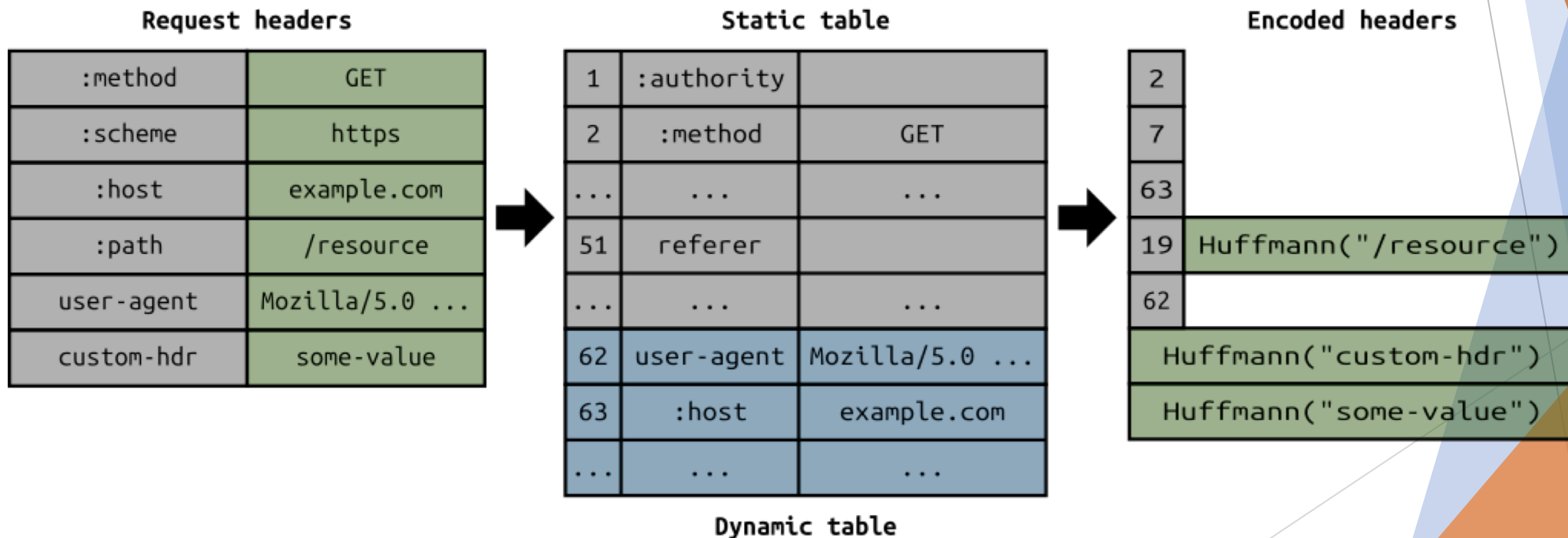
HTTP/2.0: Server push

- ▶ Push the resource to the client instead of waiting for the client to request it
 - ▶ Request html
 - ▶ Push CSS, JS, IMG...
- ▶ “PUSH_PROMISE”
 - ▶ Contains HTTP request header of the pushed elements
 - ▶ Server initiates a new stream and start pushing
 - ▶ Client can reject by sending RST_STREAM



HTTP/2.0: HTTP Header Compression

- ▶ HTTP headers are plain texts with a lot of repetitions (“HTTP/1.1”, “GET”, ...)
- ▶ HPACK compression algorithm: static table + dynamic table + static Huffman code

