



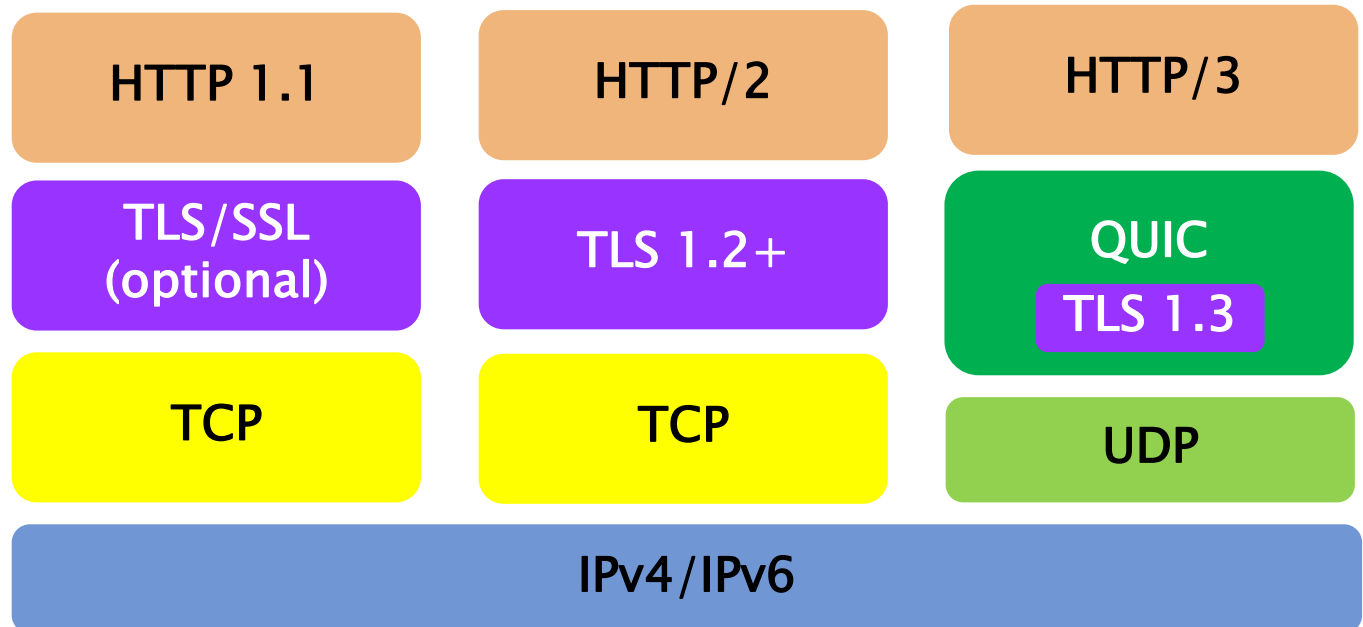
Computer Networks Advanced Course

QUIC / HTTP3

Barak Gonen

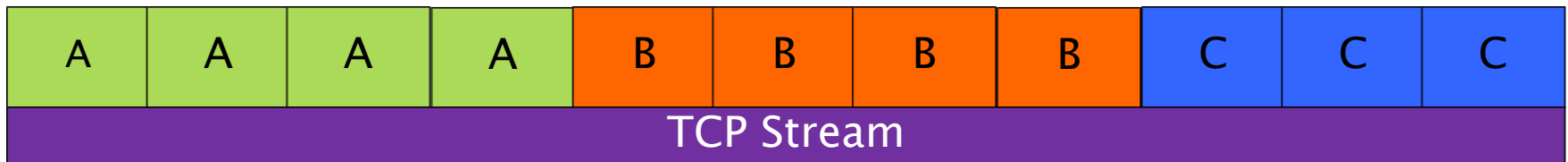
Contents

- ▶ 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

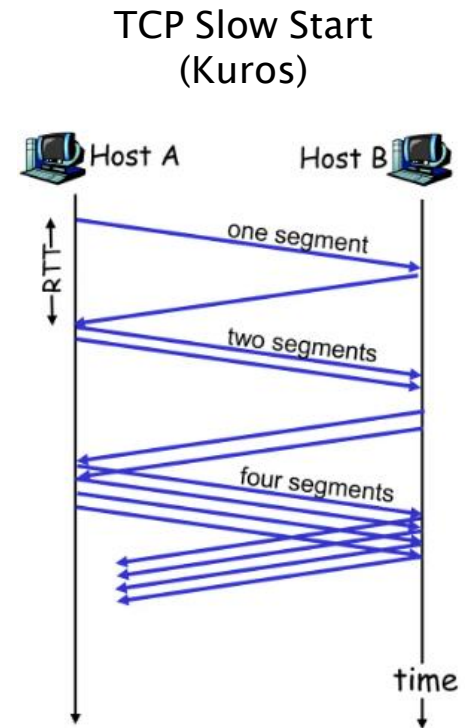
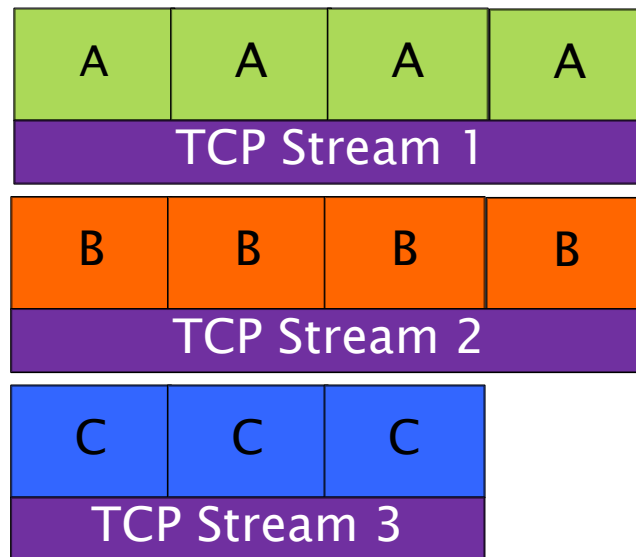


