How Facebook Brought QUIC To Billions

An Engineering Journey

QCon '21

Matt Joras

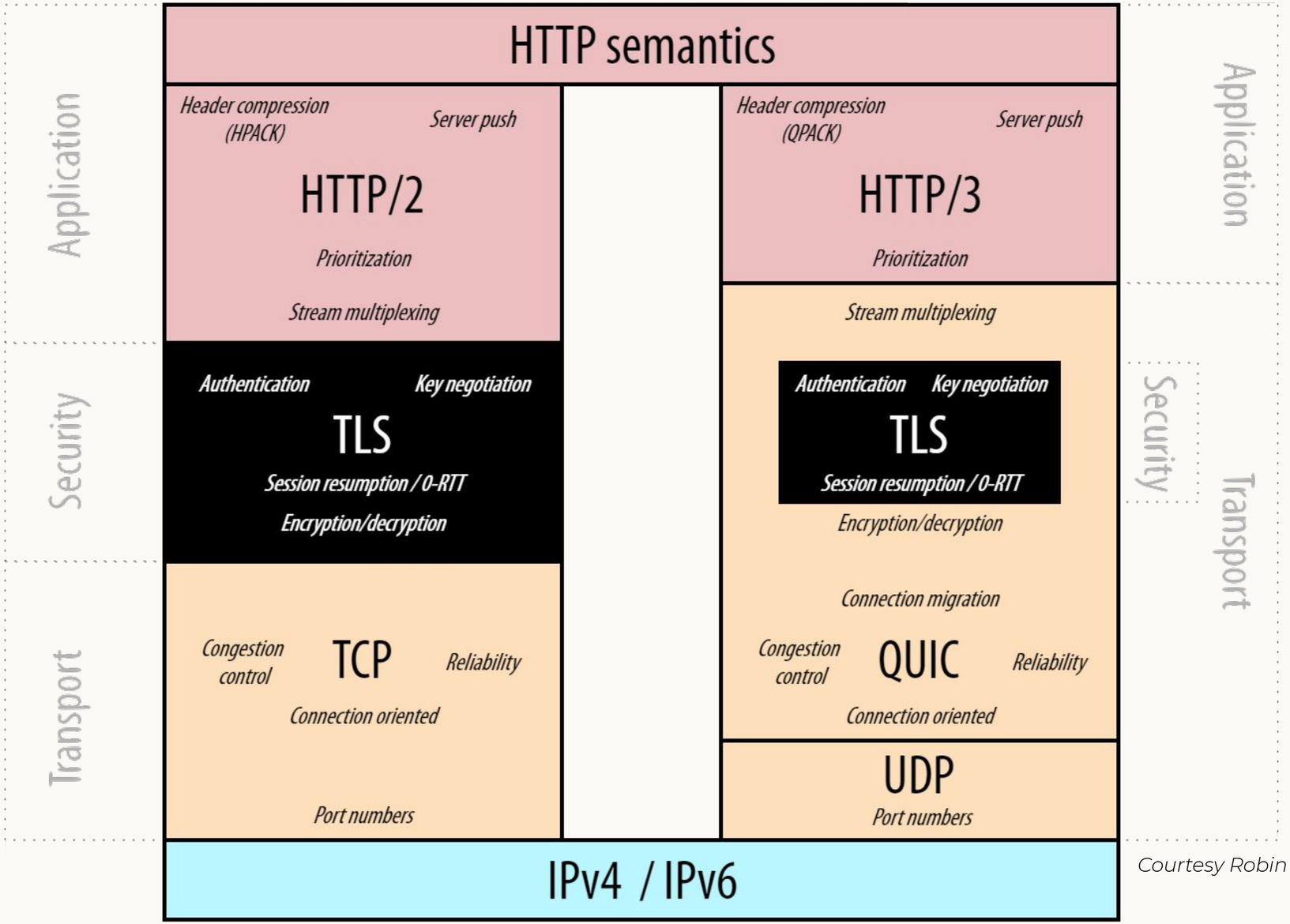
Software Engineer, Traffic Protocols
Co-chair IETF QUIC WG
mjoras@fb.com

Yang Chi

Software Engineer, Traffic Protocols
yangchi@fb.com

What is QUIC? The next Internet transport protocol.

What is QUIC?



Courtesy Robin Marx @programmingart

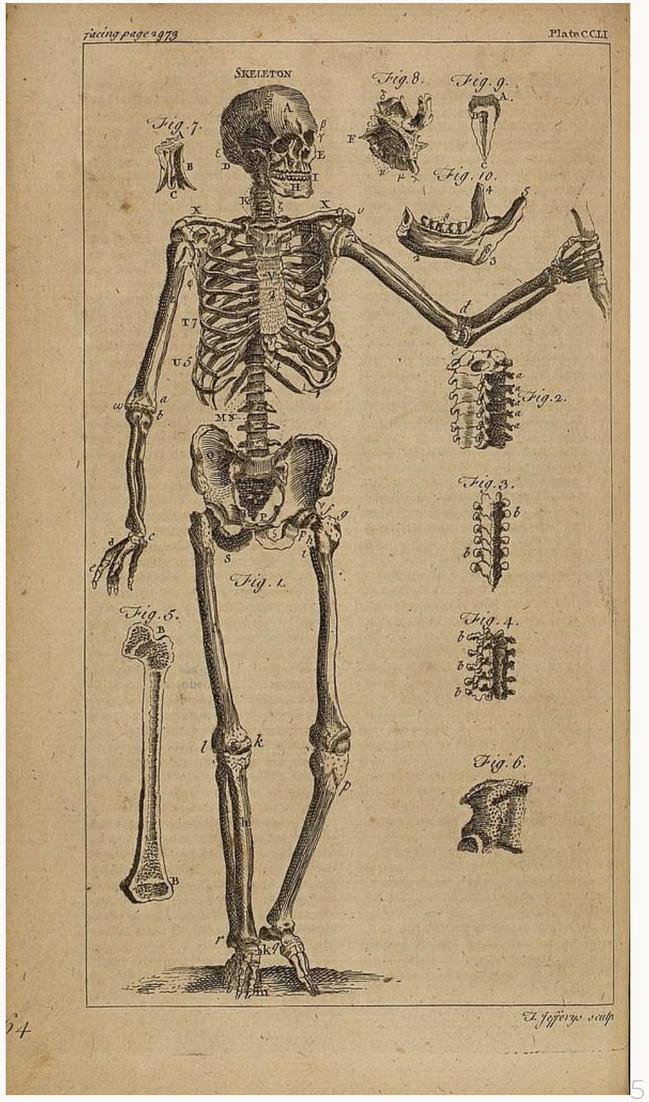
Why QUIC?

