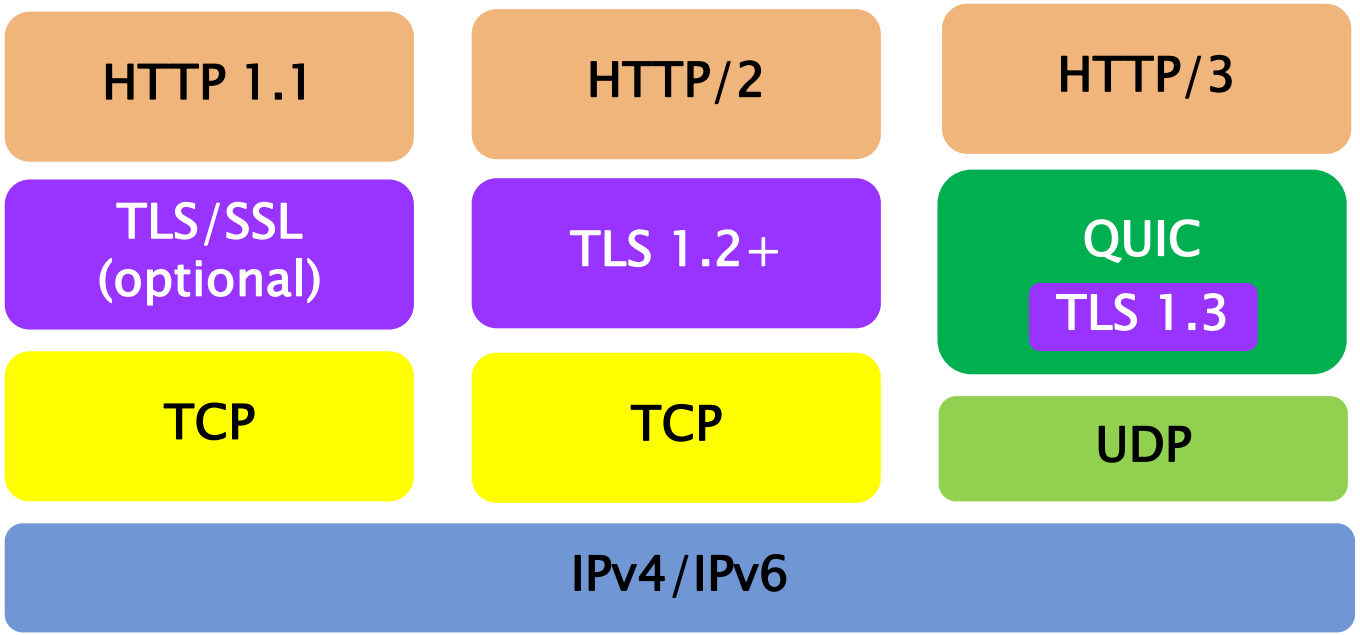
Computer Networks Advanced Course

QUIC / HTTP3

Barak Gonen

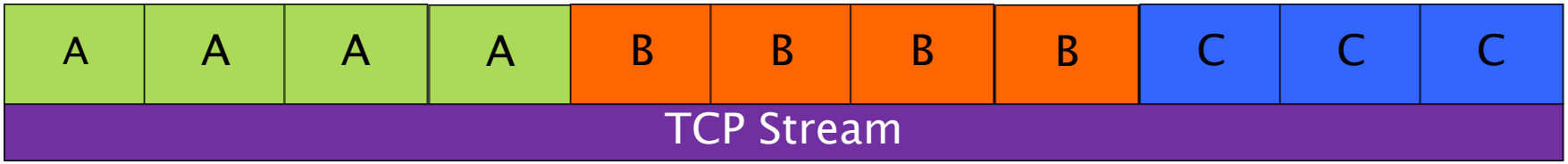
Contents

- HTTP versions: problems and solutions
- HTTP/3 over QUIC
- Wireshark hands on
  - Client initial packet
  - Server initial packet



HTTP/1.1

- A serial protocol
- One TCP session for multiple resources

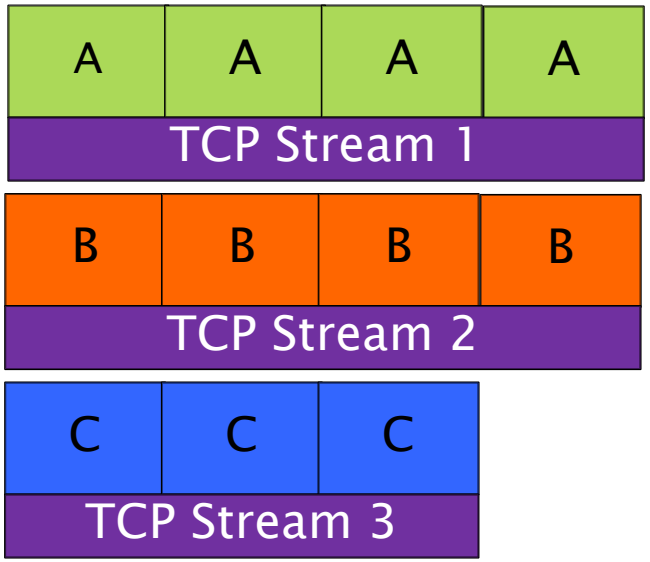


Server's response, assume client requested resources A, B, C

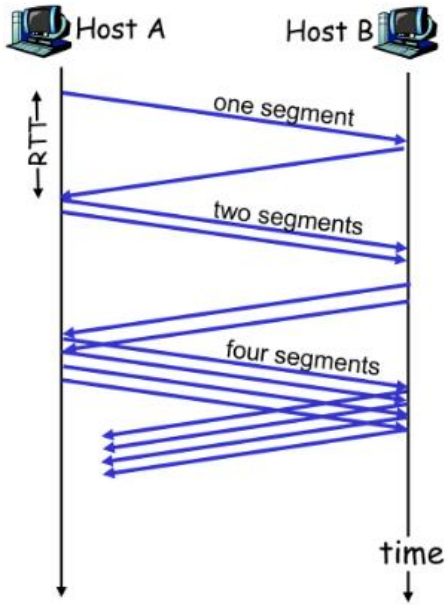
- Actually, there are "dead" times:
  - Server takes time to fetch data
  - Client takes time to process response

