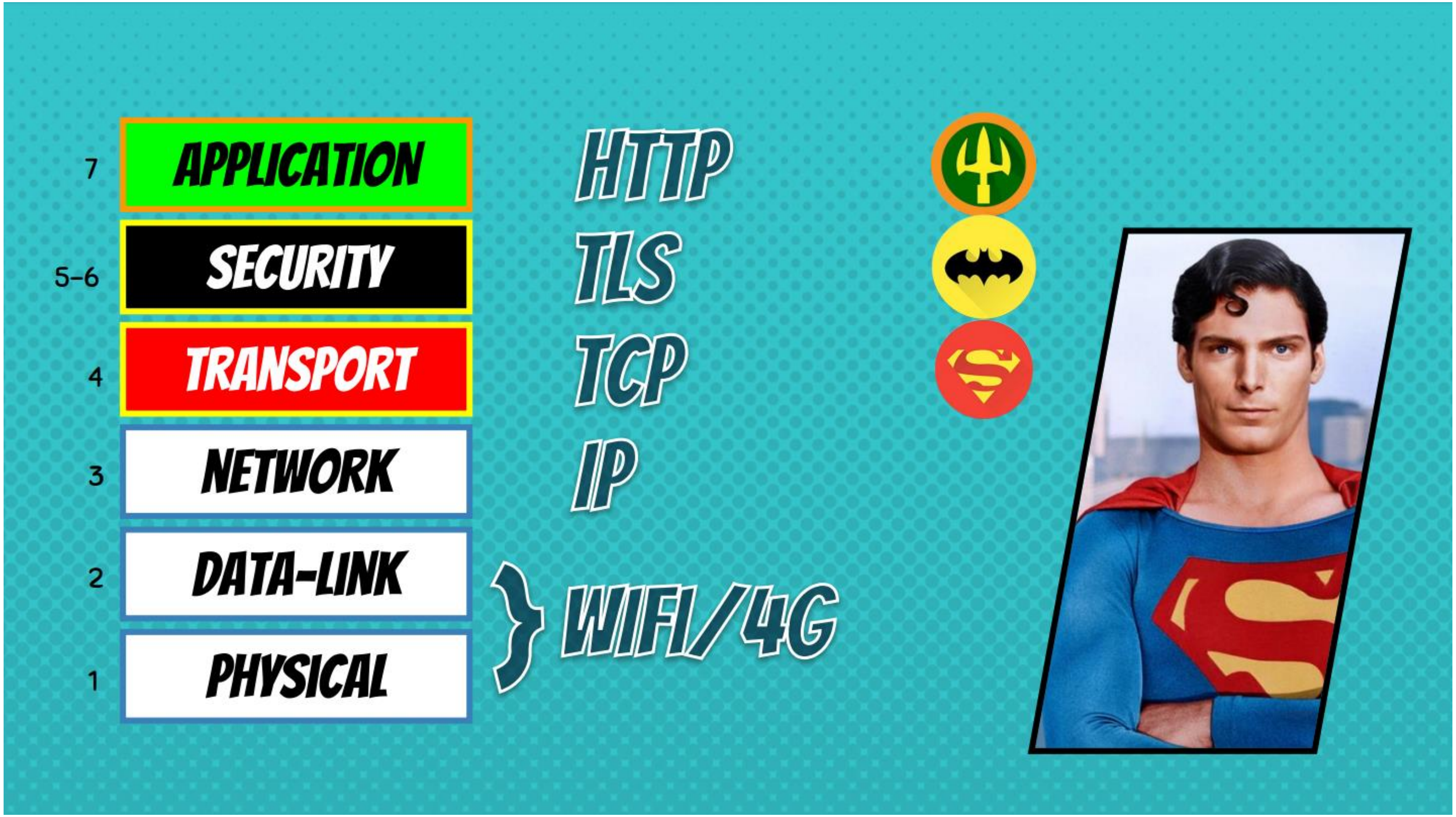
QUIC: the details

Robin Marx - @programmingart
PhD researcher – Hasselt University

UHASSELT EDM

https://quic.edm.uhasselt.be
Curl-up – Prague – March 2019

QUIC in Theory and Practice @ DeltaV 2018



My self-worth is determined by random, white men on the internet

3

Richard M. 9 months ago

Using comic book characters and attempts to be funny fell flat. Don't do such things when your audience may know little about comics and are looking for technical info. You lost me one or two minutes in.

2 REPLY

James Fuller

to me

The curl up day (Sat/Sun) are 'turn it up to 11' in terms of technical level,

https://www.youtube.com/watch?v=B1SQFjIXJtc
http://internetonmars.org/deltav.pdf

QUIC is "quite" complex…

- 6 "Core" specifications:
  - QUIC invariants : 9 pages
  - QPACK header compression : 37 pages
  - Recovery: loss detection and congestion control : 38 pages
  - TLS mapping : 43 pages
  - HTTP/3 : 53 pages
  - Core transport : 139 pages

319 pages total

(9 more than The Hobbit)

2566 issues on github

- At least 20 more side-specs
  - Applicability, manageability, spinbit, DATAGRAM, …

https://github.com/quicwg/base-drafts

1. Connection setup: Similar to TLS 1.3, but small differences

- No “End-of-early-data”
- No Record layer

- 0-RTT is done in QUIC itself
- QUIC performs all actual encrypt/decrypt operations

```
 Client                                              Server

 ClientHello
(0-RTT Application Data)  -------->
                                                ServerHello
                                       {EncryptedExtensions}
                                                  {Finished}
                          <--------       [Application Data]
{Finished}                -------->

[Application Data]        <------->       [Application Data]

 () Indicates messages protected by early data (0-RTT) keys
 {} Indicates messages protected using handshake keys
 [] Indicates messages protected using application data
    (1-RTT) keys

              Figure 1: TLS Handshake with 0-RTT
```


https://tools.ietf.org/html/draft-ietf-quic-tls-19

1. Connection setup: TLS is (largely) abstracted out

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Offset (i)                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Length (i)                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Crypto Data (*)                      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: CRYPTO Frame Format