TCP Ossification

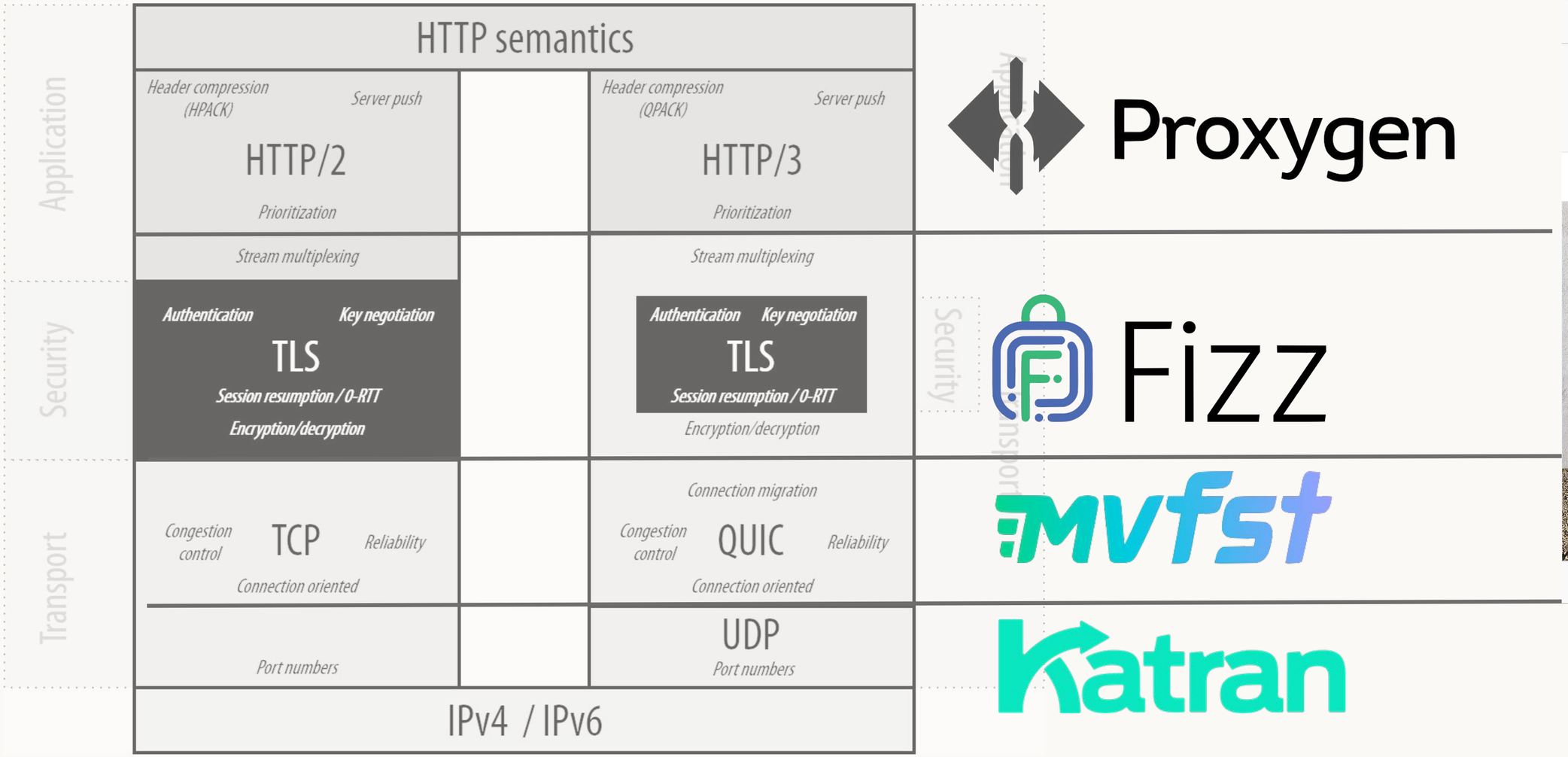
User space flexibility

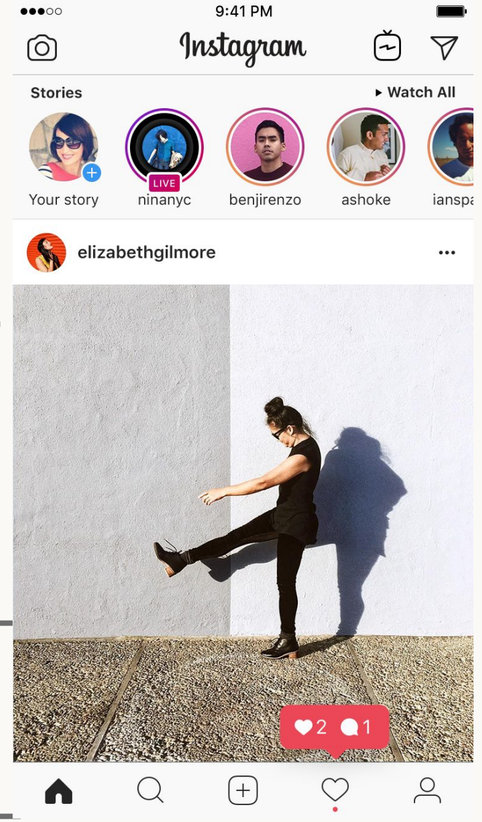
Truly independent multiplexing

State of art loss recovery



QUIC@FB


Application
Security
Transport
HTTP semantics
Header compression (HPACK)
Server push
HTTP/2
Prioritization
Header compression (QPACK)
Server push
HTTP/3
Prioritization
Stream multiplexing
Authentication
Key negotiation
TLS
Session resumption / 0-RTT
Encryption/decryption
Stream multiplexing
Authentication
Key negotiation
TLS
Session resumption / 0-RTT
Encryption/decryption
Congestion control
TCP
Reliability
Connection oriented
Connection migration
Congestion control
QUIC
Reliability
Connection oriented
Port numbers
UDP
Port numbers
IPv4 / IPv6
Proxygen
Fizz
mvfst
Katran


9:41 PM
Instagram
Stories
Watch All
Your story
ninanyc
benjirenzo
ashoke
elizabethgilmore

Sounds great. So you just turned on QUIC and everything got better?

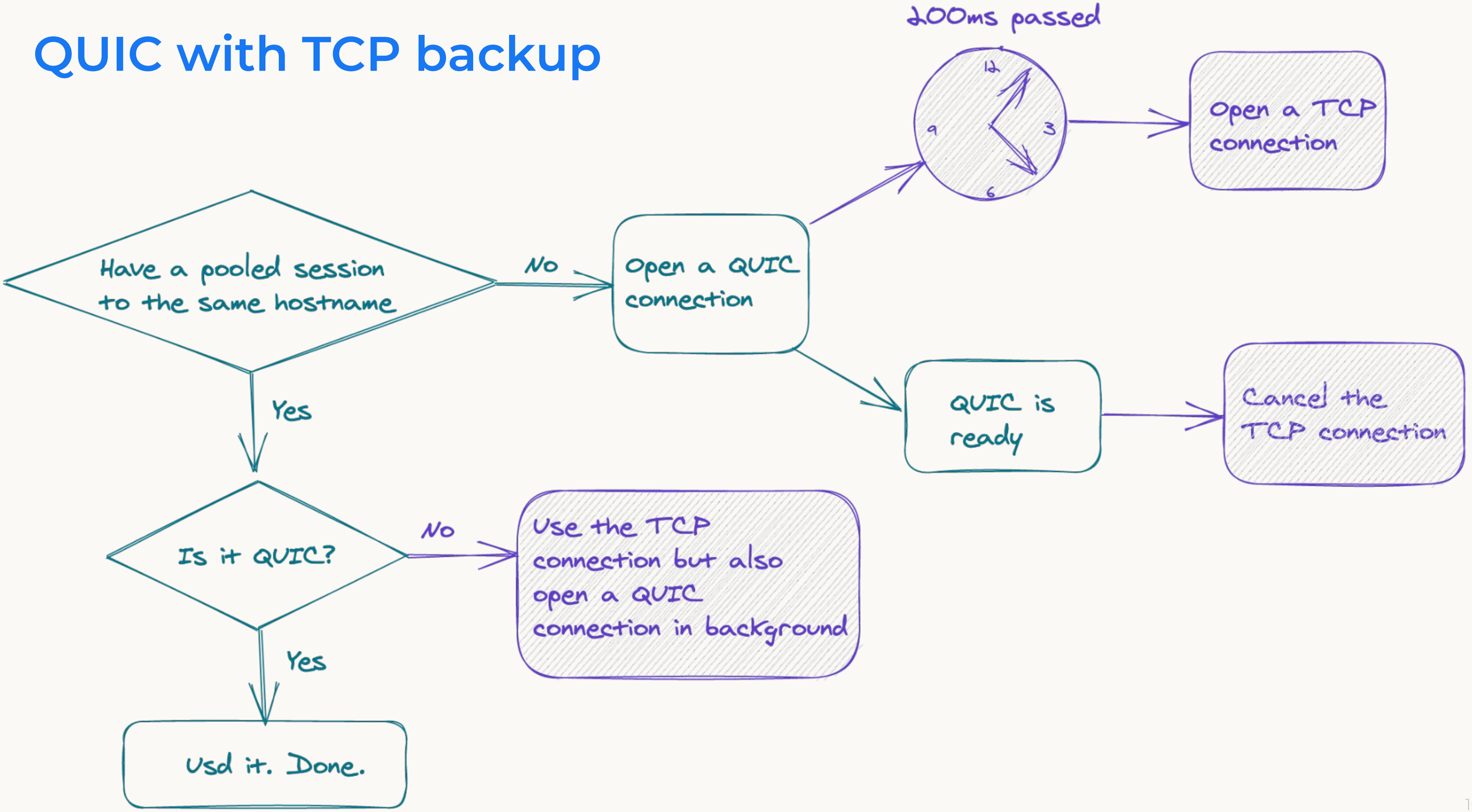
Mobile apps required significant investigation to iron out issues with QUIC.

In several cases, app behavior and some settings were implicitly tuned for TCP.

