QUIC LEAGUE 2020

SAME SPORT, DIFFERENT STRATEGIES:
a study of QUIC and HTTP/3 game plan diversity

ROBIN MARX, JORIS HERBOTS,
WIM LAMOTTE AND PETER QUAX

HASSELT UNIVERSITY - EDM

TCP IS OUT, QUIC IS IN!

TCP 2U2 0P0

QUIC LEAGUE 2020

- Stream multiplexing
- User space congestion control
- 0-RTT
- Binary framing

MORE OPTIONS FOR INDIVIDUAL STACKS TO TWEAK BEHAVIOUR

QUIC EVOLUTION

QUIC EVOLUTION

And several others!

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

00

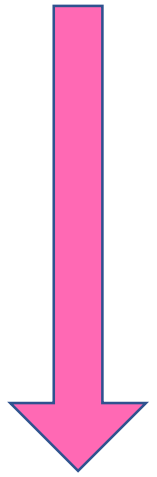
METHODOLOGY

ARE YOU WATCHING CLOSELY?

METHODOLOGY

structured endpoint logs

[qlog]



interactive tooling

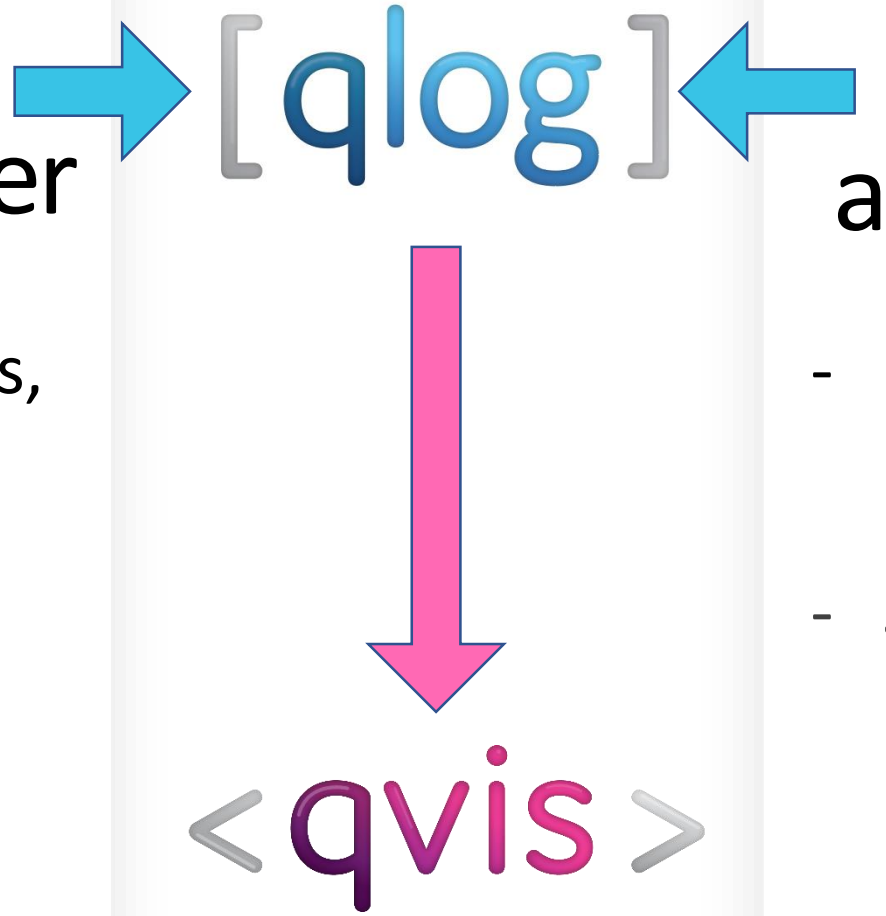
<qvis>

manual analysis

METHODOLOGY

QUIC interop runner

- Automated tests, run daily
- Client-side behaviour



Custom aioquic client

- Point at public interop endpoints