HTTP/1.1

- How can it be sped up?
- Open multiple TCP sessions
  - Not optimal
  - TCP has initial “payment” and ramp-up
- TCP is suitable for long transmissions



TCP Slow Start
(Kuros)



HTTP/1.1 GET

GET /index.html

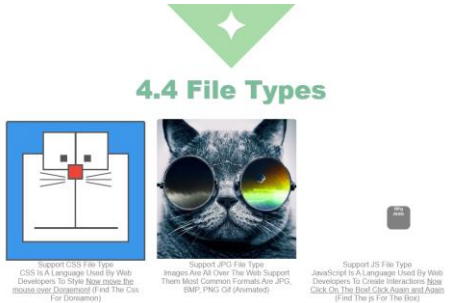
index.html 200 OK

GET /abstract.jpg

GET /doremon.css

GET /box.js

GET /favicon.ico



index.html

abstract.jpg

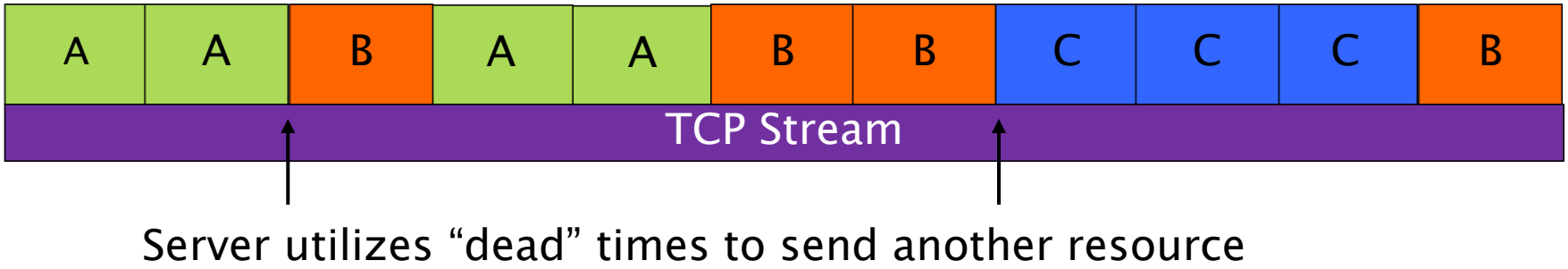
doremon.css

box.js

favicon.ico

HTTP/2

- Using a single TCP session
- Each resource is given a unique stream ID
- Hands on – filter http2, find “stream ID”