FB & IG Baselines

		
Transport	TCP + TLS 1.3	TCP + TLS 1.3
GraphQL / API requests	Single HTTP/2	Single HTTP/2
Image requests	Single HTTP/2	Up to 6 HTTP/1.1 connections
Video requests	Single HTTP/2	Up to 6 HTTP/1.1 connections
Connection sharding	GraphQL, Images and videos cannot share connections	Images and videos can use the same connections. API is over separate connection.
Server side stack	Vanilla Linux TCP + BBRv1	Vanilla Linux TCP + BBRv1

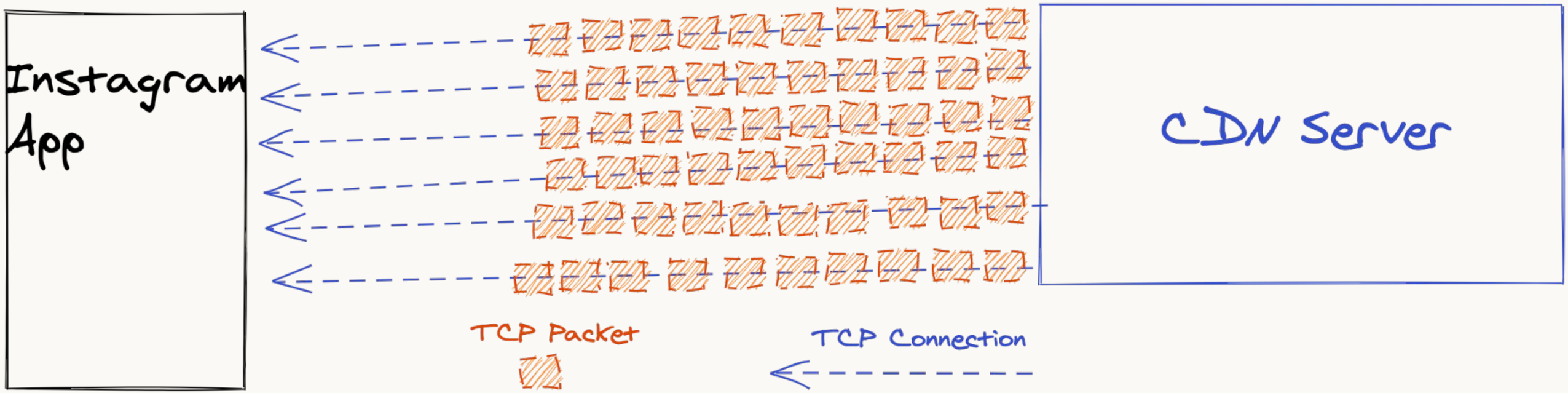
QUIC with TCP backup



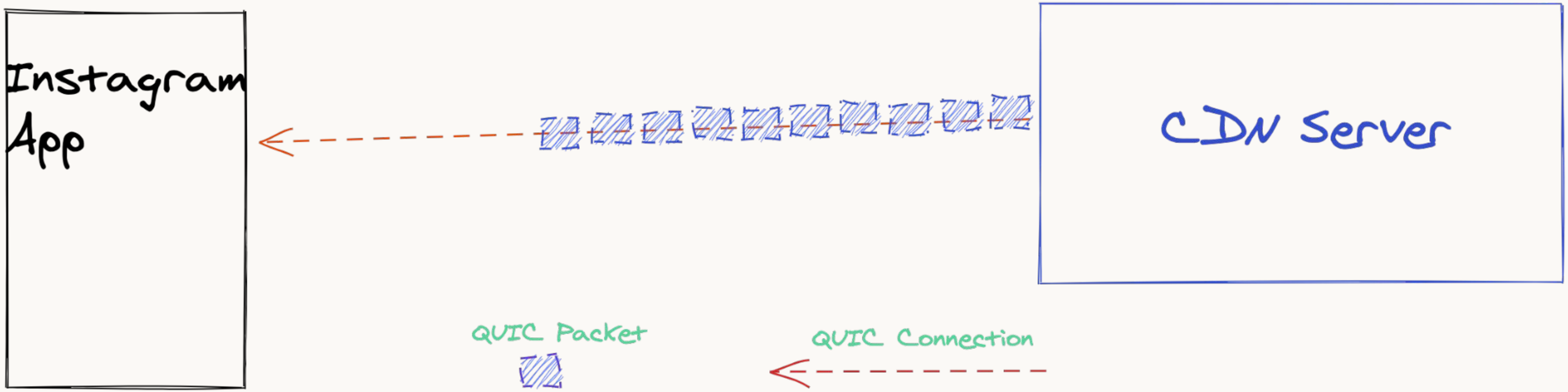
Congestion Control and Flow Control

Congestion Control	Algorithm used to minimize loss and maximize throughput.
Flow Control	Limits the amount of data the applications buffers.

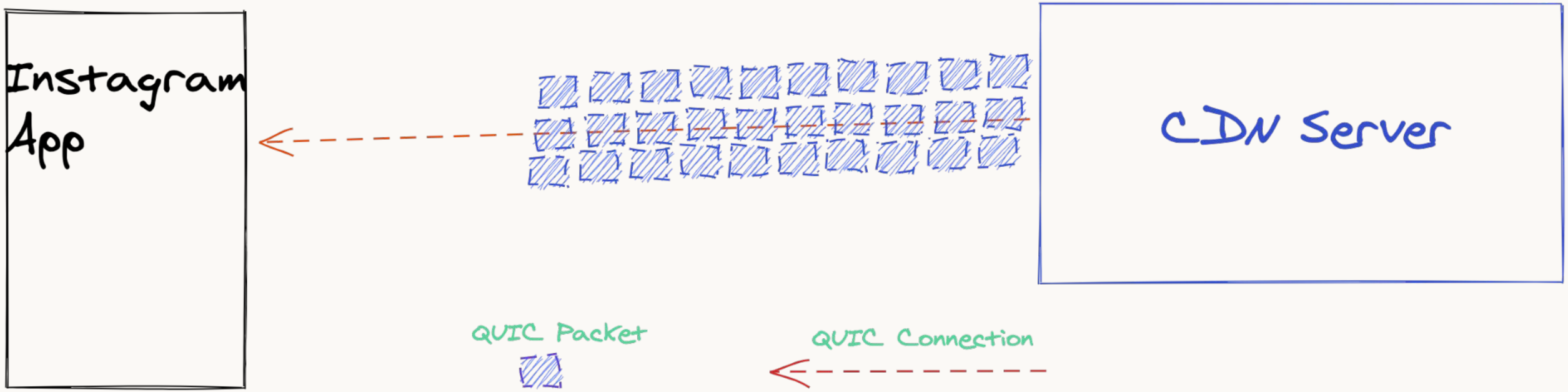
Initial flight: 6 TCP conns vs 1 QUIC conn



Initial flight: 6 TCP conns vs 1 QUIC conn



Initial flight: 6 TCP conns vs 1 QUIC conn



Congestion Control and Congestion Window

bandwidth * delay = data needed to achieve full throughput
10Mbps * 100ms = 122KB = ideal "congestion window"

TCP + HTTP/2 Has Two Levels of Flow Control

TCP Flow Control

- Determined by TCP stack.
- Shrinks and grows over time.