- Server-side behaviour

METHODOLOG

QUIC
interop runn

- Automated tes
  run daily
- Client-side
  behaviour

Verify results

1. Source code review
2. Ask the **original implementers**

Custom
oquic client

oint at public
nterop endpoints
erver-side
ehaviour

METHODOLOGY

18
teams

15
analyzed

At least
9
per test

01

FLOW CONTROL

DON’T MAKE THAT BUFFER SUFFER

FLOW CONTROL

TCP 2U 20 0P 0

Single connection-level buffer

RECEIVE WINDOW

QUIC LEAGUE 2020

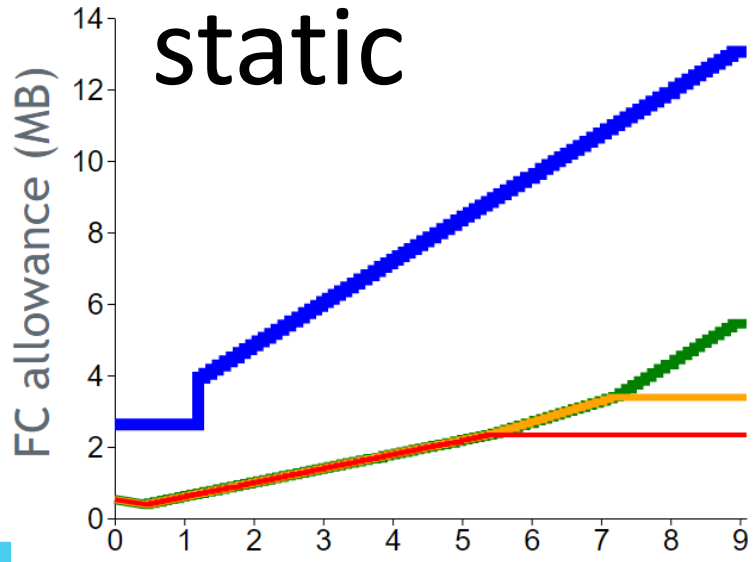
1. Connection-level limit
2. Stream-level limits
3. Stream count limit

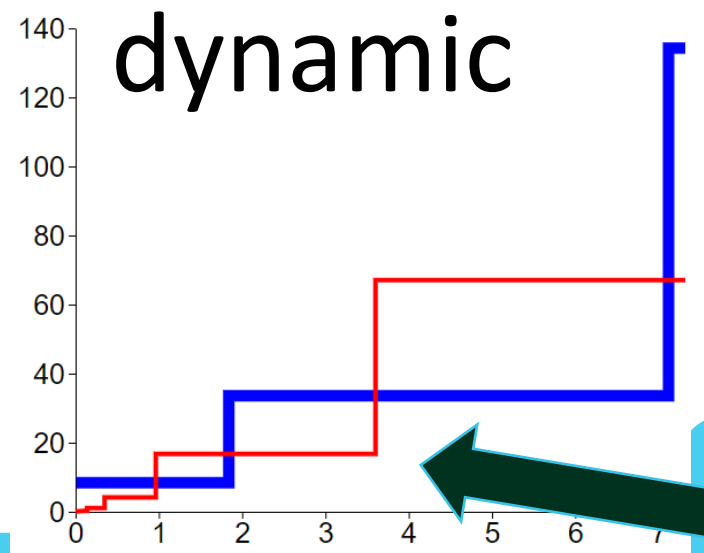
} increase often **OR** sending can STALL

3 MAIN FC APPROACHES

1. static: 5000 received, you get 5000 more
2. dynamic: 5000 received, you get **10000** more
3. autotune: fluctuate based on RTT/application behavior