HTTP/2 Server Push

GET /index.html

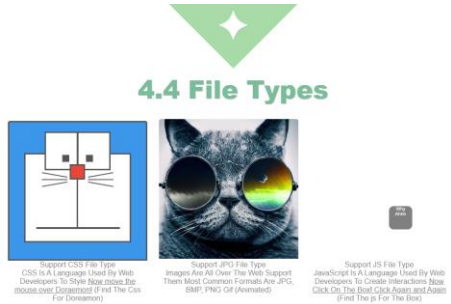
index.html 200 OK

abstract.jpg 200 OK

doremon.css 200 OK

box.js 200 OK

favicon.ico 200 OK



index.html

abstract.jpg

doremon.css

box.js

favicon.ico

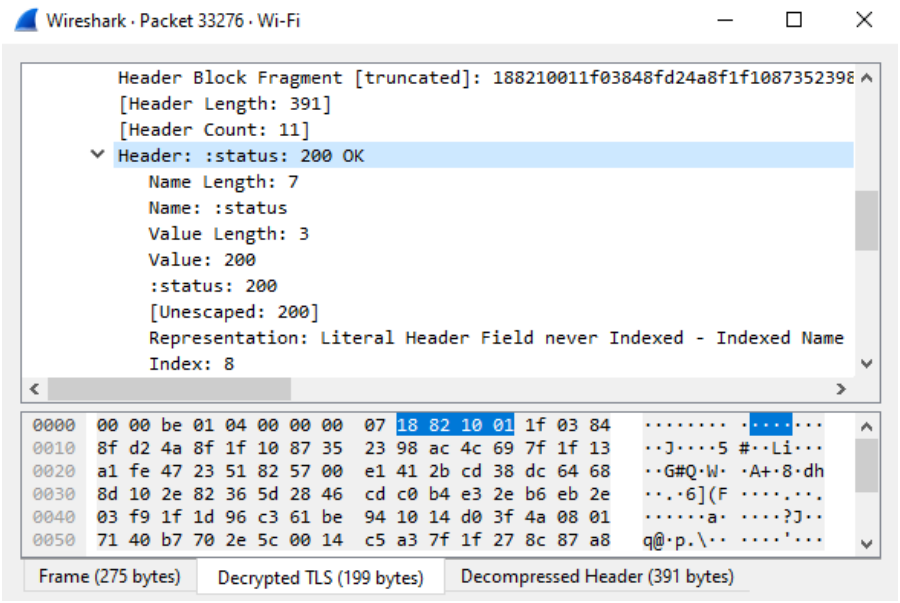
HTTP/2 HPACK

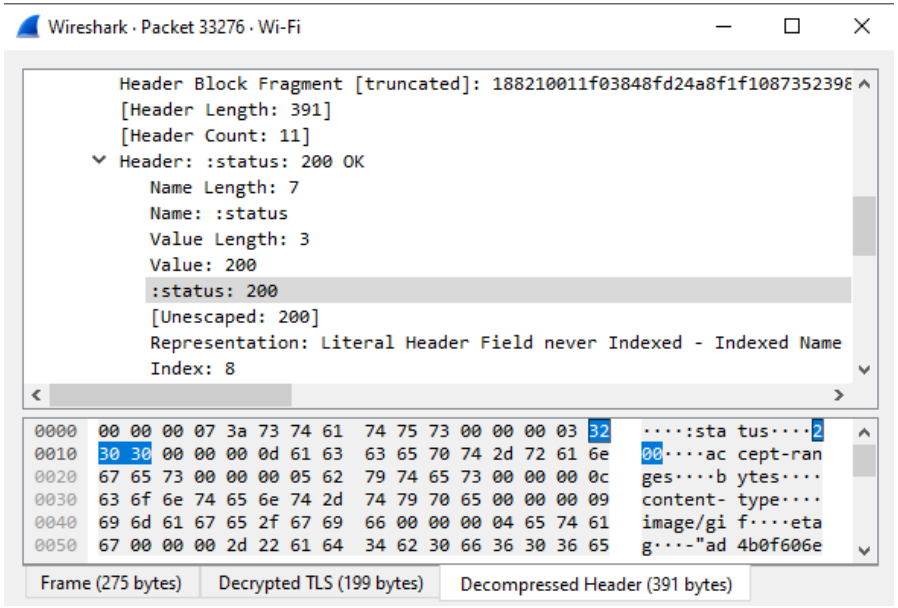
- Compress HTTP headers to reduce size
  - Static table for common values

Index	Header name	Header value
1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
5	:path	/index.html

  - Dynamic table, Huffman algorithm

HTTP/2 HPACK





"status 200" and length fields compressed to only 4 bytes

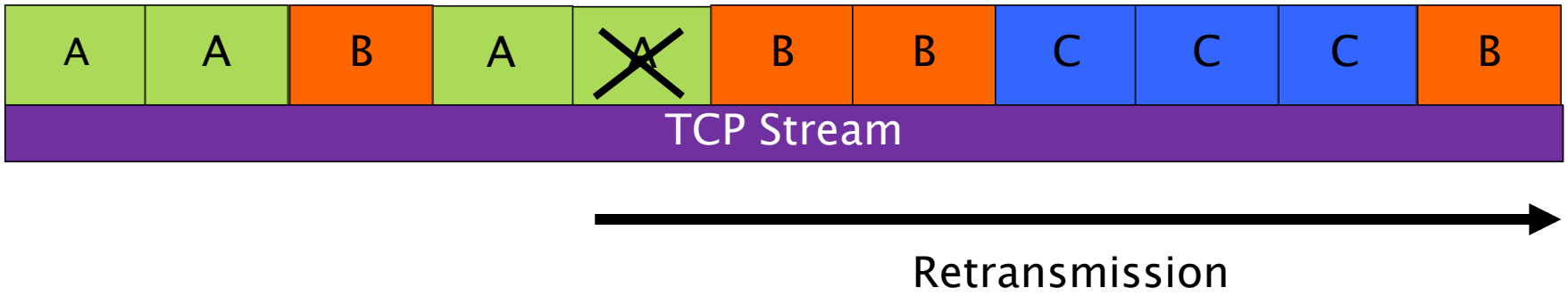
Decompressed header

HTTP3 Main Motivations

- Bypass HTTP2 Head of line blocking issue
- Bypass TCP “Ossification” issue
- Reduce RTT to first byte of data
- Support connection migration