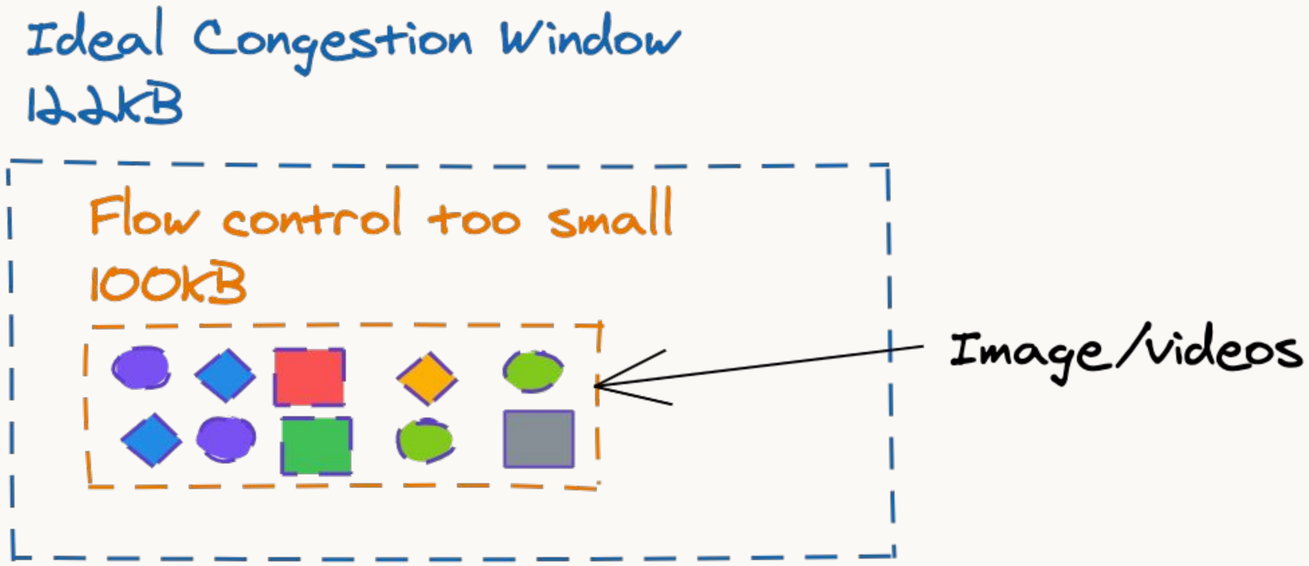
HTTP/2 Flow Control

- Set by application.
- Static in size.