Server's response, assume client
requested resources A, B, C

- ▶ 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



HTTP/1.1 GET



GET /index.html



index.html 200 OK



GET /abstract.jpg



GET /doremon.css



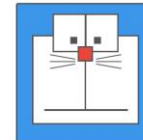
GET /box.js



GET /favicon.ico



4.4 File Types



Support CSS File Type
CSS is a Language Used by Web
Developers To Style Pages (css).
css (source code) (download) (view the code
for download)



Support JPG File Type
Images Are All Over The Web Support
Them Most Common Formats Are JPG,
BMP, PNG GIF (animated)



Support JS File Type
JavaScript is a Language Used By Web
Developers To Create Interactions (js).
Click On The Best Click Again and Again
(find The js For The Best)

index.html

abstract.jpg

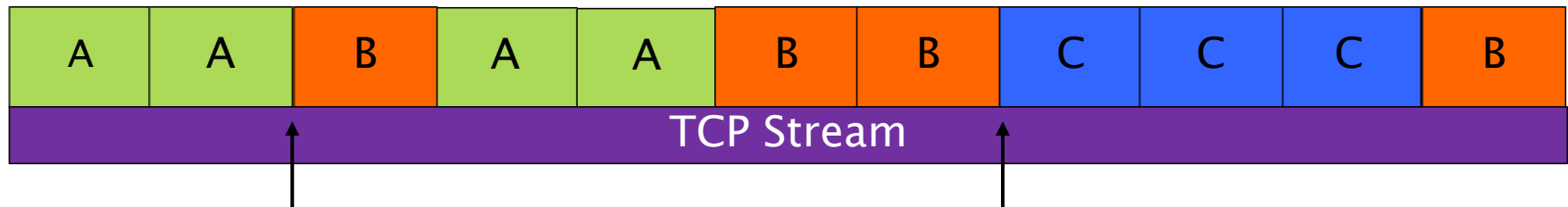
doremon.css

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”



Server utilizes “dead” times to send another resource

HTTP/2 Server Push



GET /index.html



index.html 200 OK



abstract.jpg 200 OK



doremon.css 200 OK



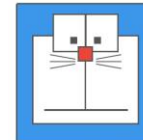
box.js 200 OK



favicon.ico 200 OK



4.4 File Types



Support CSS File Type
CSS is a Language Used by Web
Developers To Style Web Pages.
Browsers (User Agents) Support It and The CSS
For Document



Support JPEG File Type
Images Are All Over The Web Support
Them Most Common Formats Are JPEG,
PNG, GIF (Animated)



Support JS File Type
JavaScript is a Language Used By Web
Developers To Create Interactions Like
Click On The Button Click Again and Again
If not The js For The Best

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

Wireshark · Packet 33276 · Wi-Fi

Header Block Fragment [truncated]: 188210011f03848fd24a8f1f108735239e ^
[Header Length: 391]
[Header Count: 11]

▼ Header: :status: 200 OK
Name Length: 7
Name: :status
Value Length: 3
Value: 200
:status: 200
[Unescaped: 200]
Representation: Literal Header Field never Indexed - Indexed Name
Index: 8

0000 00 00 be 01 04 00 00 00 07 18 82 10 01 1f 03 84
0010 8f d2 4a 8f 1f 10 87 35 23 98 ac 4c 69 7f 1f 13 ..J...5 #..Li..
0020 a1 fe 47 23 51 82 57 00 e1 41 2b cd 38 dc 64 68 ..G#Q·W· ·A+·8·dh
0030 8d 10 2e 82 36 5d 28 46 cd c0 b4 e3 2e b6 eb 2e ..·6](F
0040 03 f9 1f 1d 96 c3 61 be 94 10 14 d0 3f 4a 08 01a?J..
0050 71 40 b7 70 2e 5c 00 14 c5 a3 7f 1f 27 8c 87 a8 q@·p·\·'