TLS is just another opaque bytestream

cryptoData = TLS.getTLSstuff()
sendCryptoFrame(cryptoData)
...
TLS.processTLSstuff(cryptoData)
TLS.onNewKeyAvailable(keyCallback)

https://tools.ietf.org/html/draft-ietf-quic-tls-19

1. Connection setup: TLS is (largely) abstracted out: CRYPTO frames

```
RX Initial(0x0), Version: 0xff00000f, Dest CID: 0x1156a3c691c16e8c390499d90aac254b21, Src CID: 0x0736553ca2daa8
     PKN: 0, payload length: 1184
  CRYPTO (0x18)
  length=304 offset=0
  data=SSL3_MT_CLIENT_HELLO (PROBABLY, but could be different or more if split or coalesced packet)
  PADDING (0x0)
  length= 859
```

```
TX Handshake(0x2), Version: 0xff00000f, Dest CID: 0x0736553ca2daa8, Src CID: 0x0f72f17fbaab34b215b0990e570661
     PKN: 0, payload length: 1220
  CRYPTO (0x18)
  length=1200 offset=0
  data=SSL3_MT_ENCRYPTED_EXTENSIONS (PROBABLY, but could be different or more if split or coalesced packet)
```

1. Connection setup: Different encryption levels and packet number spaces

4 different "keys"

- Initial (AEAD) : Can't be changed, but can be read
- 0-RTT (pre-shared) : Can't be changed or read, but replayed
- Handshake
- 1-RTT

Packet Type	Encryption Level	PN Space
Initial	Initial secrets	Initial
0-RTT Protected	0-RTT	0/1-RTT
Handshake	Handshake	Handshake
Short Header	1-RTT	0/1-RTT

Table 1: Encryption Levels by Packet Type

0, 1

0, 1, 2, 3

0, 1, 2, …, 10

4 … 9000+

https://github.com/quicwg/base-drafts/issues/1018

1. Connection setup: Different encryption levels and packet number spaces

4 different "keys"

- Initial (AEAD) : Can't be changed, but can be read
- 0-RTT (pre-shared) : Can't be changed or read, but replayed
- Handshake
- 1-RTT

Packet Type	Encryption Level	PN Space
Initial	Initial secrets	Initial
0-RTT Protected	0-RTT	0/1-RTT
Handshake	Handshake	Handshake
Short Header	1-RTT	0/1-RTT

Table 1: Encryption Levels by Packet Type

everyone else	Winquic (works because PN gaps are allowed)
0, 1	0, 4
0, 1, 2, 3	1, 2, 3, 5
0, 1, 2, …, 10	6, 7, 8, …, 17
4 … 9000+	17 … 9000+

https://github.com/quicwg/base-drafts/issues/1018

1. Connection setup: Each encryption level has a separate packet type too

In this version of QUIC, the following packet types with the long header are defined:

Type	Name	Section
0x0	Initial	Section 17.2.2
0x1	0-RTT	Section 17.2.3
0x2	Handshake	Section 17.2.4
0x3	Retry	Section 17.2.5

Table 5: Long Header Packet Types

Only 1-RTT packets use the “short” header

https://tools.ietf.org/html/draft-ietf-quic-tls-19

1. Long packet headers: lots of information

```
+-+-+-+-+-+-+-+-+
|1|1| 2 |R R|P P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Version (32)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|DCIL(4)|SCIL(4)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (0/32..144)         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Connection ID (0/32..144)            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Length (i)                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Packet Number (8/16/24/32)               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Payload (*)                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

               Figure 12: Handshake Protected Packet
```

1. Long packet headers: information that never changes afterwards

```
+-+-+-+-+-+-+-+-+
|1|1| 2 |R R|P P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Ve[illegible]on (32)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|DC[illegible]4)|SC[illegible])|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (0/32..144)         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Conne[illegible]n ID (0/32..144)            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Le[illegible] (i)                      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Packet Number (8/16/24/32)               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Payload (*)                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Handshake Protected Packet

https://tools.ietf.org/html/draft-ietf-quic-transport-19

1. Short packet headers: optimized headers for most data packets

```
17.3.  Short Header Packets

   This version of QUIC defines a single packet type which uses the
   short packet header.