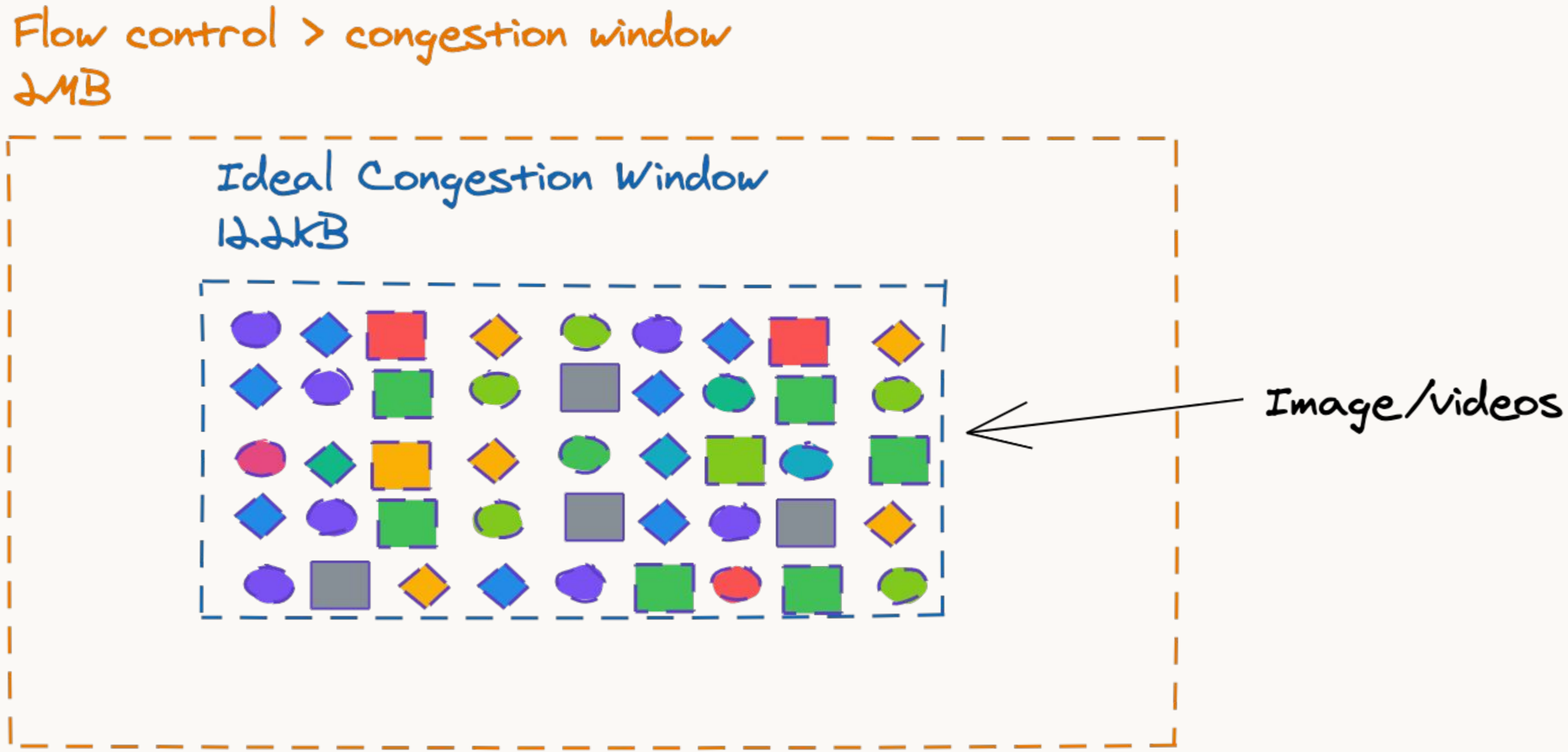
QUIC Flow Control

- Set by application.
- Static in size.