Frame (275 bytes) Decrypted TLS (199 bytes) Decompressed Header (391 bytes)

“status 200” and length fields compressed to only 4 bytes

Wireshark · Packet 33276 · Wi-Fi

Header Block Fragment [truncated]: 188210011f03848fd24a8f1f108735239e ^
[Header Length: 391]
[Header Count: 11]

▼ Header: :status: 200 OK
Name Length: 7
Name: :status
Value Length: 3
Value: 200
:status: 200
[Unescaped: 200]
Representation: Literal Header Field never Indexed - Indexed Name
Index: 8

0000 00 00 00 07 3a 73 74 61 74 75 73 00 00 00 03 32:sta tus....2
0010 30 30 00 00 00 0d 61 63 63 65 70 74 2d 72 61 6e 00....ac cept-ran
0020 67 65 73 00 00 00 05 62 79 74 65 73 00 00 00 0c ges....b ytes....
0030 63 6f 6e 74 65 6e 74 2d 74 79 70 65 00 00 00 09 content- type....
0040 69 6d 61 67 65 2f 67 69 66 00 00 00 04 65 74 61 image/gi f....eta
0050 67 00 00 00 2d 22 61 64 34 62 30 66 36 30 36 65 g...."ad 4b0f606e

Frame (275 bytes) Decrypted TLS (199 bytes) Decompressed Header (391 bytes)