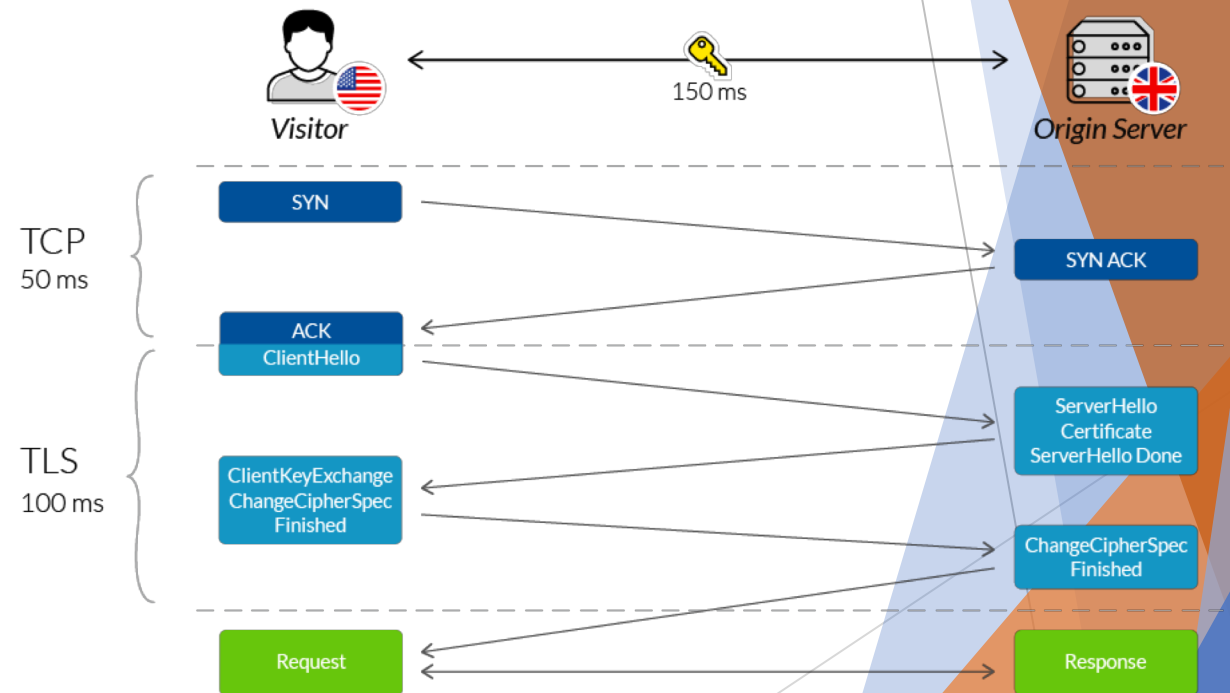


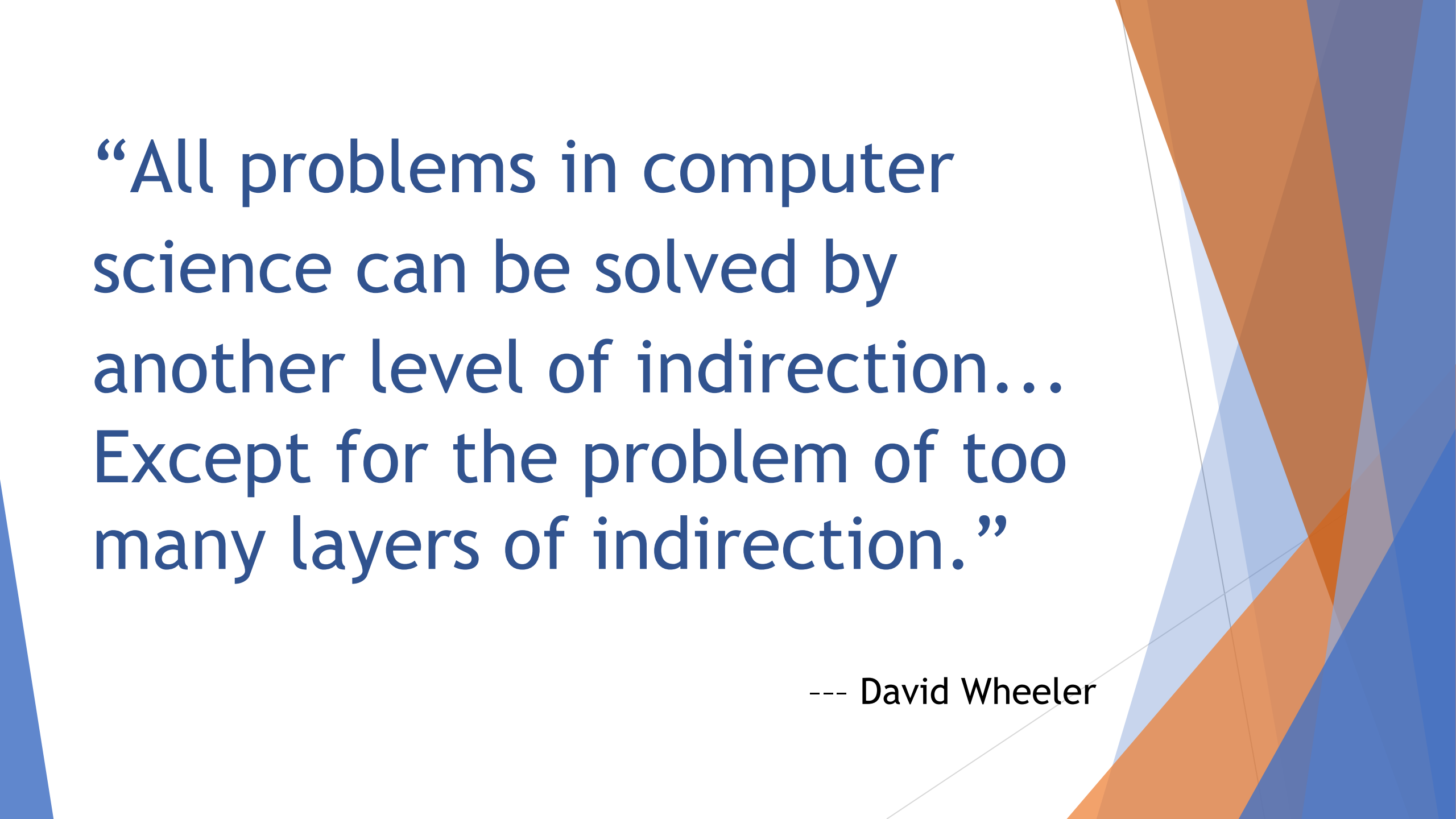
QUIC: Quick UDP Internet Connections

(Again) By Google

Problem with TLS/TCP

- ▶ TCP headers unencrypted
 - ▶ Middleboxes - Firewall, NAT
- ▶ TCP commonly implemented in OS kernel
 - ▶ Update really slow
 - ▶ Sizeable user populations lag behind