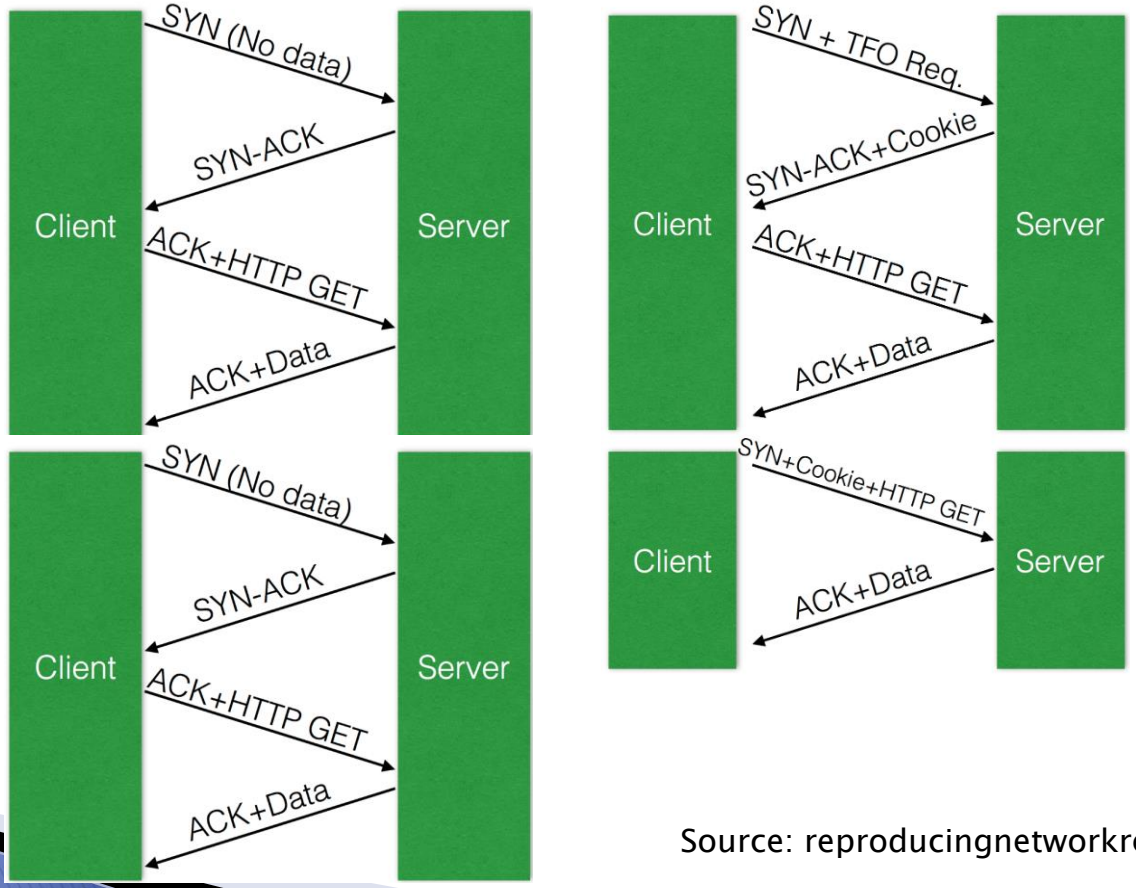
HTTP/2 – Head of Line Blocking

- Assume lost packet – last packet of "A"
- Will resources B, C keep downloading?



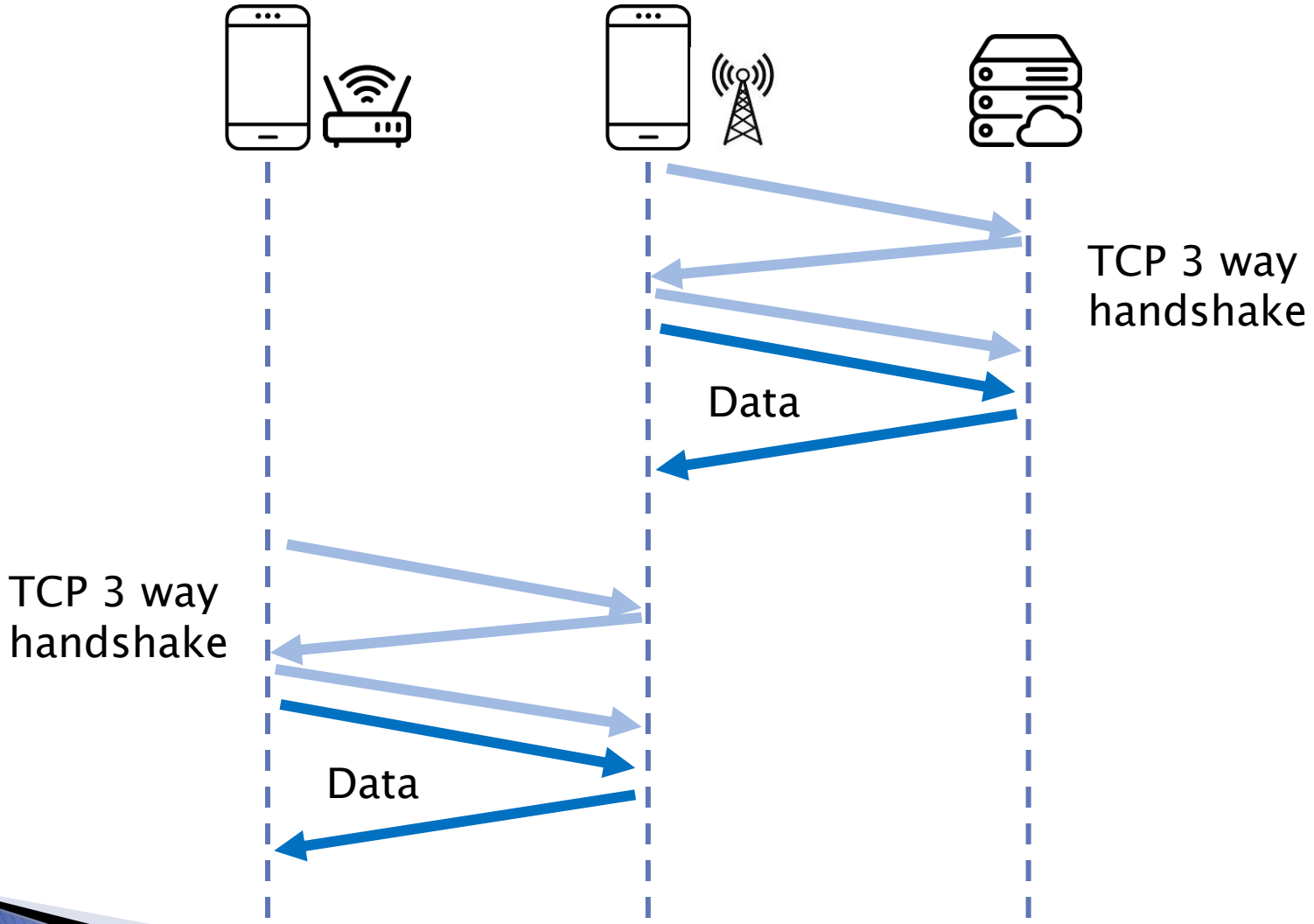
TCP "Ossification"

- Due to middleboxes, improvements become not practical
- Example – TCP fast open 2009
- Fails passing Deep Packet Inspection (DPI)