TCP 2U2 0P0



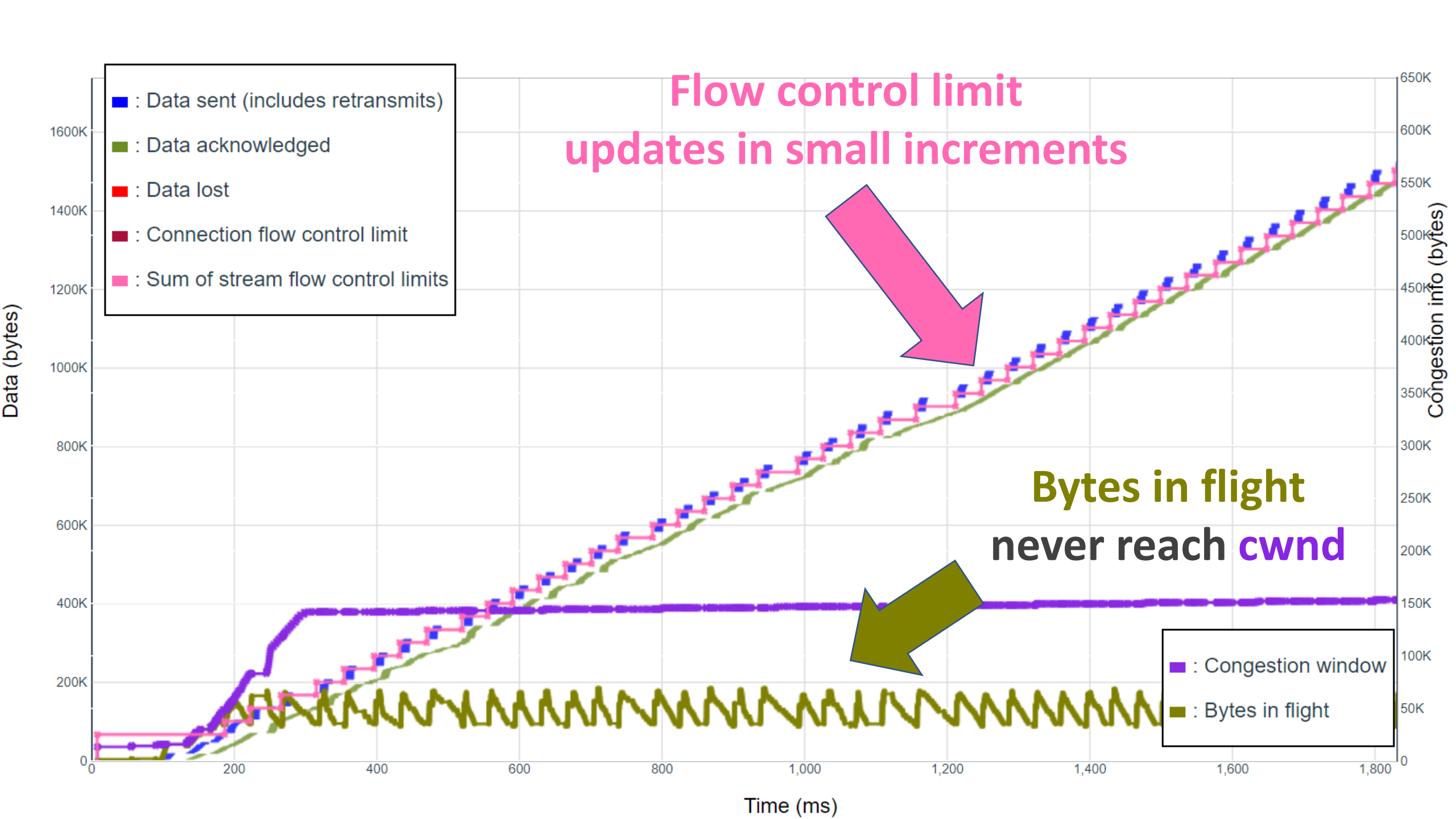


STALL

FLOW CONTROL

Flow control approach	Adoption
static	8/12
growing	3/12
autotune	1/12

“We have not yet spent time fine-tuning or testing Flow Control”