- ▶ Handshake Delay
 - ▶ 1 RTT for TCP + 2 RTT for TLS
 - ▶ c is constant
- ▶ Head-of-line blocking delay
 - ▶ One TCP stream in HTTP/2
 - ▶ Packet lost? Wait...





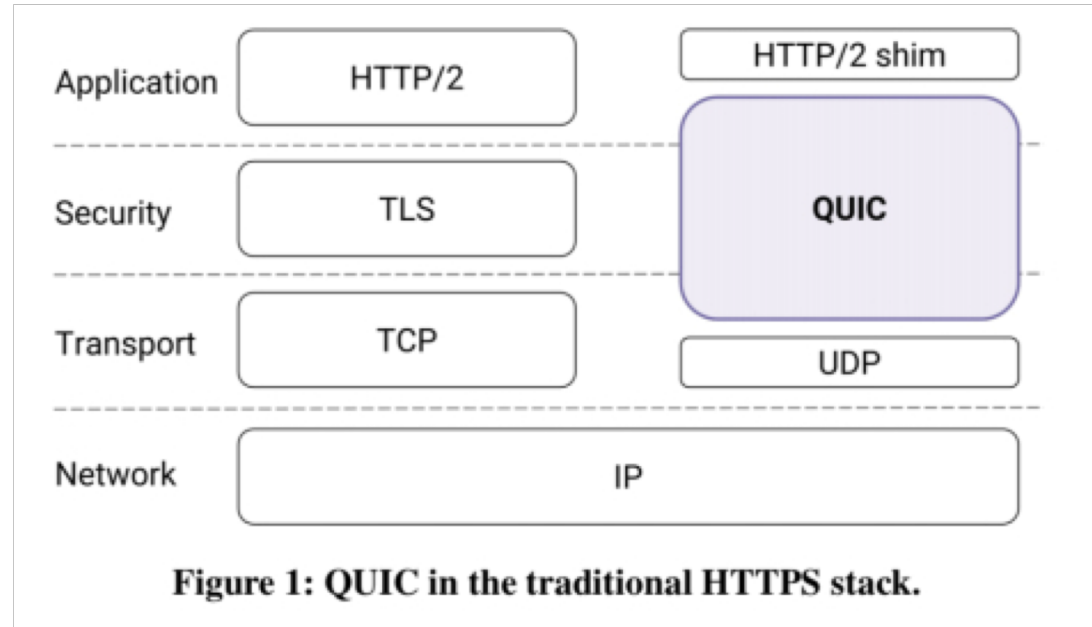
“All problems in computer science can be solved by another level of indirection... Except for the problem of too many layers of indirection.”