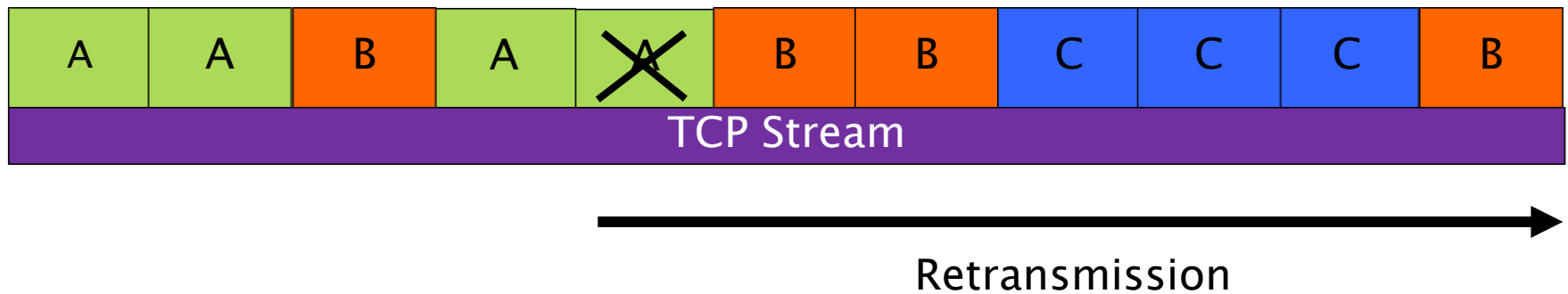
Decompressed header

HTTP3 Main Motivations

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

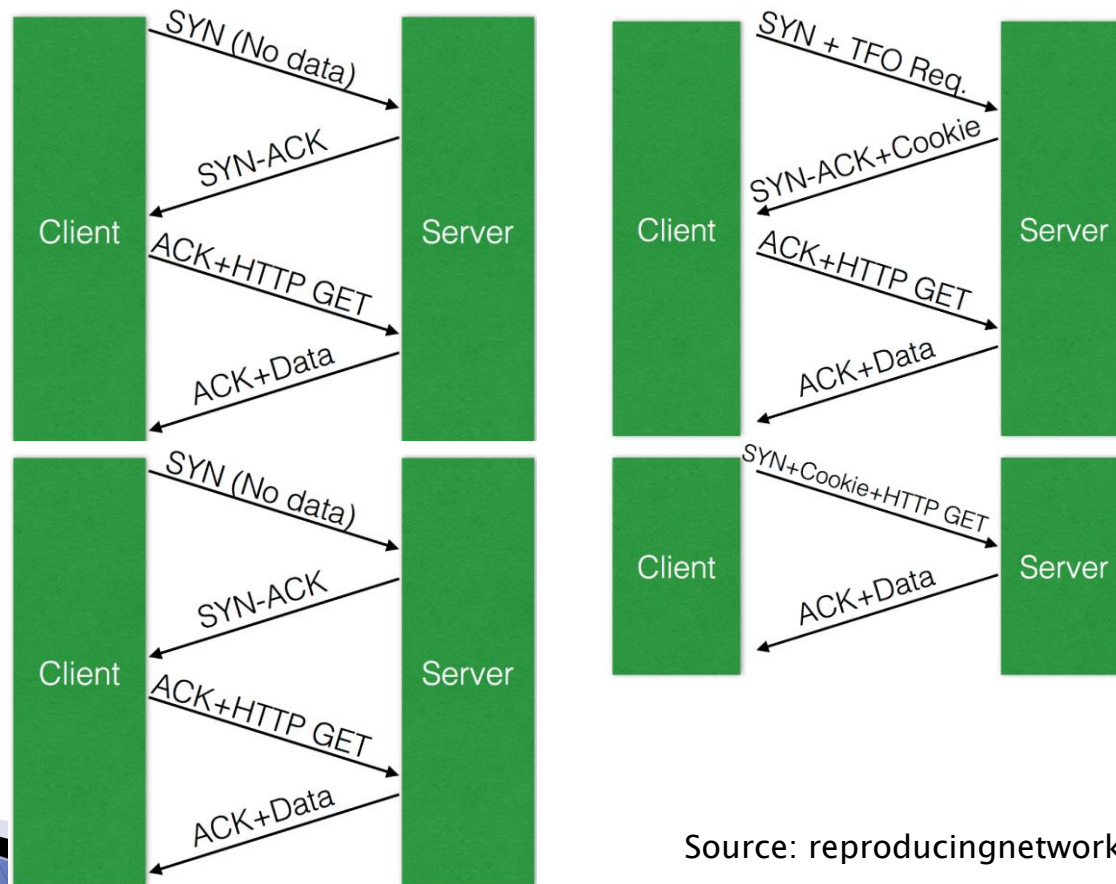