    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+
   |0|1|S|R|R|K|P P|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                Destination Connection ID (0..144)           ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Packet Number (8/16/24/32)              ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Protected Payload (*)                   ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                Figure 14: Short Header Packet Format
```


https://tools.ietf.org/html/draft-ietf-quic-transport-19

1. Header protection : actually more than 4 keys!

N different "keys"

- Initial (AEAD) : Can't be changed, but can be read
- 0-RTT (pre-shared) : Can't be changed or read, but replayed
- Handshake
- 1-RTT

- Header Protection key (same for all packet types)
  - Protects flags + Packet Number **(NOT connectionID!)**
- Unlimited amount of new 1-RTT keys
  - KEY_PHASE bit in the flags

2. ConnectionID

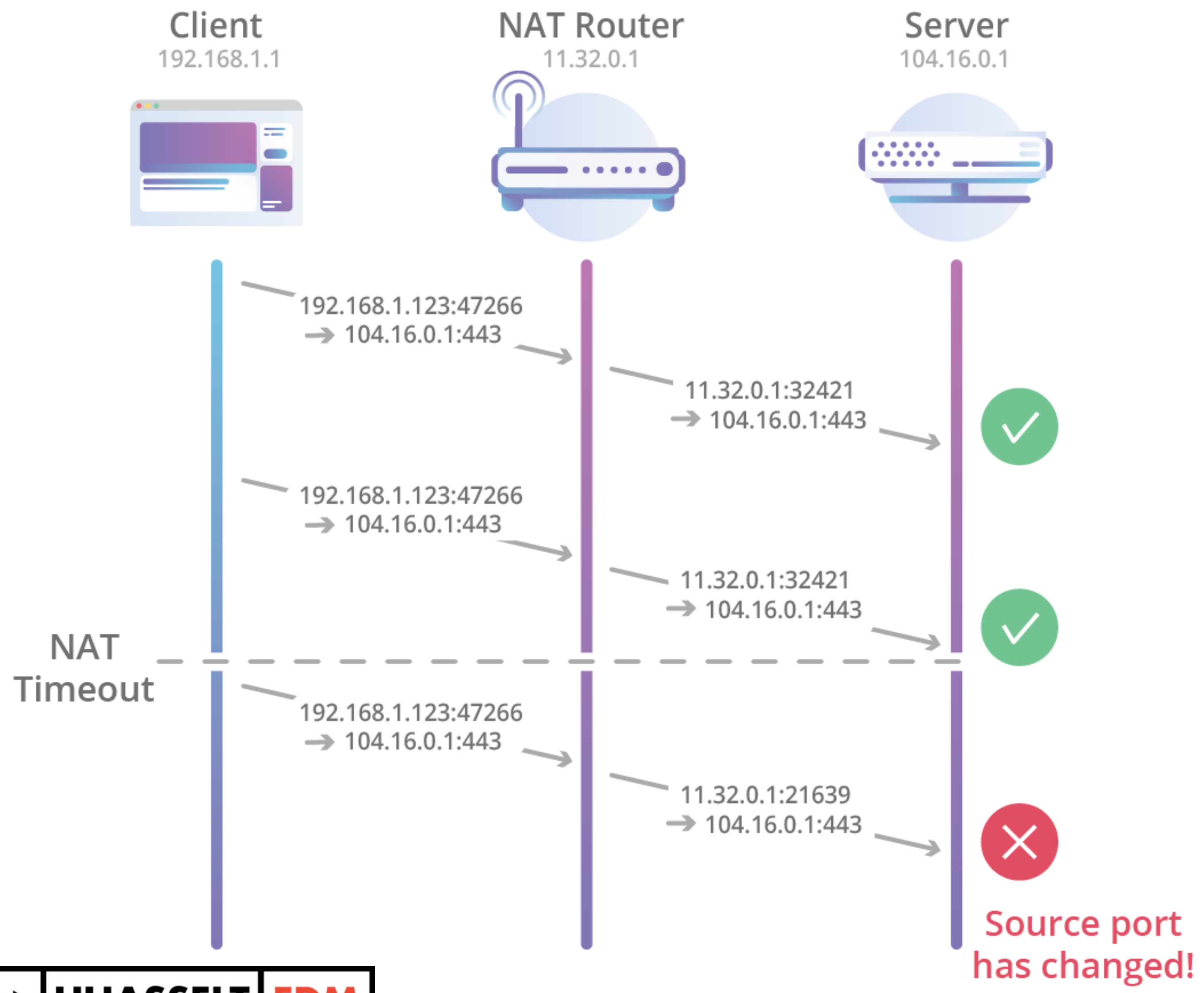
```
|DCIL(4)|SCIL(4)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (0/32..144)         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Connection ID (0/32..144)            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Destination Connection ID (0..144)           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Why do we need it?
- Why do we need 2? And then back to 1?
- Why is it so large?

https://tools.ietf.org/html/draft-ietf-quic-transport-19

2. ConnectionID: Connection migration and NAT rebinding



1) 4G <-> Wifi
2) Network Address Translation

Both cases can change IP and/or port

ConnectionID can stay the same
(or change to previously agreed upon values)

https://blog.cloudflare.com/the-road-to-quic/

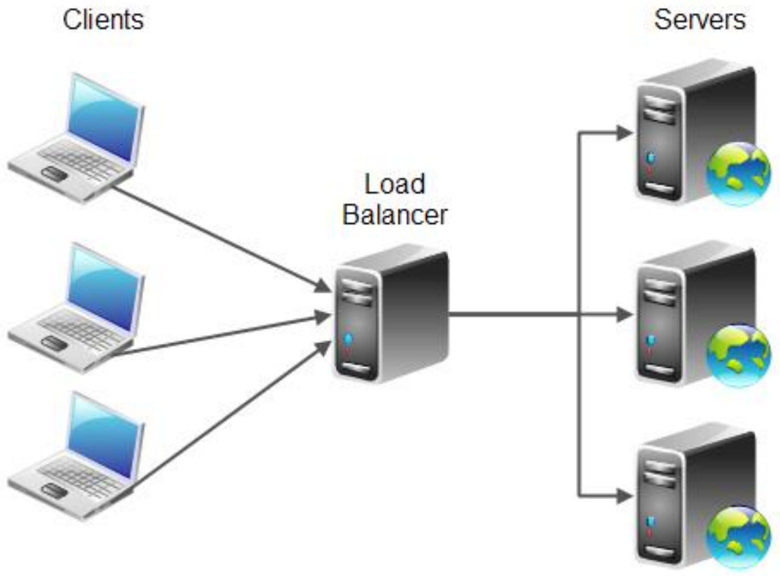
2. ConnectionID: Asymmetric for Routing/Load balancing

Originally: just 1 ConnectionID

Later: source vs destination

Want server to choose CID, to use for routing/load balancing

- Need traffic to go to same origin server
- Put routing info inside the CID for stateless load balancers



Statefull:

Remember that 0xabcdefab goes to server 1

Stateless:

0xabcdefa2 always goes to server 2

→ Client doesn't know this

→ Server 2 has to choose its own CID

https://datatracker.ietf.org/doc/draft-duke-quic-load-balancers/

image: http://tutorials.jenkov.com/software-architecture/load-balancing.html

2. ConnectionID: Client chooses for server, server overrides

```
RX Initial(0x0), Version: 0xff00000f, Dest CID: 0x1156a3c691c16e8c390499d90aac254b21, Src CID: 0x0736553ca2daa8
     PKN: 0, payload length: 1184
  CRYPTO (0x18)
  length=304 offset=0
  data=SSL3_MT_CLIENT_HELLO (PROBABLY, but could be different or more if split or coalesced packet)
  PADDING (0x0)
  length= 859
```

```
TX Handshake(0x2), Version: 0xff00000f, Dest CID: 0x0736553ca2daa8, Src CID: 0x0f72f17fbaab34b215b0990e570661
     PKN: 0, payload length: 1220
  CRYPTO (0x18)
  length=1200 offset=0
  data=SSL3_MT_ENCRYPTED_EXTENSIONS (PROBABLY, but could be different or more if split or coalesced packet)
```

2. ConnectionID: Generic metadata storage field

Load balancer wants to support changing CIDs

→ Same routing info should be encoded in different CIDs

→ Cannot do this for just random values

→ CID is routing metadata but encrypted!