--- David Wheeler

QUIC

Introduction

- ▶ TCP+TLS+HTTP2
- ▶ Application Layer
- ▶ HTTPS Performance+
- ▶ E2E Encrypted
- ▶ Secure
- ▶ Rapid Deployment



QUIC

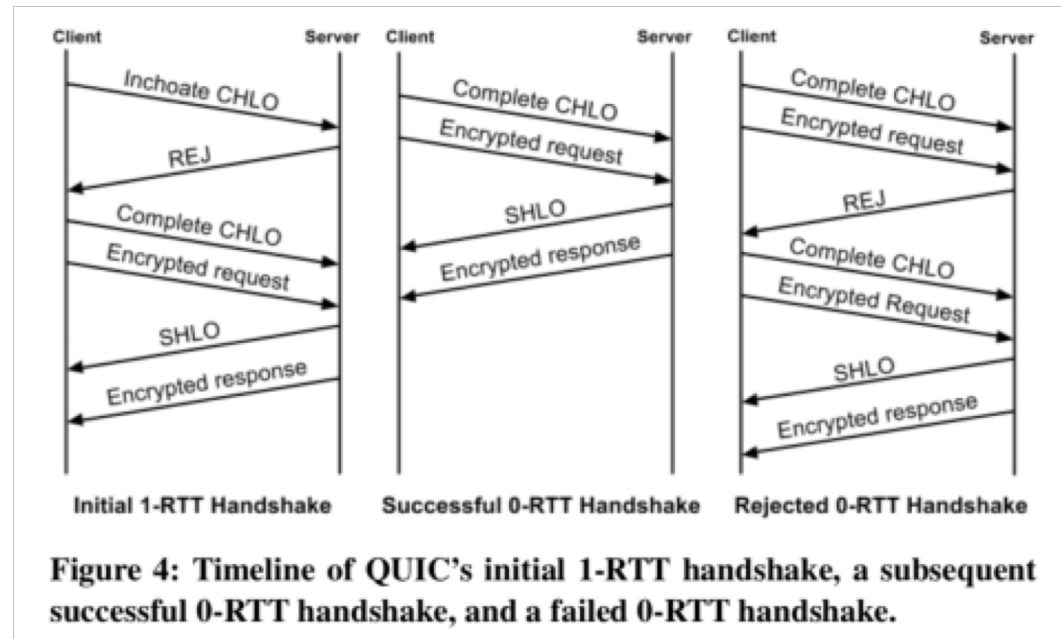
Key Advantages

- ▶ Connection establishment latency
- ▶ Improved congestion control
- ▶ Multiplexing without head-of-line blocking
- ▶ Forward error correction
- ▶ Connection migration

QUIC Advantages #1

Connection Establishment Latency

- ▶ Combined handshake
 - ▶ Inchoate and complete CHLO
 - ▶ SHLO
 - ▶ 1-RTT
- ▶ Client cache long-term Diffie-Hellman public key
 - ▶ 0-RTT



QUIC Advantages #2

Congestion Control

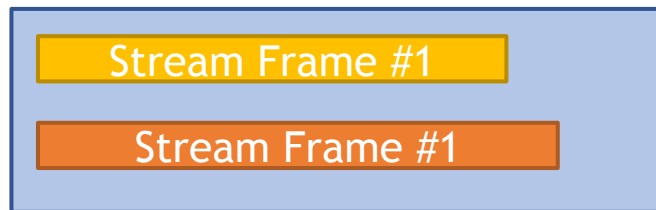
- ▶ Pluggable Interface Design
 - ▶ Easy to switch
 - ▶ Easy to experiment
 - ▶ Easy to update
- ▶ ACK carries more messages
 - ▶ More information
- ▶ Better use of packet number