HTTP/2 – Head of Line Blocking

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



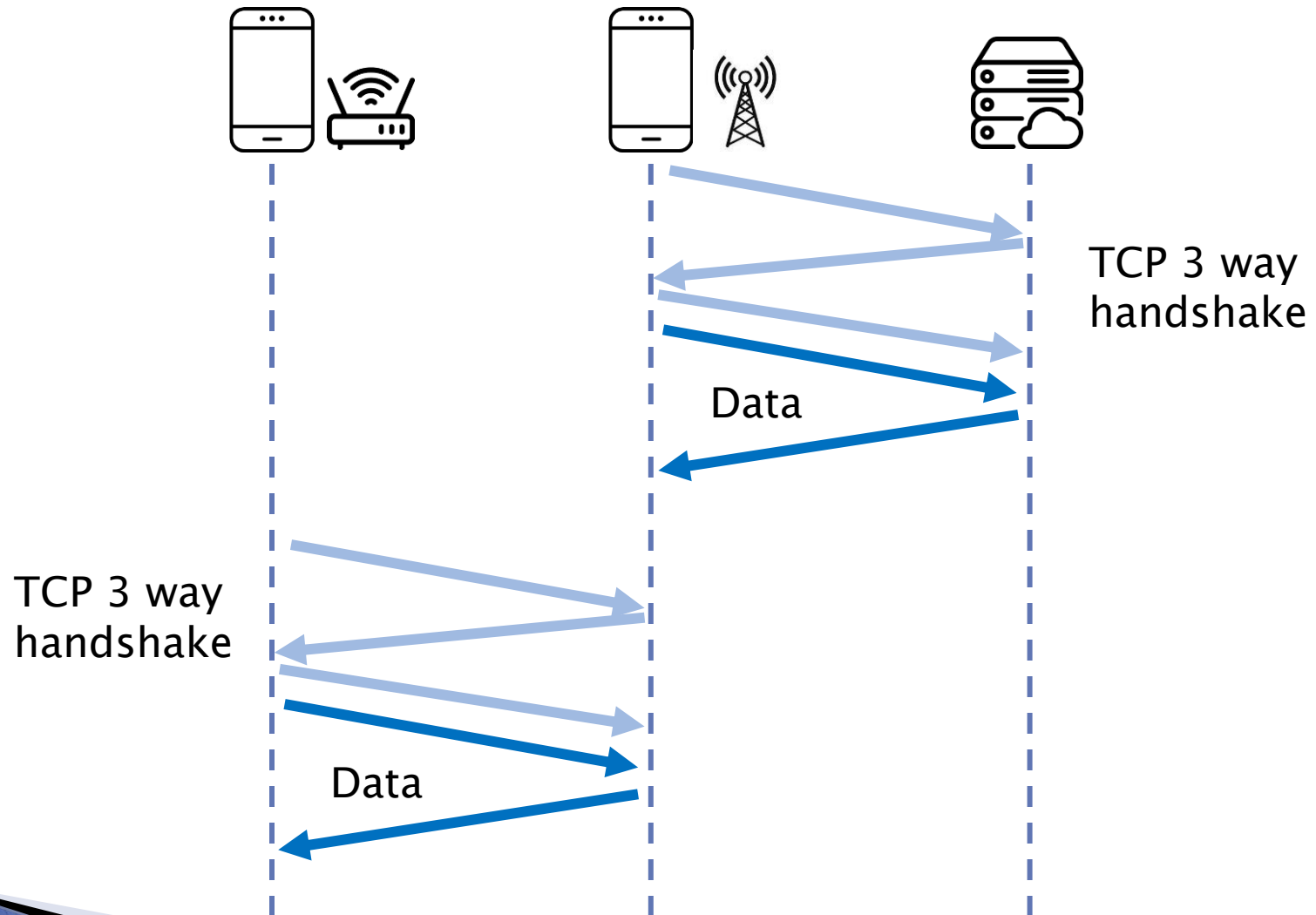
TCP “Ossification”