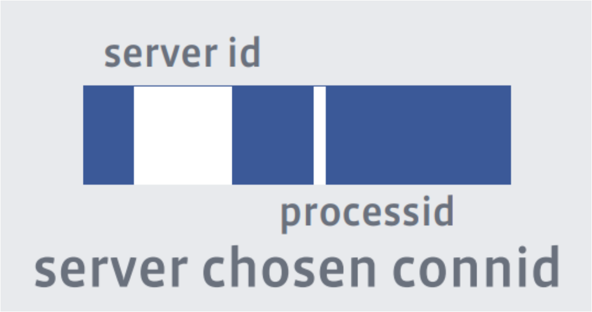
→ Metadata can be large, so CID up to 18 bytes!

First CID:
- Decrypted: 0xabcdefa2
- Encrypted: 0x12345678

After CID change:
- Decrypted: 0xaabbccd2
- Encrypted: 0x87654321

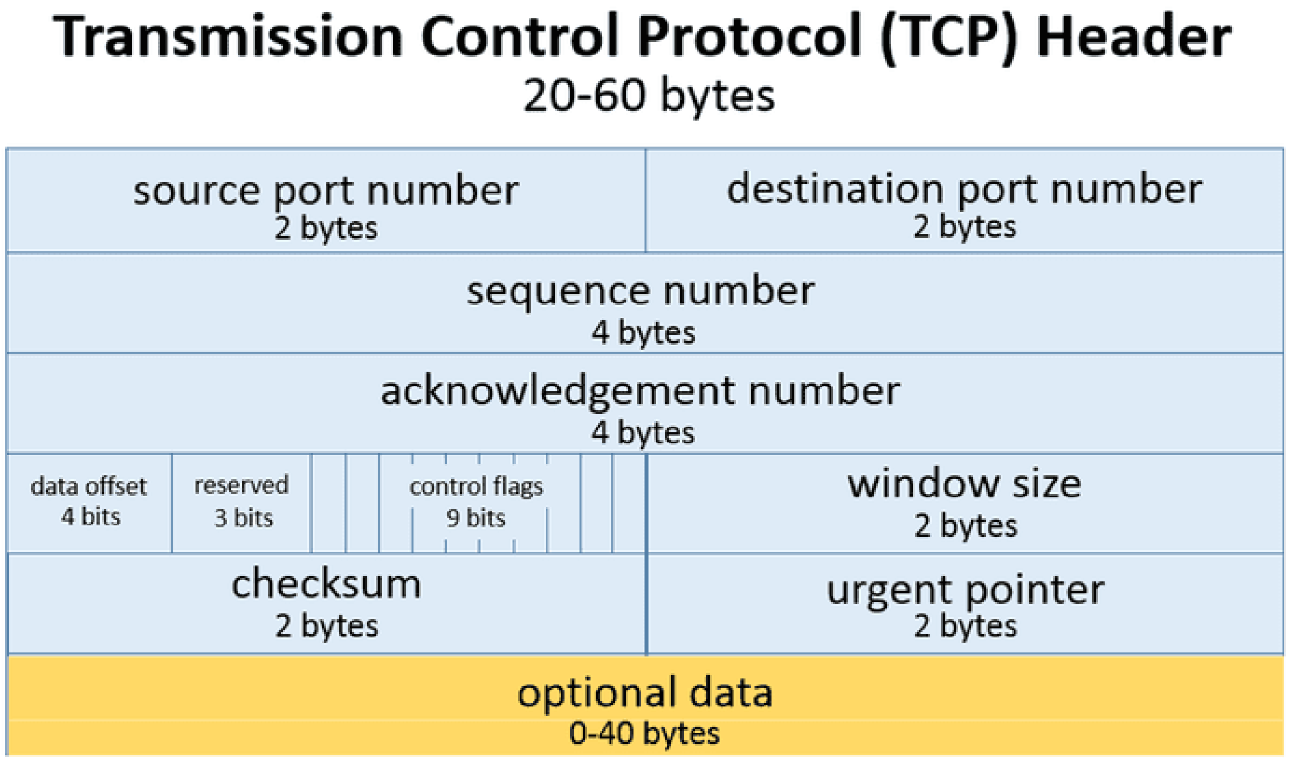
Facebook also encodes process ID

→ Seamless handover on server upgrade



https://datatracker.ietf.org/doc/draft-duke-quic-load-balancers/
https://conferences2.sigcomm.org/co-next/2018/slides/epiq-keynote.pdf
https://www.youtube.com/watch?v=8lYHNzoPS2o

3. Minimize overhead



- Fields are always the same size (even if values are small)
- Good for fast processing, bad for minimizing overhead
- 1st example was Short header, there are many more!

Image: https://www.lifewire.com/tcp-headers-and-udp-headers-explained-817970

3. Minimize overhead: QUIC saves on bits like crazy!

- Variable Length Integer Encoding + optional fields

2Bit	Length	Usable Bits	Range
00	1	6	0-63
01	2	14	0-16383
10	4	30	0-1073741823
11	8	62	0-4611686018427387903