QUIC Advantages #3

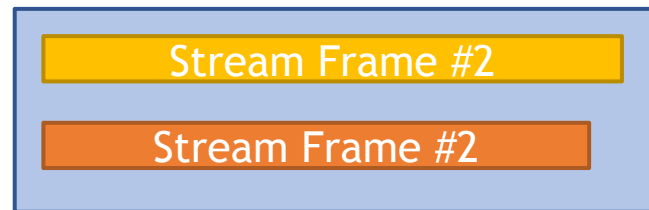
Multiplexing

- ▶ Solve Head-of-line blocking delay
 - ▶ Stream - lightweight TCP connection without handshakes
 - ▶ Multiple streams in one connection
 - ▶ One QUIC packets can carries multiple stream frames

One QUIC Connection



QUIC Packet #1
(UDP Packet)



QUIC Packet #2
(UDP Packet)

QUIC Advantages #4

Forward error correction

- ▶ Skip
- ▶ Describe in Sec 7.3 in paper
- ▶ Benefits not compelling
- ▶ Removed from QUIC in early 2016

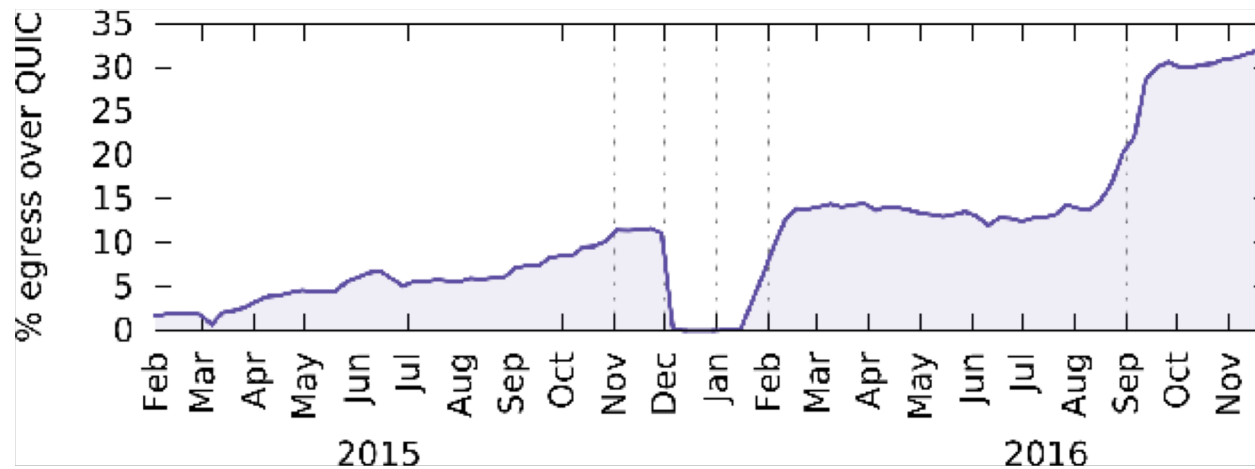
QUIC Advantages #5

Connection Migration

- ▶ TCP
 - ▶ Src IP:Port + Dest IP:Port + protocol - 5-tuple Identification
 - ▶ Client change IP, NAT change port... -> connection break
- ▶ QUIC
 - ▶ Connection ID
 - ▶ Connection remains even network environment changes

QUIC Results #1

- ▶ Well tested and deployed widely
 - ▶ Chrome, YouTube, Google Search App, ...
 - ▶ 7% of the internet traffic