- ▶ 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



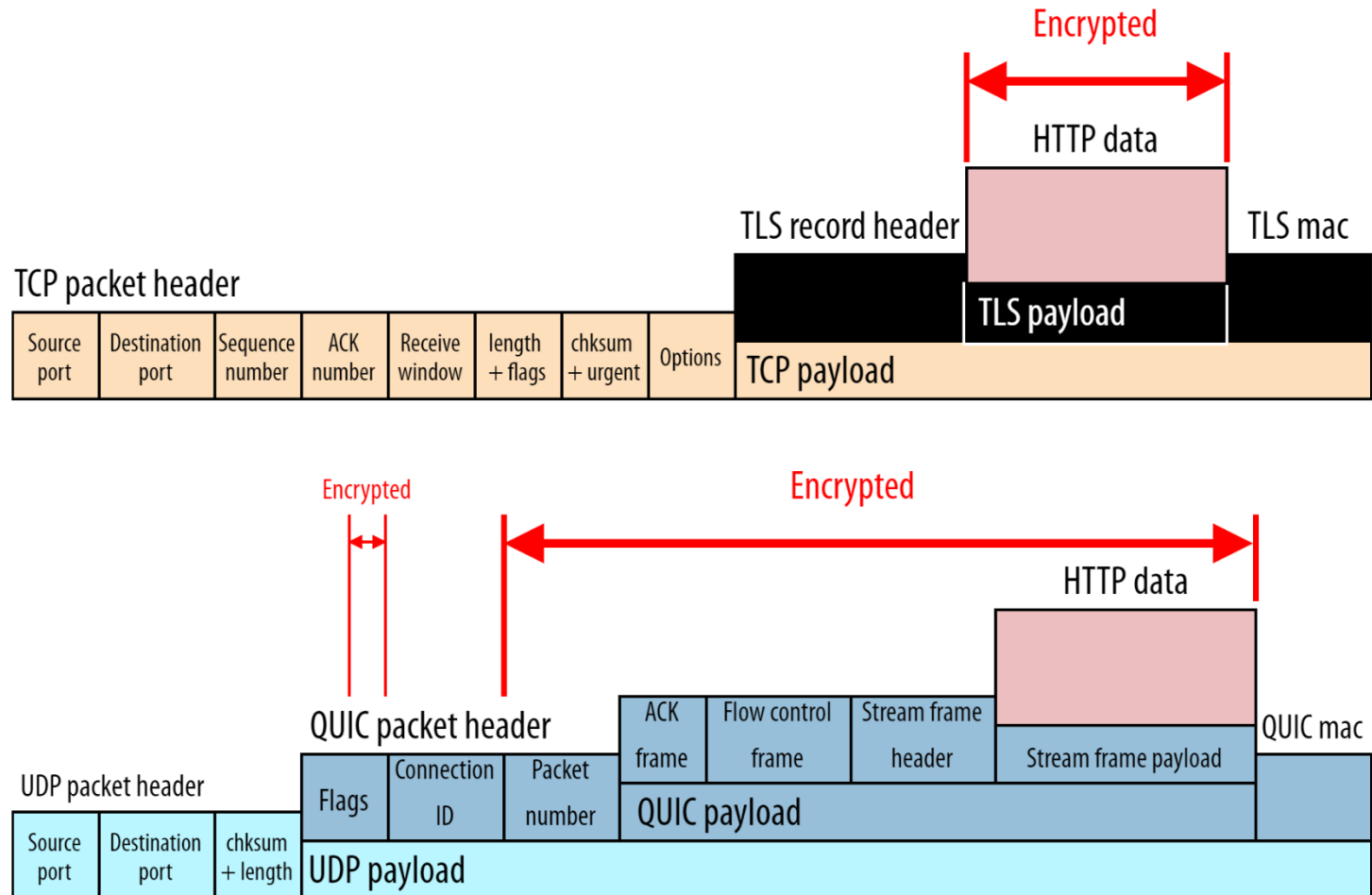
QUIC

- ▶ 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

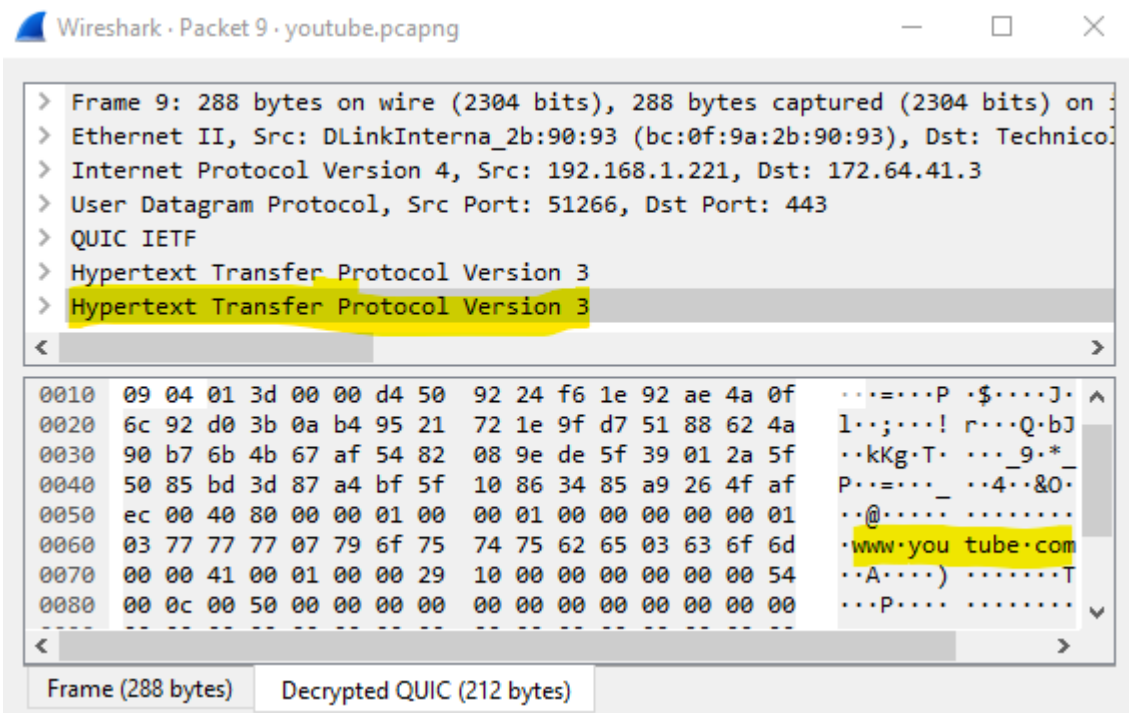
✓ Server Name Indication extension

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”

▼ Header: alt-svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Name Length: 7
Name: alt-svc
Value Length: 46
Value: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000