Table 4: Summary of Integer Encodings

(i) = VLIE field
[…] = optional field

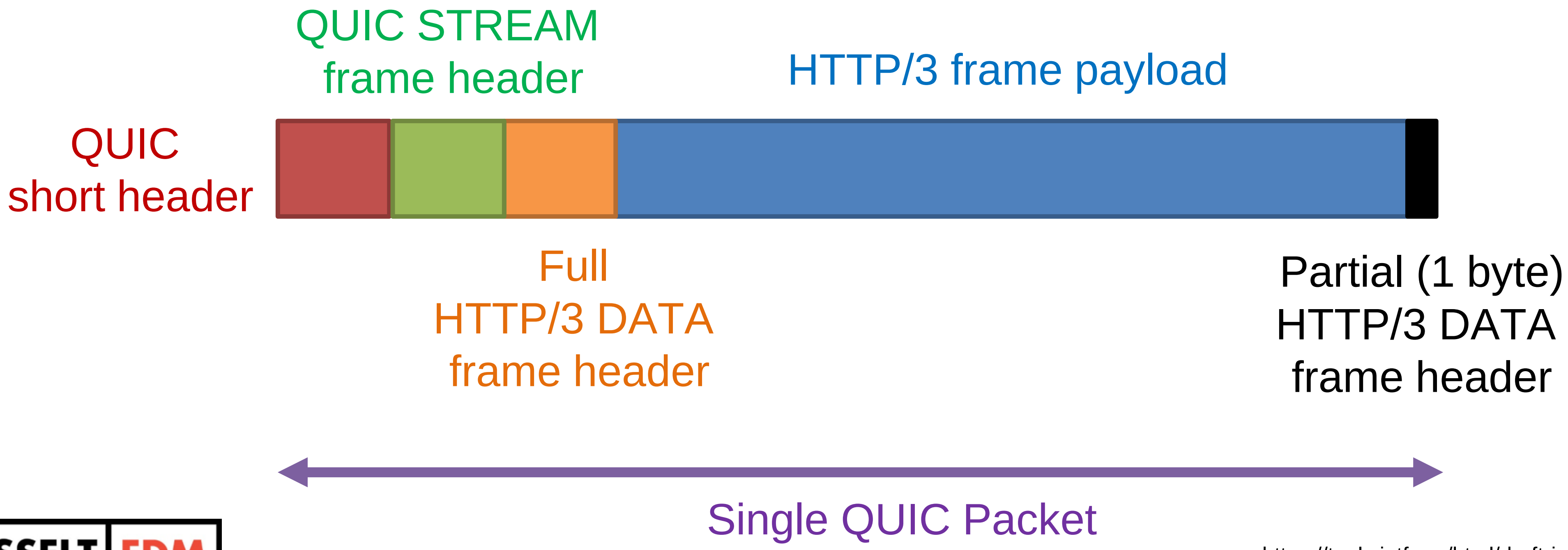
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Stream ID (i)                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         [Offset (i)]                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         [Length (i)]                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


https://tools.ietf.org/html/draft-ietf-quic-transport-19#section-16

3. Minimize overhead: Parsing problems (H3 frames can cross multiple packets)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Type (i)                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Length (i)                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Frame Payload (*)                     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: HTTP/3 frame format



https://tools.ietf.org/html/draft-ietf-quic-transport-19#section-16

3. Minimize overhead: But not always in the same way…

- Not everything is VLIE though

```
|DCIL(4)|SCIL(4)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Destination Connection ID (0/32..144)         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Source Connection ID (0/32..144)            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- DCIL : Destination Connection ID Length
  - Faster parsing (e.g., routing, load balancing)
- DCIL = 0 : no DCID set
- For all other values: have to do +3!

0000 = 0 bytes
0001 = 4 bytes
0010 = 5 bytes
…
1111 = 18 bytes

https://tools.ietf.org/html/draft-ietf-quic-transport-19#section-16

3. Minimize overhead: It gets worse

- Packet numbers are ~delta-encoded
  - Even large packet numbers use only a small amount of bytes
    - More packets on the wire = more bytes needed
    - Typically 1 or 2 is enough!
  - Done by "delta-encoding" based on the largest acknowledged packet
  - Complex algorithm that allows leaving out several least-significant bits as well

- Packet number length in 2 flag bits (used to be VLIE though)
  - 2 bits, number always at least 1 byte long, so always +1:
    - 00 = 1 byte, 01 = 2 bytes, 10 = 3 bytes, 11 = 4 bytes

https://tools.ietf.org/html/draft-ietf-quic-transport-19#appendix-A

3. Minimize overhead: yet another trick up my sleeve

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Largest Acknowledged (i)                ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      First ACK Block (i)                    ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Gap (i)                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Additional ACK Block (i)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Gap (i)                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Additional ACK Block (i)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Gap (i)                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Additional ACK Block (i)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: ACK Block Section

Full packet number

Not full numbers, but COUNTS, relative to the previous values

- Counts stay very small

"The number of packets in the gap is one higher than the encoded value of the Gap Field"

Most of these will fit in 1 byte each

https://tools.ietf.org/html/draft-ietf-quic-transport-19#appendix-A

3. Minimize overhead: yet another trick up my sleeve: Reprise

- ACK delay: measured in microseconds
  - Indicates the processing delay to generate the ACK
  - Allows better estimates for network delay (total – processing = network)
  - Maybe not everyone needs this high resolution…

  - Value in the packet needs to be multiplied by 2^ack_delay_exponent
    - Peer can indicate ack_delay_exponent in transport parameters
    - Default: 2^3

  - E.g., delay was 400 microseconds, value on the wire is 50
    - Anything lower than 8 microseconds cannot be encoded

https://tools.ietf.org/html/draft-ietf-quic-transport-19#appendix-A

4. HTTP/3 vs HTTP/2 : Some simplification

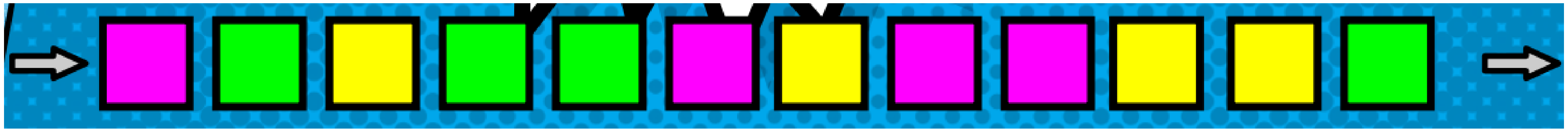
- QUIC takes over several things that were in H2
  - Flow Control
  - Streams and stream management
  - Padding frames

- Fun fact: without HTTP/3 implementations, QUIC was tested with HTTP/0.9