QUIC Results #2

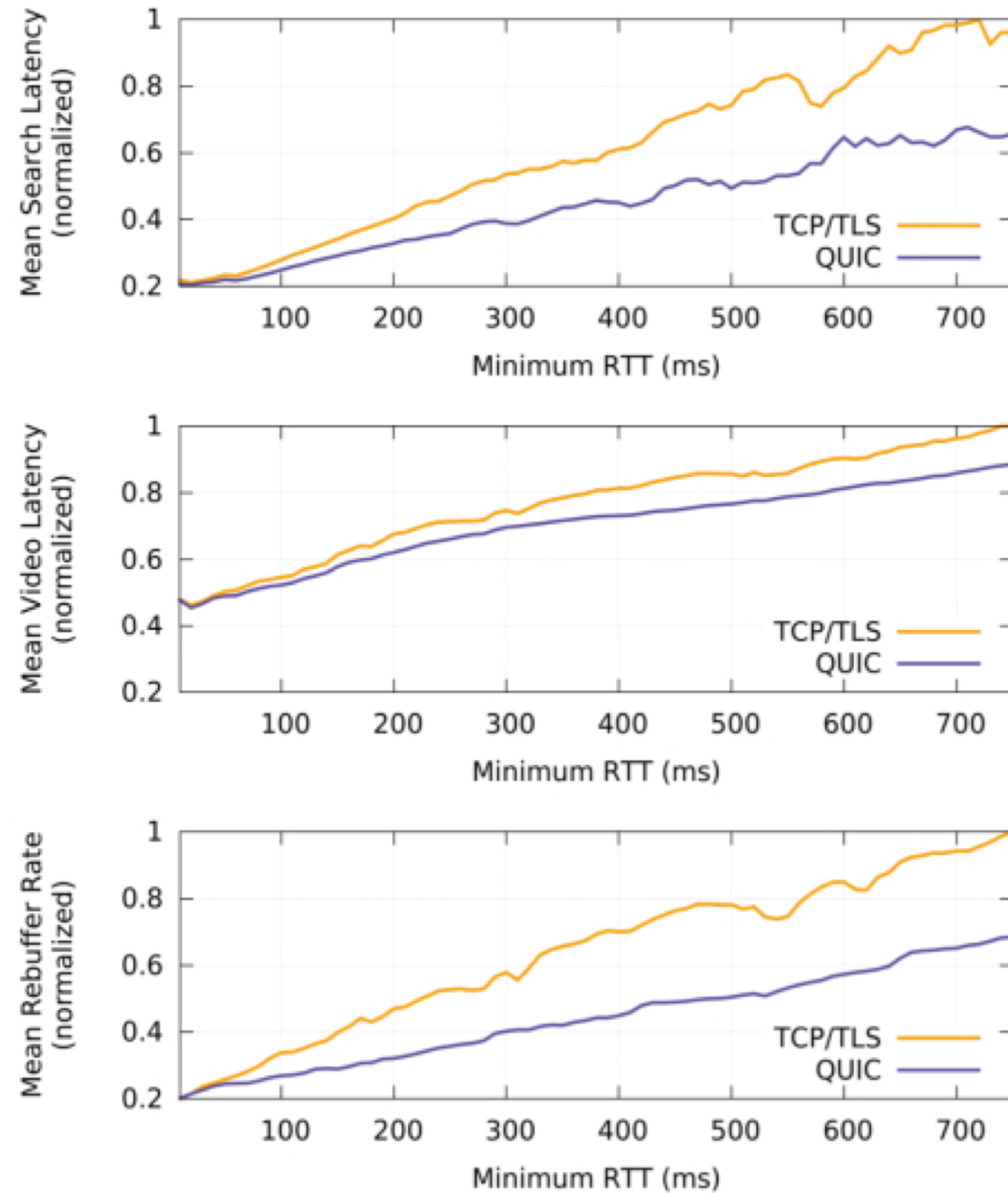


Figure 9: Comparison of QUIC_g and TCP_g for various metrics, versus

QUIC Future Work

- ▶ Alternative congestion control algorithm
- ▶ Reduce CPU cost
 - ▶ Twice of the TCP
- ▶ Improve performance on mobile devices
 - ▶ Mobile app -> invisible handshake, compressed content...
 - ▶ CPU bottleneck
- ▶ MTU discover for QUIC
 - ▶ MTU: maximum packet size
 - ▶ MTU now sets to a fixed tested value: 1450 bytes

QUIC References

- ▶ <https://tools.ietf.org/html/draft-ietf-quic-transport-09>
- ▶ <https://dl.acm.org/citation.cfm?id=3098842>
- ▶ <https://www.chromium.org/quic>
- ▶ <https://docs.google.com/document/d/1gY9-YNDNAB1eip-RTPbqphgySwSNSDHLq9D5Bty4FSU/edit>
- ▶ <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/8b935debf13bd176a08326738f5f88ad115a071e.pdf>