Source: reproducingnetworkresearch

TCP has no Connection Migration



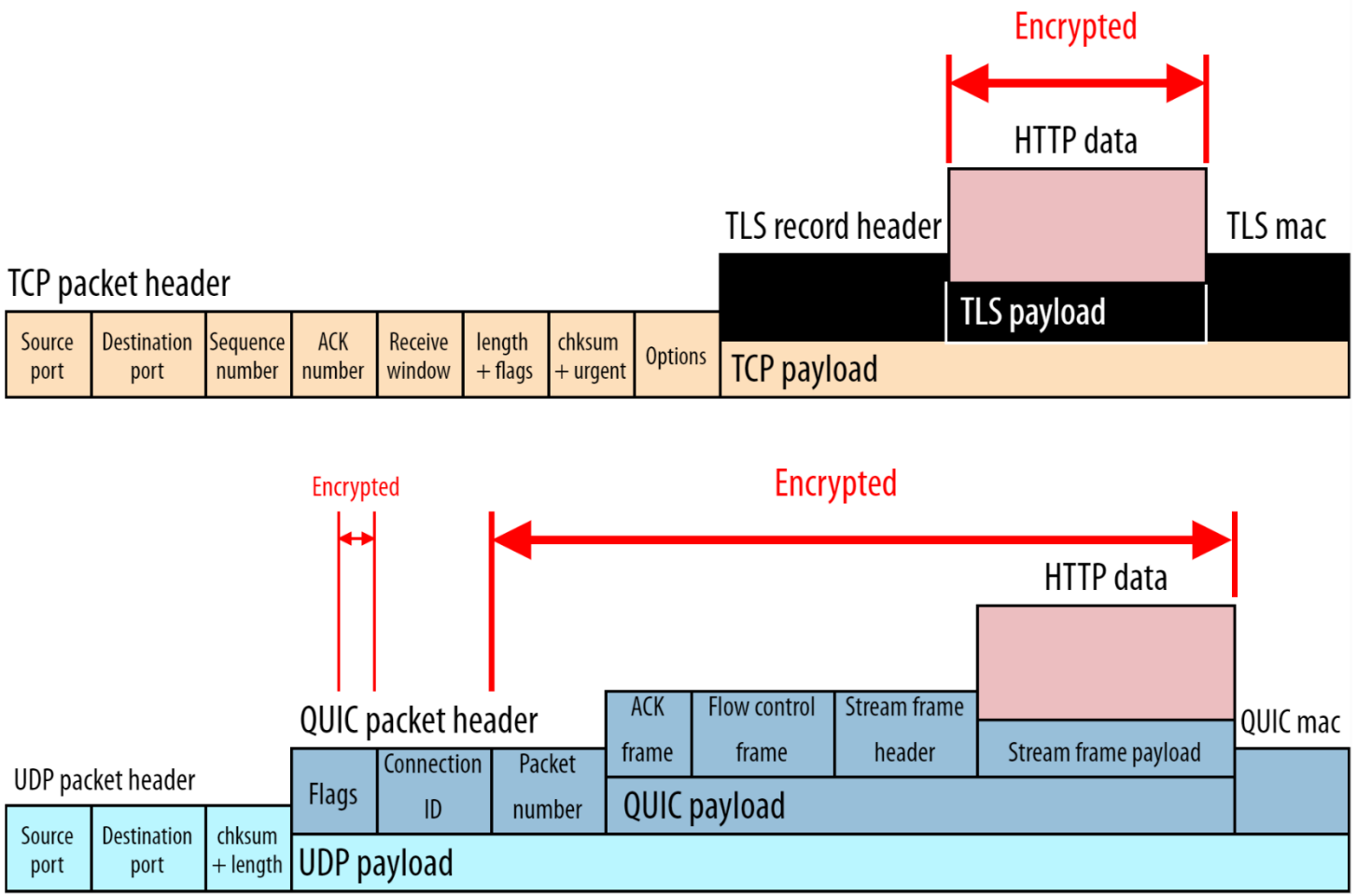
QUIC

- Stands for: Quick UDP Internet Connections
- UDP Port 443
- QUIC history:
  - Google QUIC – 2012
  - 2016 – submitted to IETF – Internet Engineering Task Force
  - May 2021 – standardized

QUIC Key Principles

- Handles classic TCP responsibilities:
  - Set up connection – handshake
  - Reliability – ACK's, retransmissions
  - Flow control
  - Congestion control
  - *"Readers, familiar with TCP's loss detection and congestion control will find algorithms here that parallel well-known TCP ones." [from QUIC specification]*
- TLS 1.3 embedded
  - Some fields are encrypted, no QUIC without TLS 1.3

QUIC embedding TLS 1.3



Source: https://www.smashingmagazine.com/2021/08/http3-core-concepts-part1/

Hands On

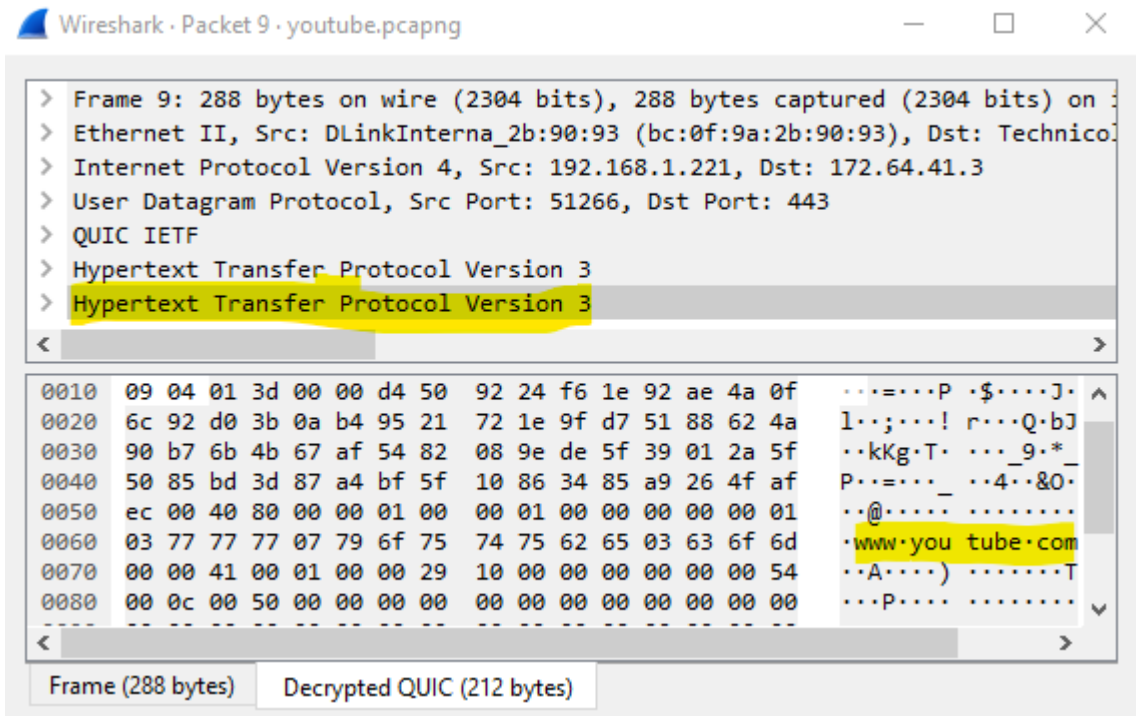
- quic_sniff.pcapng
- sslkeylogfileQUIC.txt