```
RX Protected1RTT(0x5), Dest CID: 0x0f72f17fbaab34b215b0990e570661
     PKN: 0, spinbit: 0
  STREAM (0x17)  FIN=1 LEN=1 OFF=1
  StreamID (0x00) length=16 offset=0
    474554202f696e6465782e68746d6c0a          'GET /index.html\n'
```

https://tools.ietf.org/html/draft-ietf-quic-transport-19
https://tools.ietf.org/html/draft-ietf-quic-http-19
https://github.com/rmarx/quicker

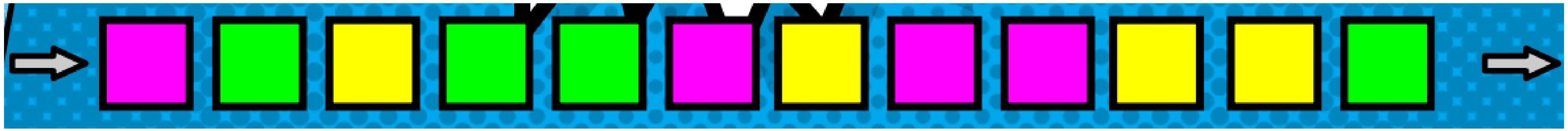
4. HTTP/3 vs HTTP/2 : Head-of-line blocking



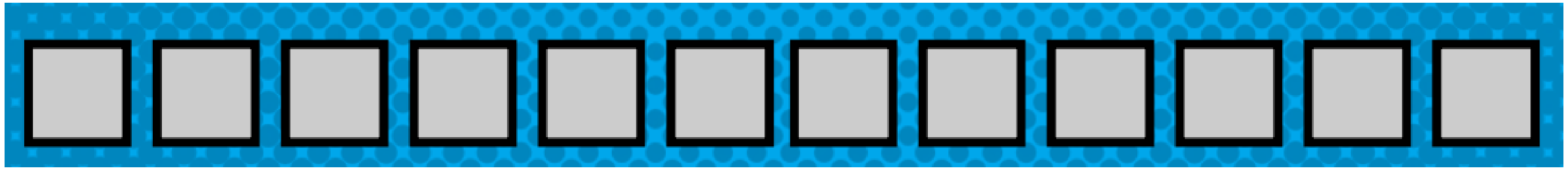
HTTP/2 multiplexes multiple streams on 1 TCP connection

https://www.youtube.com/watch?v=B1SQFjIXJtc
http://internetonmars.org/deltav.pdf

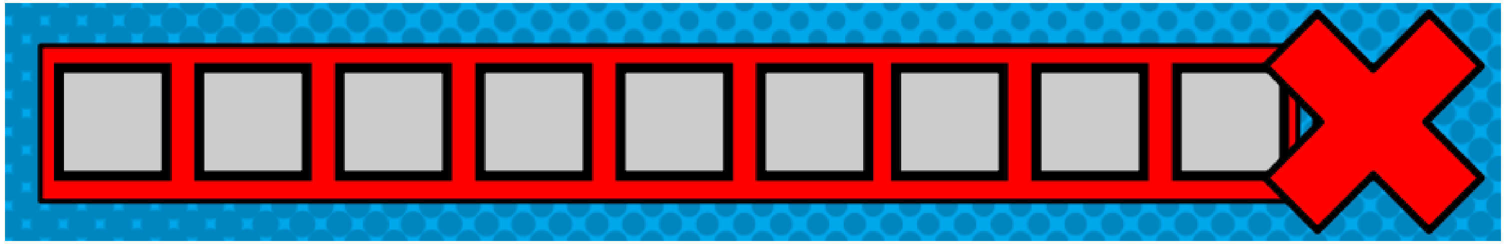
4. HTTP/3 vs HTTP/2 : Head-of-line blocking



HTTP/2 multiplexes multiple streams on 1 TCP connection



Sadly, TCP doesn't know this…

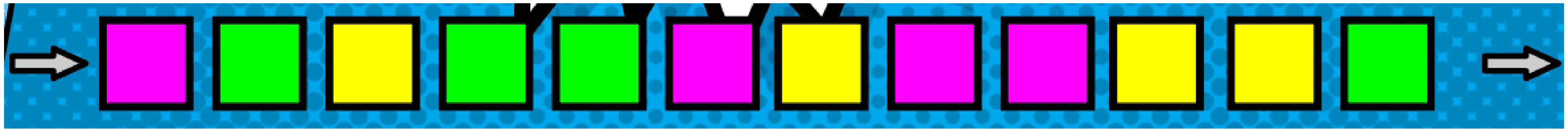


When 1 packet is lost, all the rest needs to wait

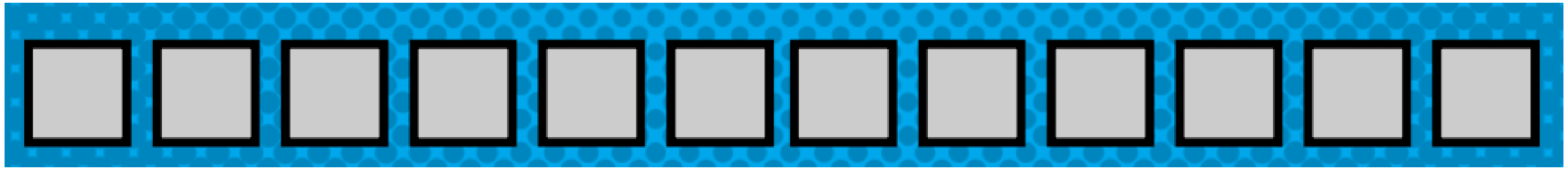
UHASSELT EDM

https://www.youtube.com/watch?v=B1SQFjIXJtc
http://internetonmars.org/deltav.pdf

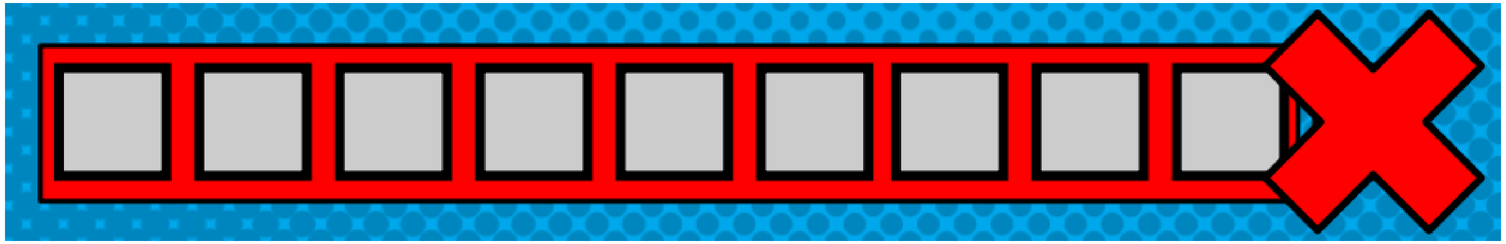
4. HTTP/3 vs HTTP/2 : Head-of-line blocking



HTTP/2 multiplexes multiple streams on 1 TCP connection



Sadly, TCP doesn’t know this…