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

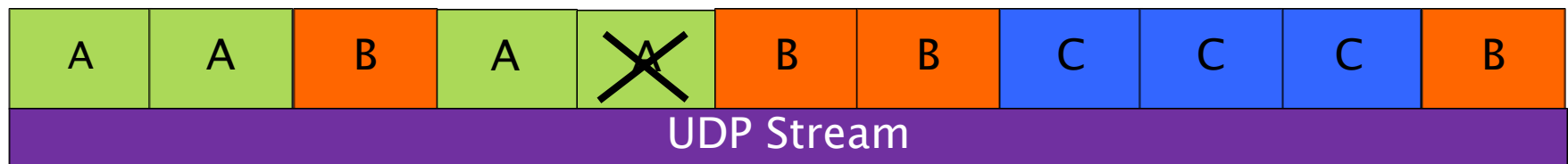
HTTP3 Main Motivations

Recall these issues

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

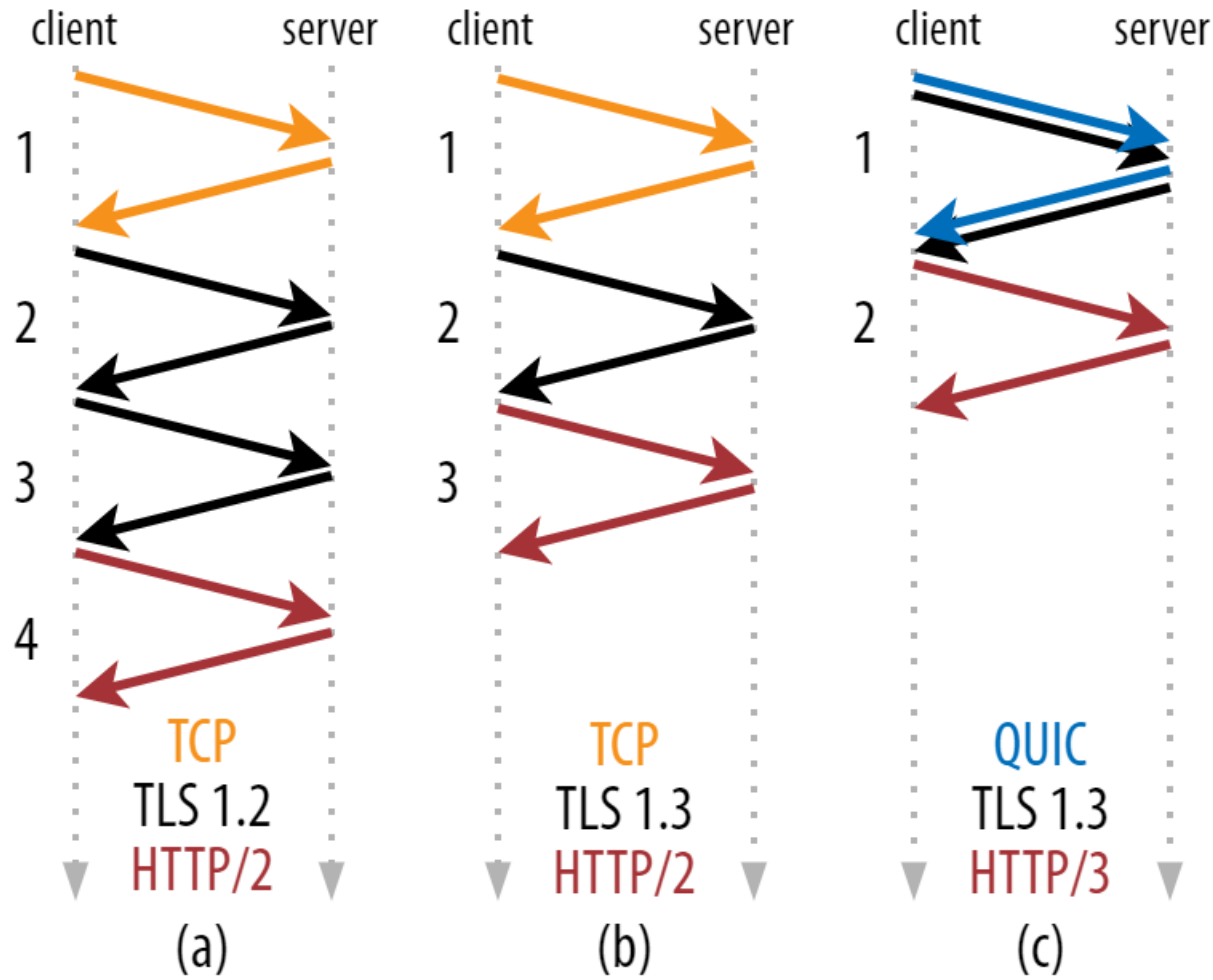
HTTP/3 – No Head of Line Blocking