Flow Control Constrains Bandwidth

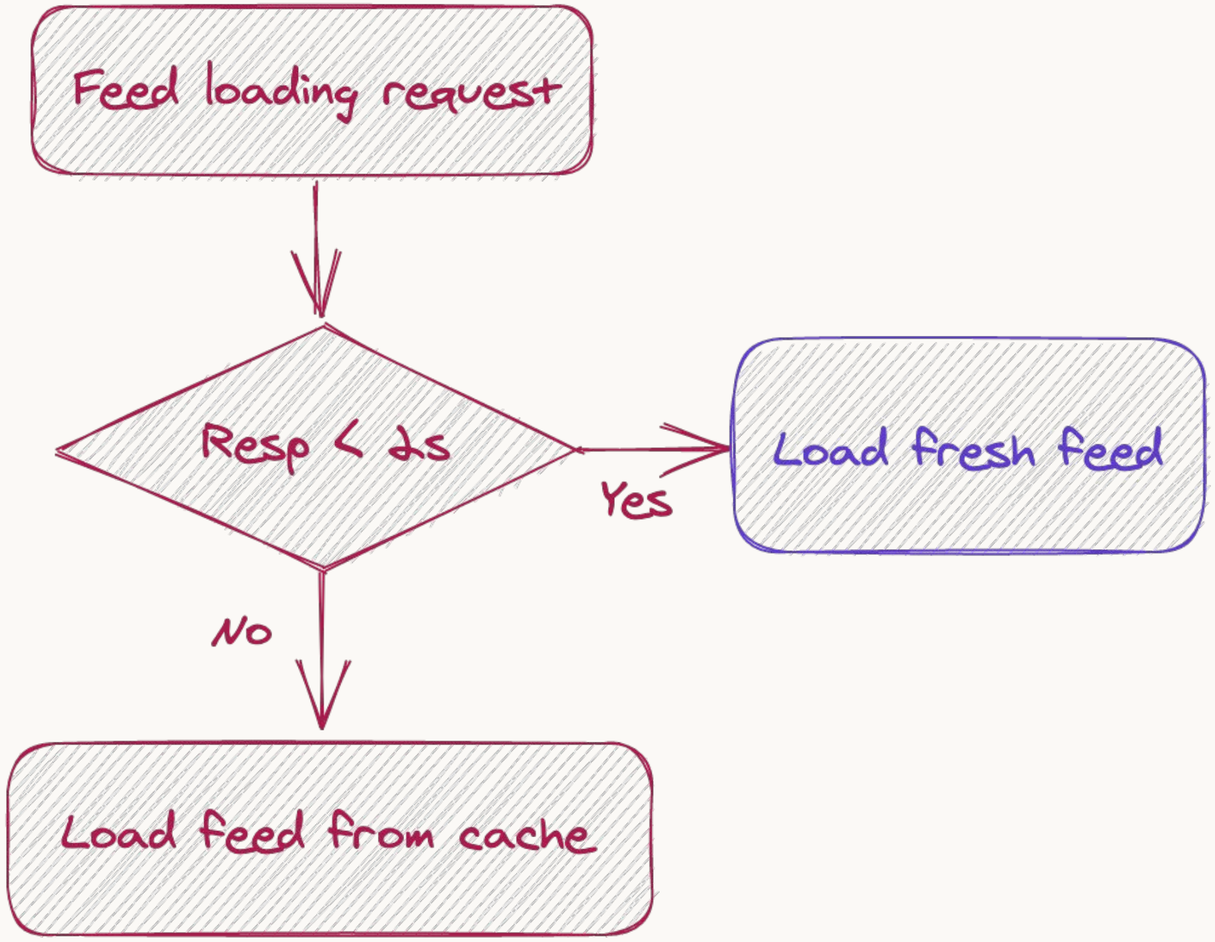


Increasing Flow Control Solves The Problem



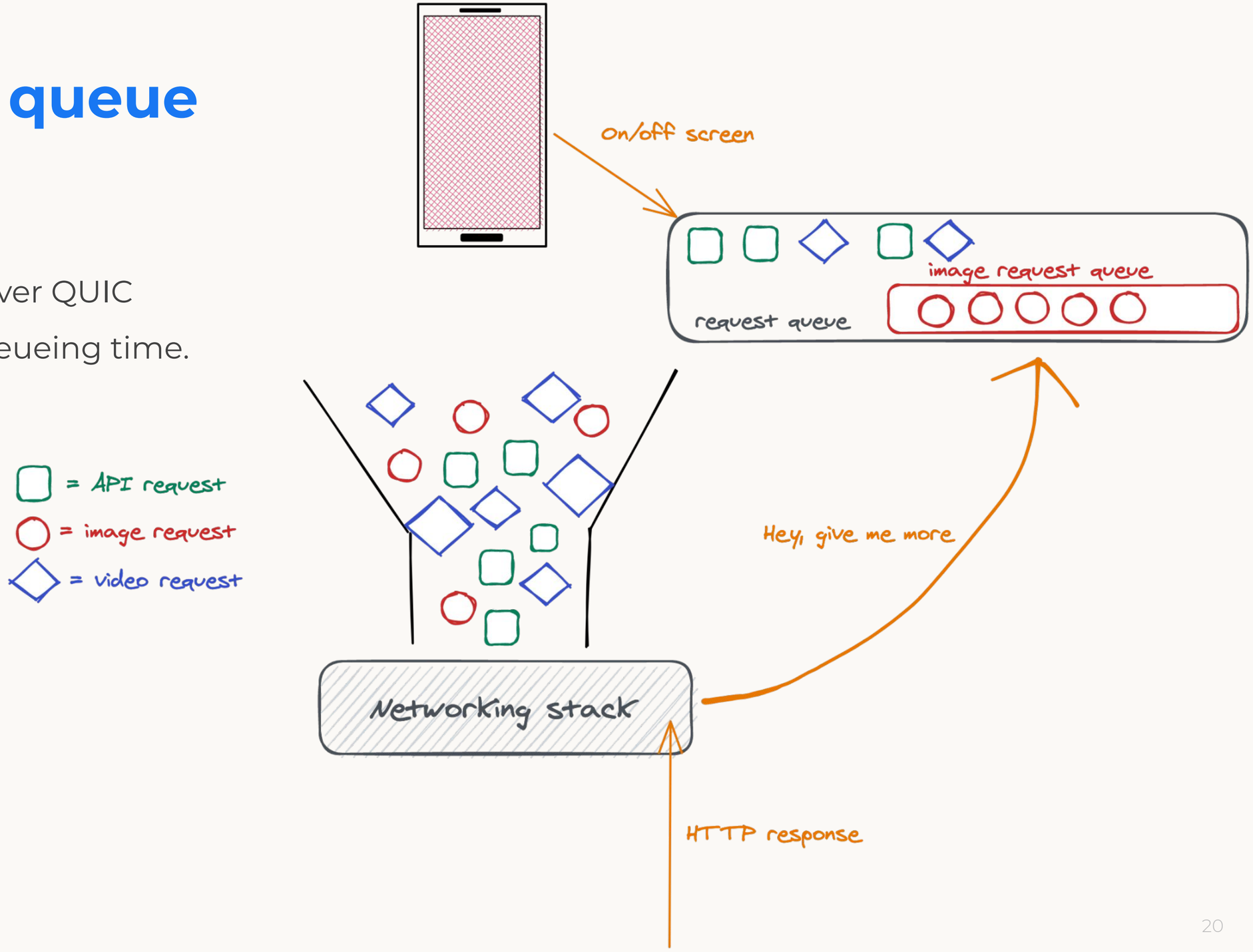
Clever heuristics aren’t always that clever