When 1 packet is lost, all the rest needs to wait

QUIC allows the rest to pass

https://www.youtube.com/watch?v=B1SQFjIXJtc
http://internetonmars.org/deltav.pdf

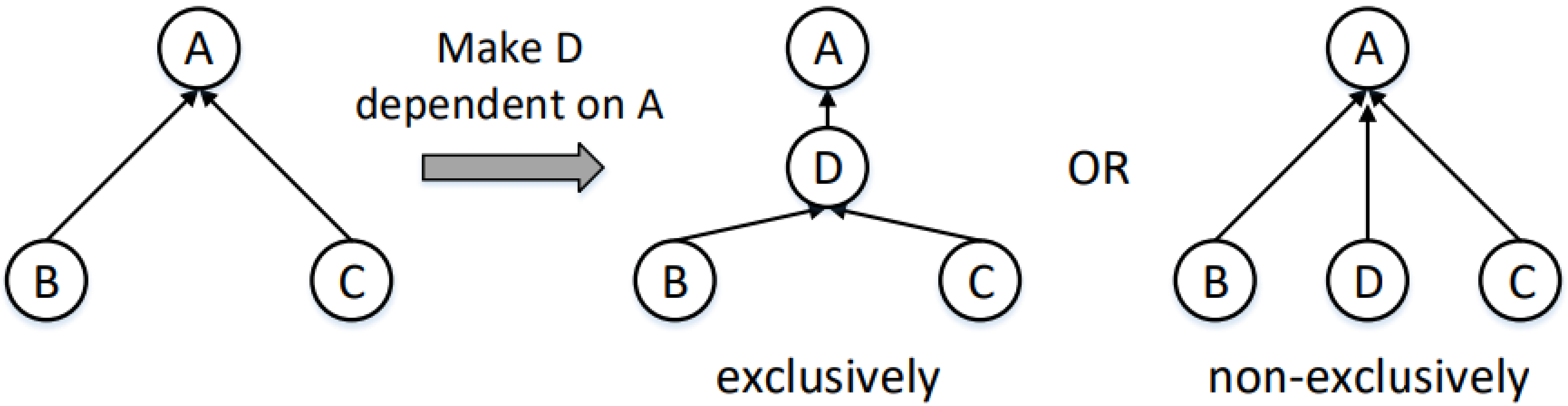
4. HTTP/3 vs HTTP/2 : Head-of-line Blocking

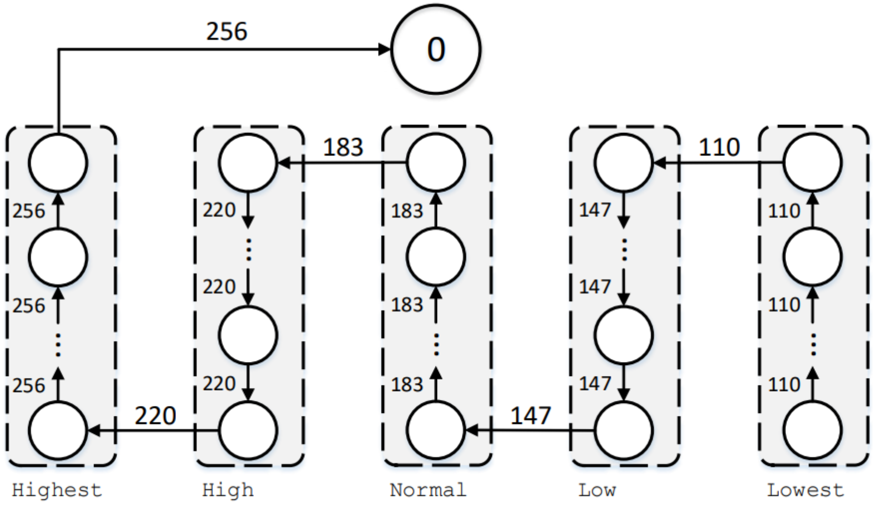
- QUIC knows about (HTTP/3) streams
- Processes them independently
- Means there is no strict ordering between streams anymore!

- In HTTP/2: arrives in the order you put it on the wire, even if on different streams
- In HTTP/3: arrives ordered per-stream, but not across streams
  - E.g.,
    if packet 1 for stream A, which was sent first, is lost,
    packet 2 for stream B, which was sent second, will arrive before retransmit of 1

https://tools.ietf.org/html/draft-ietf-quic-transport-19
https://tools.ietf.org/html/draft-ietf-quic-http-19

4. HTTP/3 vs HTTP/2 : Exclusive priorities





What if you add 2 nodes
exclusively to A at the same time?

https://tools.ietf.org/html/draft-ietf-quic-transport-19
https://tools.ietf.org/html/draft-ietf-quic-http-19
https://speeder.edm.uhasselt.be/www18/files/h2priorities_mwijnants_www2018.pdf

4. HTTP/3 vs HTTP/2 : Exclusive priorities

What if you add 2 nodes B and C
exclusively to A at the same time?