Hands On – Secure DNS

- Packet 6 – use secure DNS
  - Which DNS server is contacted?
  - Look for “Extension: server name”
- Follow UDP stream– packet 9
  - Which application layer protocol is used?
  - The DNS query is for which domain?
  - Use “Decrypted QUIC” tab

Answers

```
Server Name Indication extension
    Server Name list length: 28
    Server Name Type: host_name (0)
    Server Name length: 25
    Server Name: chrome.cloudflare-dns.com
```



QUIC- Switch from TCP

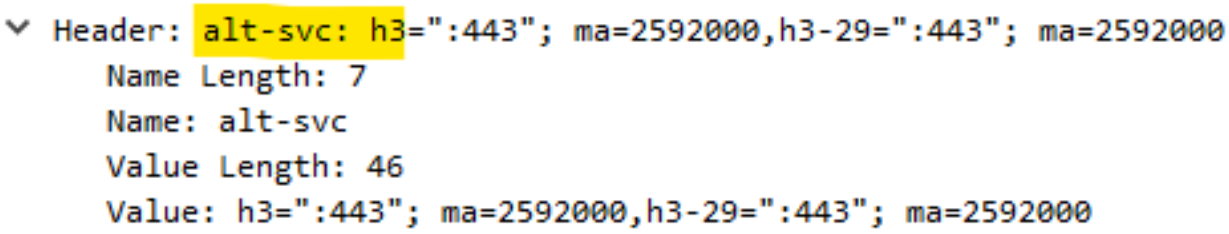
- Within TLS, field allowing you to announce QUIC support.
  - Conversation may start with HTTP2/TCP and switch to HTTP3/QUIC
  - The server will propose “alt-svc”
  - The client may connect with QUIC next time
- If you do not connect for the first time, you already apply directly in QUIC

Hands On – Switch from TCP

- Find the first TCP SYN and follow stream
- Which alternative service is proposed by the server?
  - Find the “alt-svc” in the first HEADERS packet sent by the server

Answers

- Packet 22 – TCP SYN to youtube (google)
- Packet 48 – “alt-svc h3”



Hands On – Client Hello, Server Hello

- Find the client hello to youtube
- Use following filter:
  - tls.handshake.extensions_server_name == "www.youtube.com"
- Use “follow UDP stream” to find server hello

Client Initial Packet – #70

- TLS 1.3 Client Hello
- ALPN – Application Layer Protocol Negotiation – HTTP
- Packet number 0
- Extension – QUIC params
  - Flow control – the server sets boundaries to the client
    - How many streams
    - Limit per streams
    - Limit of overall data
- Connection ID

Server Initial Packet – #87

- Server starts with the CID the client chose
  - Note that besides the 1st digit, it's the same
- ACK – Largest acknowledge 1 (client initial packet)
- TLS v1.3, certificate
- Extension QUIC transport parameters – packet 350