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



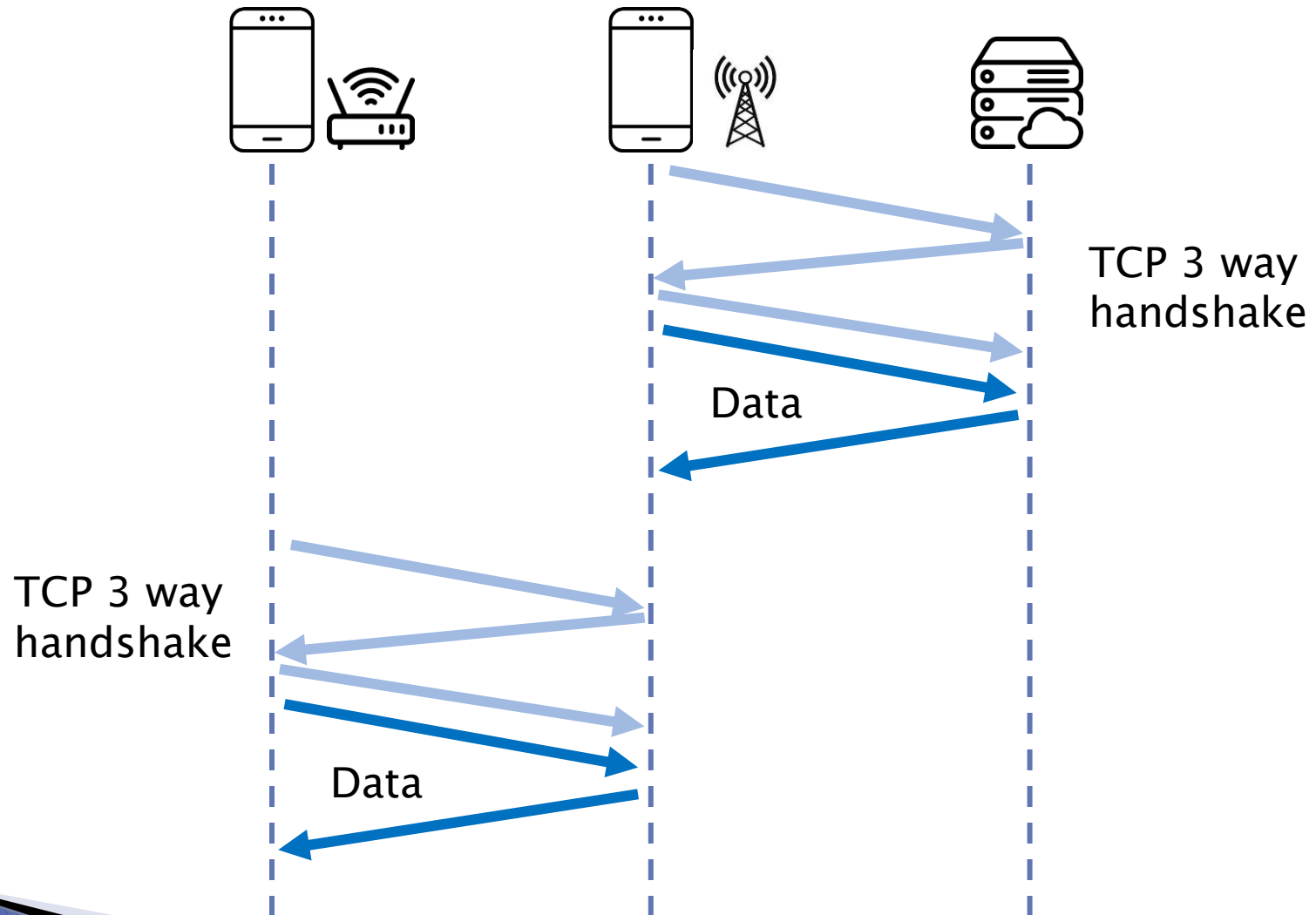
➡
Retransmission

QUIC RTT

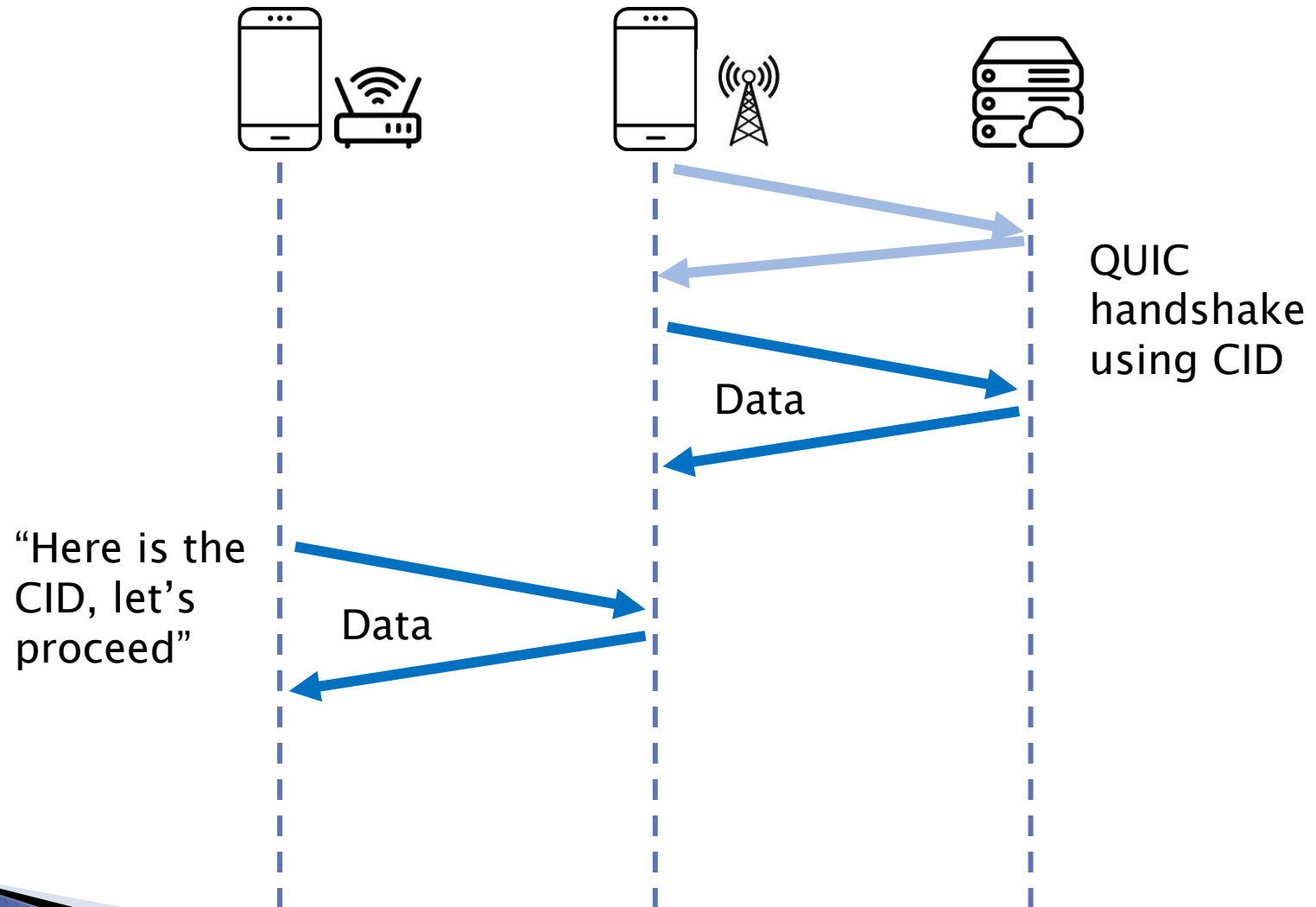


Source: <https://www.smashingmagazine.com/2021/08/http3-core-concepts-part1/>

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