```
A          A          A
           |          |
           B          C
           |          |
           C          B
```

C arrived first,
then B

B arrived first,
then C

“Solution” 1: No more exclusive priorities in HTTP/3

“Solution” 2: Send priority updates on 1 “control stream”

(only updates, not the initial, because HOL blocking!)

https://tools.ietf.org/html/draft-ietf-quic-transport-19
https://tools.ietf.org/html/draft-ietf-quic-http-19
https://speeder.edm.uhasselt.be/www18/files/h2priorities_mwijnants_www2018.pdf
https://github.com/quicwg/base-drafts/issues/2502

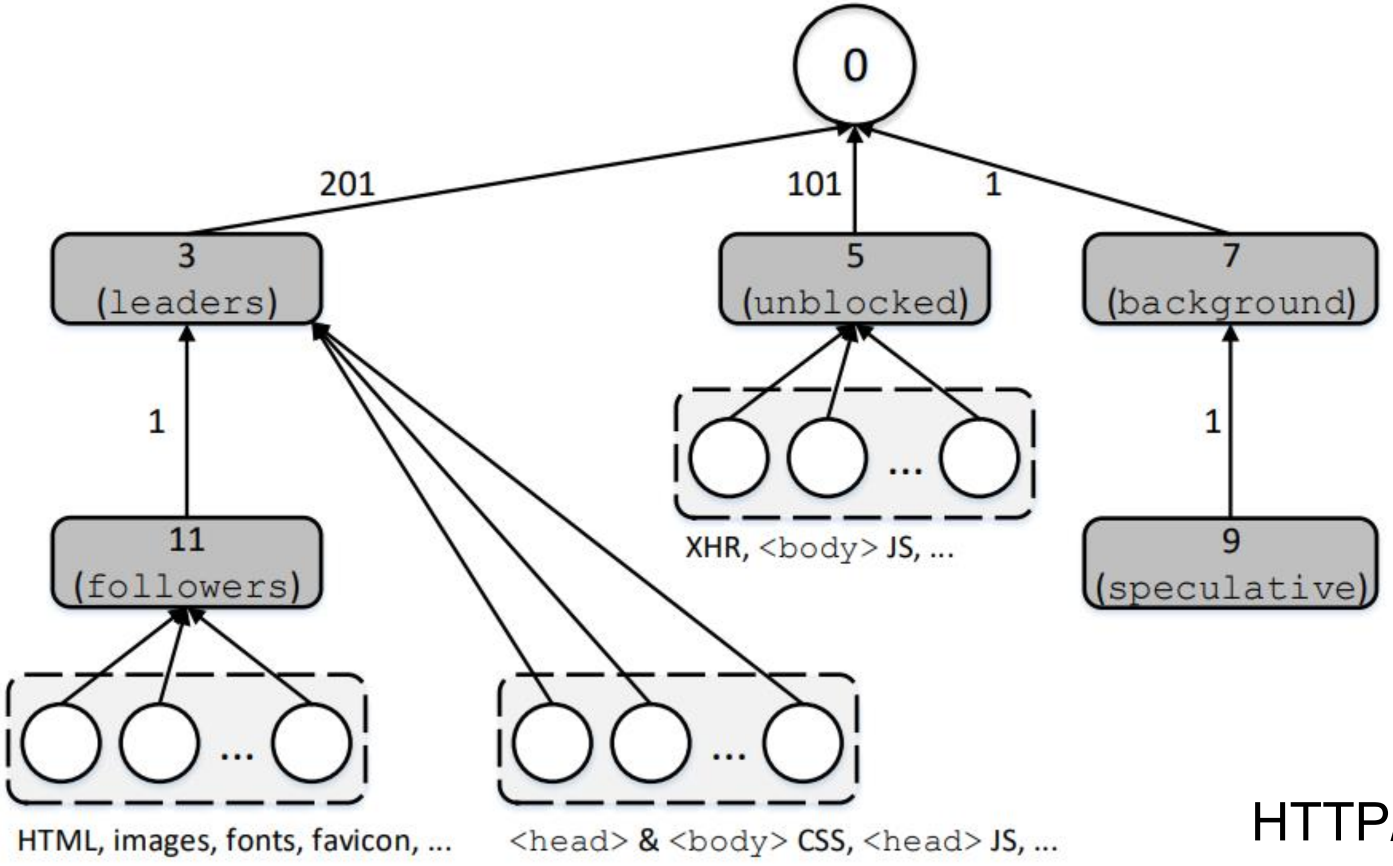
4. HTTP/3 vs HTTP/2 : QPACK vs HPACK

- HTTP header compression
- HPACK: just send encoding information with the header
- QPACK:
  - Separate encoder and decoder streams
  - Either accept HOL-blocking, or keep sending literals until confirmed

  - Cannot just send with header, because other streams might start using encoded value and arrive before the encoding info…

https://tools.ietf.org/html/draft-ietf-quic-transport-19
https://tools.ietf.org/html/draft-ietf-quic-http-19
https://tools.ietf.org/html/draft-ietf-quic-qpack-07

4. HTTP/3 vs HTTP/2 : Priorities (again)



HTTP/2: “fake” streams as grouping nodes
HTTP/3: explicit “placeholder” support built-in

https://tools.ietf.org/html/draft-ietf-quic-transport-19
https://speeder.edm.uhasselt.be/www18/files/h2priorities_mwijnants_www2018.pdf
https://tools.ietf.org/html/draft-ietf-quic-http-19

4. HTTP/3 vs HTTP/2 : Push

- PUSH_PROMISE
  - HTTP/2: STREAM_ID
  - HTTP/3: PUSH_ID (more flexible)

- DUPLICATE_PUSH (again: saves on bytes)
  - Due to reordering, DUPLICATE_PUSH frames can arrive before the corresponding PUSH_PROMISE frame

https://tools.ietf.org/html/draft-ietf-quic-transport-19
https://tools.ietf.org/html/draft-ietf-quic-http-19
https://github.com/quicwg/base-drafts/pull/2527

5. Additional stuff I could talk about

- Max_uni/bidi transport parameters + asymmetric streams
- Flow control
  - Connection vs stream-level
- Recovery
  - Combination of various best practices for loss detection
  - Pluggable congestion control

- Linkability
- Prioritization and buffer bloat
- QUIC standardized logging and debugging tools
- …

https://tools.ietf.org/html/draft-ietf-quic-transport-19
https://tools.ietf.org/html/draft-ietf-quic-http-19