- Enabled QUIC and HTTP/3 for GraphQL
  - -> Regressed images and video



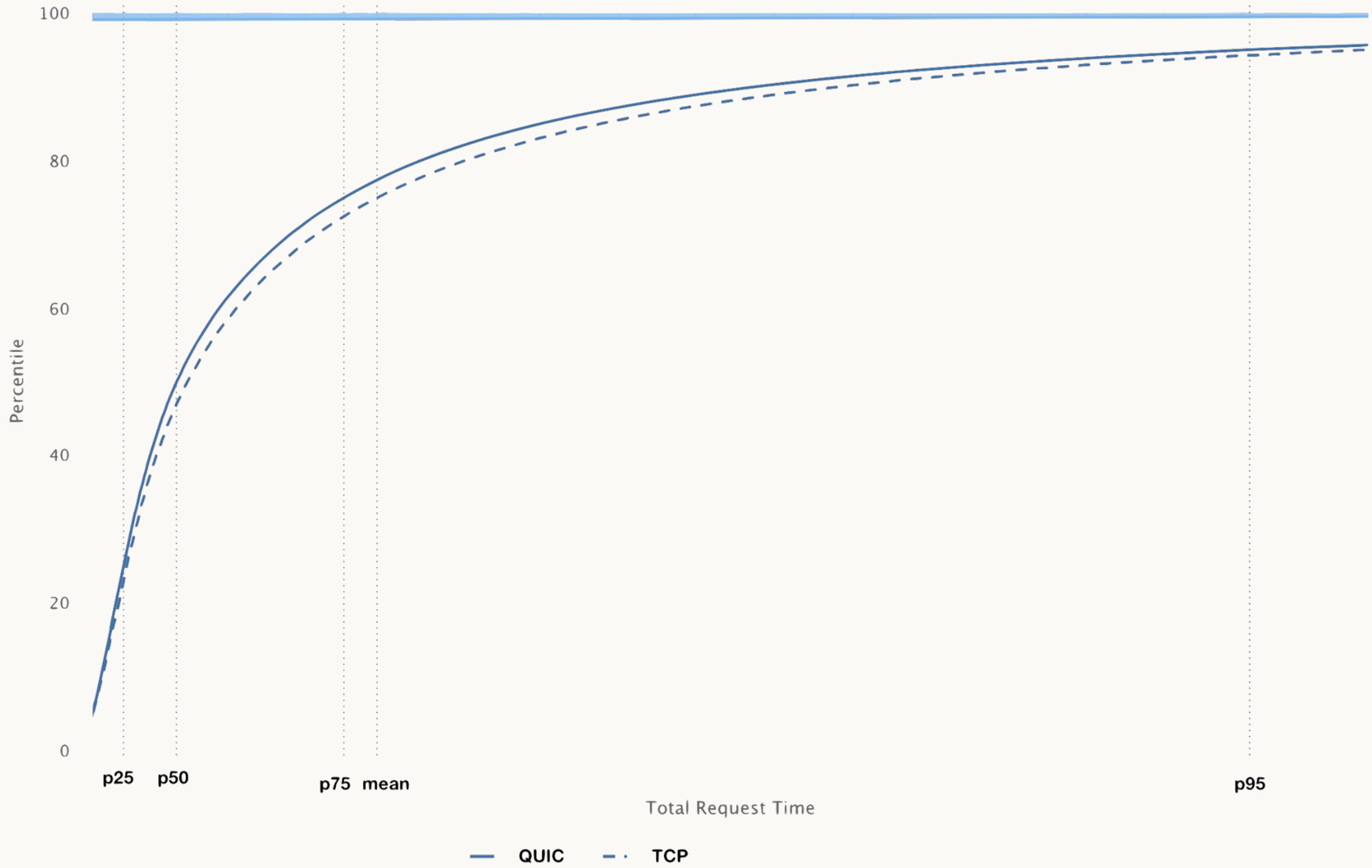
App request queue

- Higher video quality over QUIC
  - -> Longer image queueing time.