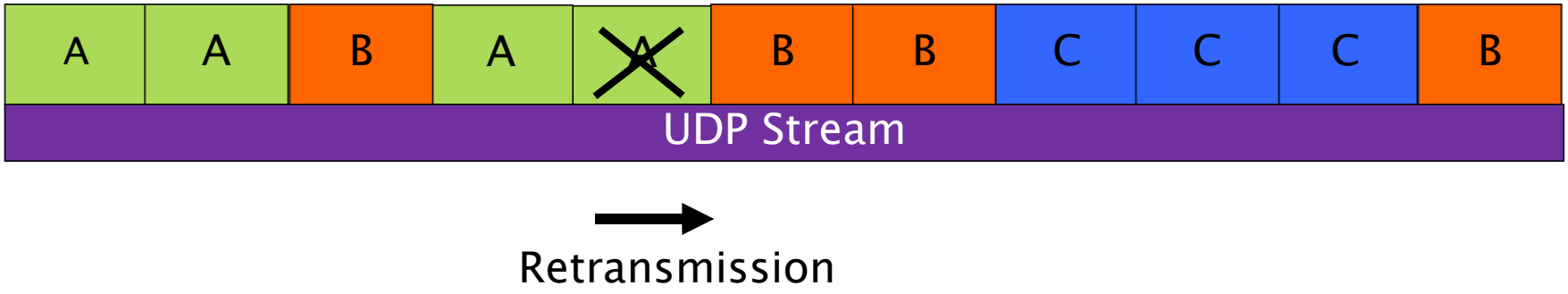
HTTP3 Main Motivations

Recall these issues

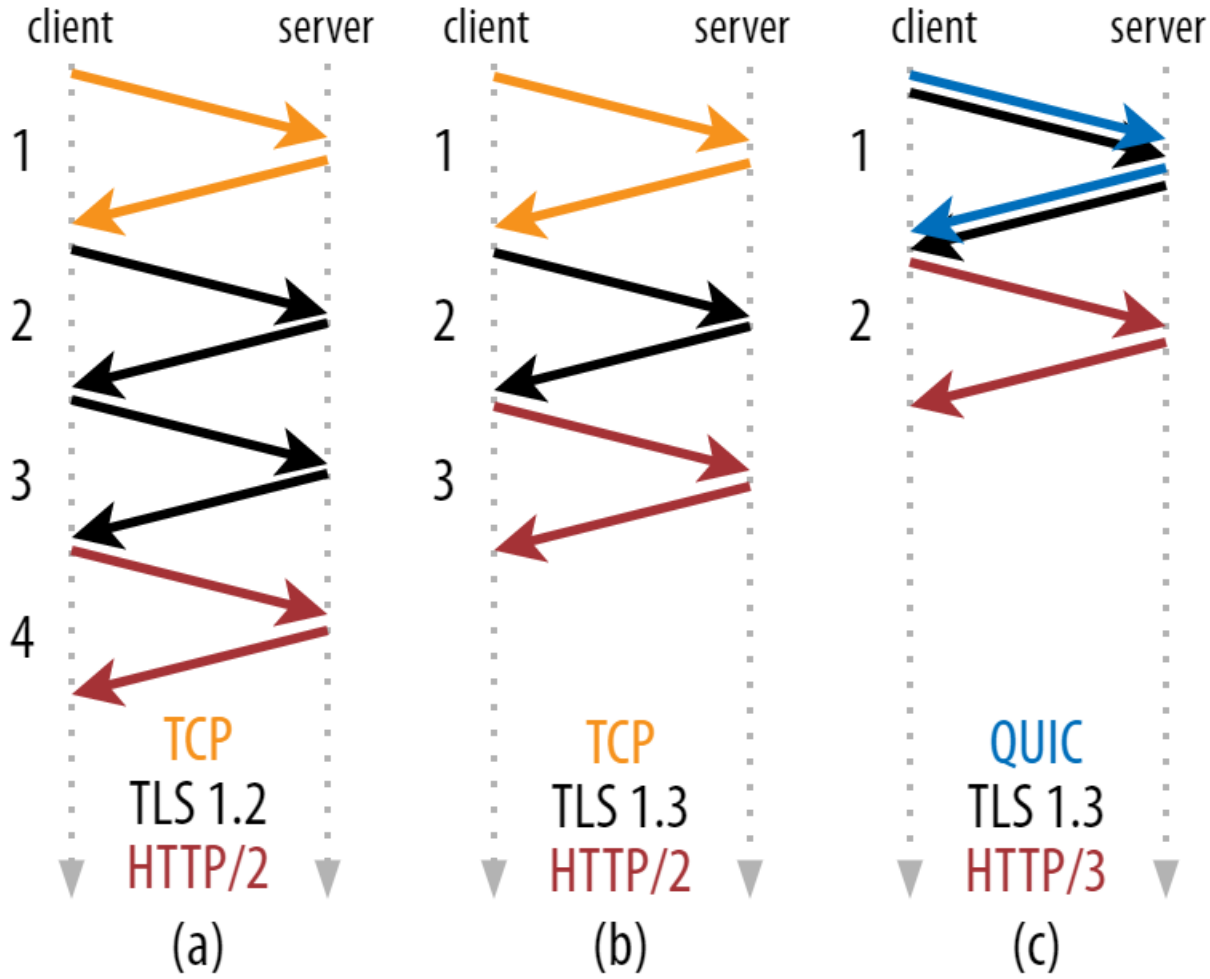
- Bypass HTTP2 Head of line blocking issue
- Bypass TCP “Ossification” issue
- Reduce RTT to first byte of data
- Support connection migration

HTTP/3 – No Head of Line Blocking

- Assume lost packet – last packet of "A"
- Which stream will pause?
- Which packets will be retransmitted?