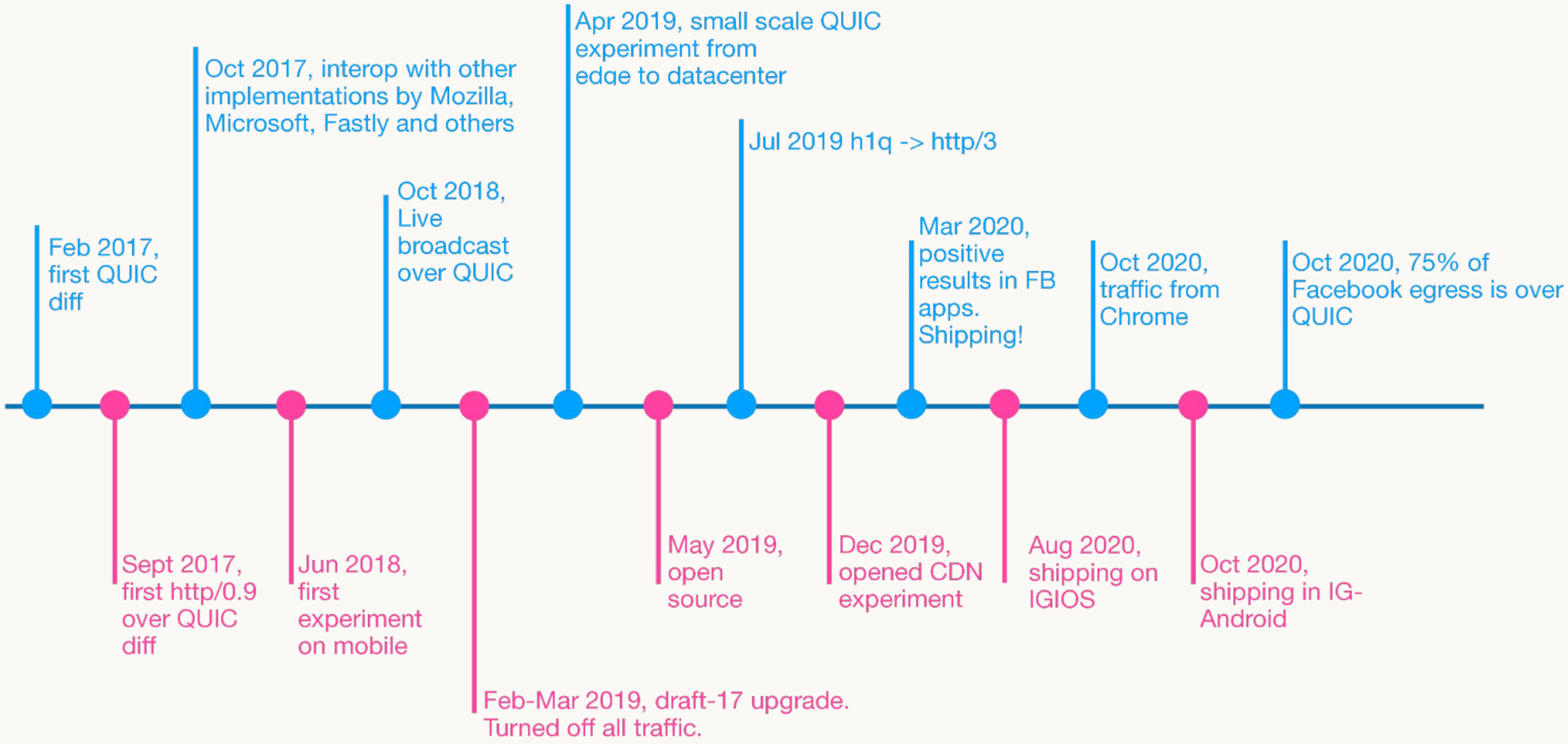


QUIC Had a Transformative Effect on Video

- 22% increase in MTBR
- 20% decrease in video stalls
- 8% reduced video error rate



The voyage


Feb 2017, first QUIC diff
Sept 2017, first http/0.9 over QUIC diff
Oct 2017, interop with other implementations by Mozilla, Microsoft, Fastly and others
Jun 2018, first experiment on mobile
Oct 2018, Live broadcast over QUIC
Feb-Mar 2019, draft-17 upgrade. Turned off all traffic.
Apr 2019, small scale QUIC experiment from edge to datacenter
May 2019, open source
Jul 2019 h1q -> http/3
Dec 2019, opened CDN experiment
Mar 2020, positive results in FB apps. Shipping!
Aug 2020, shipping on IGIOS
Oct 2020, traffic from Chrome
Oct 2020, shipping in IG-Android
Oct 2020, 75% of Facebook egress is over QUIC

Takeaways

Tuning is needed for QUIC to deliver great result

- Application behavior can be tricky and implicitly coupled to TCP behavior.
- Often HTTP and Transport metrics look good while Application metrics will regress.
- Putting in the effort to tune usage for QUIC is worth the effort.

Future Work

0-RTT and Connection migration

- Didn’t need them to show improvements over TCP.
- Connection reuse from pooling already extremely high in our stack.
- 0-RTT in progress.
- Connection migration has a lot of operational complexity.

Thank you