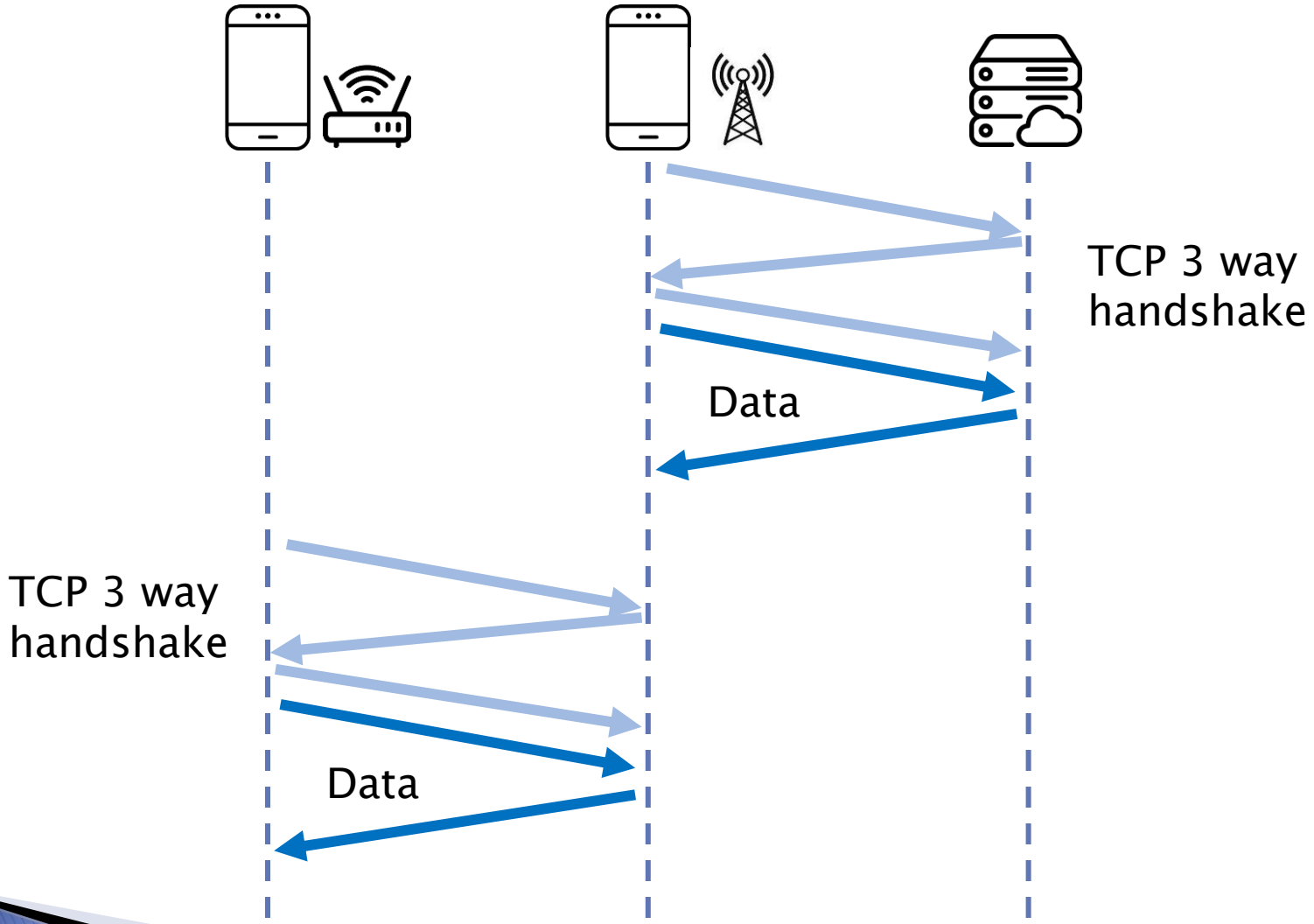


QUIC RTT

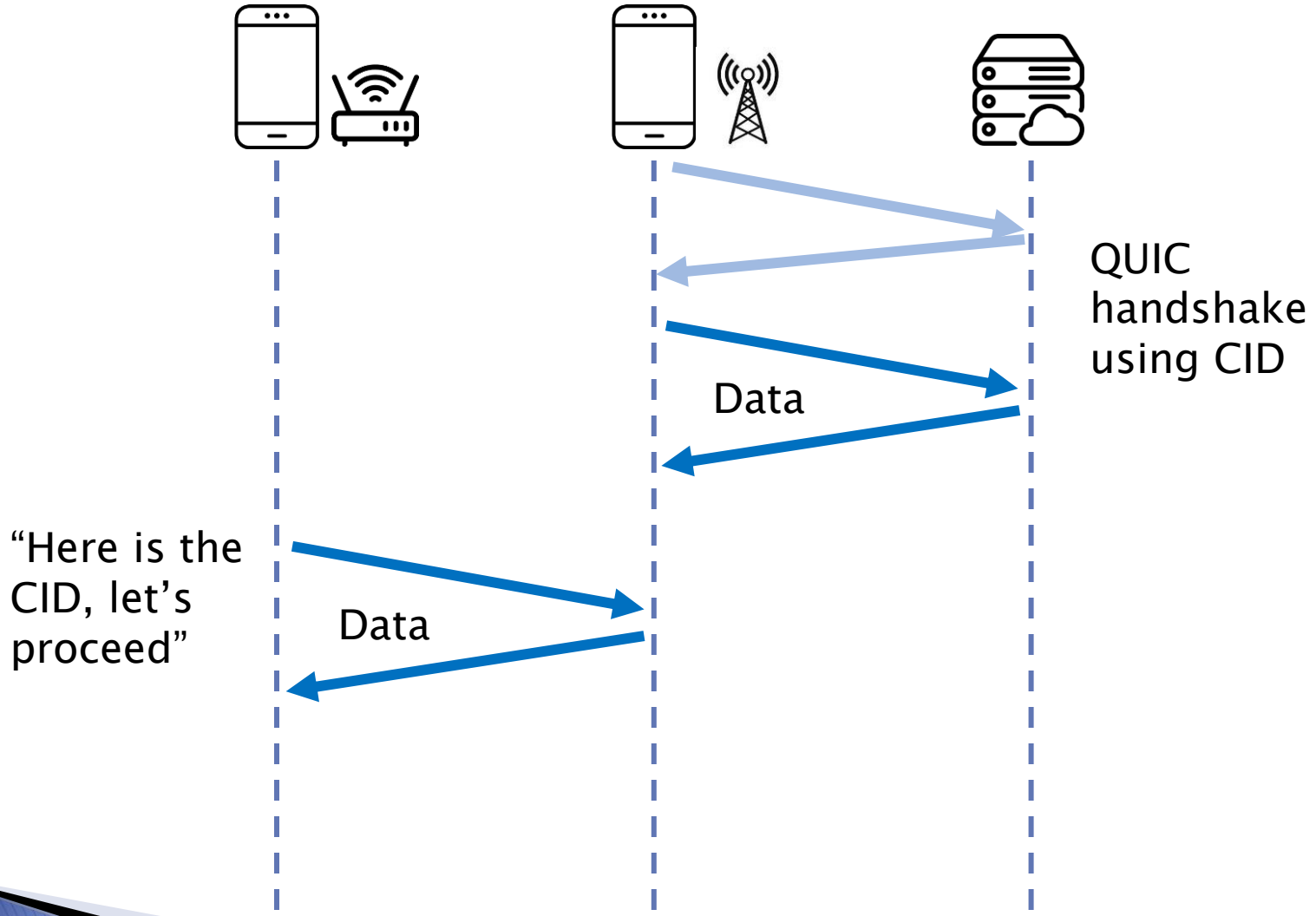


Source: https://www.smashingmagazine.com/2021/08/http3-core-concepts-part1/

TCP has no Connection Migration



QUIC Connection Migration



HTTP3 Main Motivations

- Bypass HTTP2 Head of line blocking issue 
- Bypass TCP “Ossification” issue 
- Reduce RTT to first byte of data 
- Support connection migration