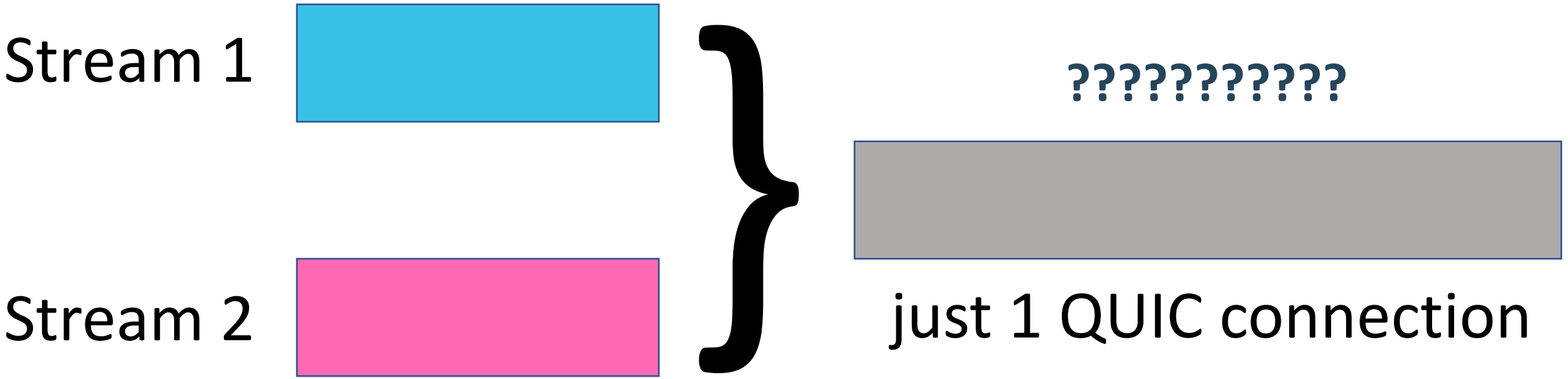

: Data sent (includes retransmits)
: Data acknowledged
: Data lost
: Connection flow control limit
: Sum of stream flow control limits
Flow control limit
updates in small increments
Bytes in flight
never reach cwnd
: Congestion window
: Bytes in flight
Data (bytes)
Congestion info (bytes)
Time (ms)
0
200
400
600
800
1,000
1,200
1,400
1,600
1,800
0
200K
400K
600K
800K
1000K
1200K
1400K
1600K
0
50K
100K
150K
200K
250K
300K
350K
400K
450K
500K
550K
600K
650K

MULTIPLEXING

THE SPICE MUST FLOW

02

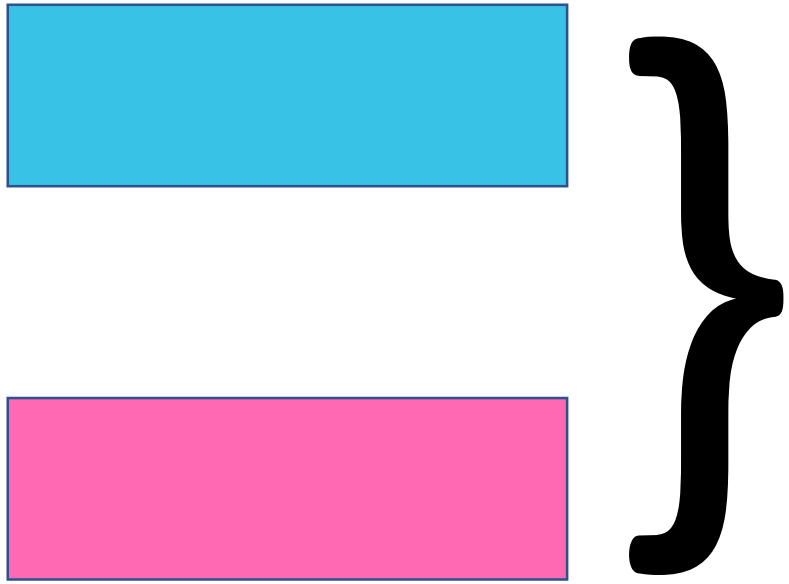
STREAM BANDWIDTH DISTRIBUTION



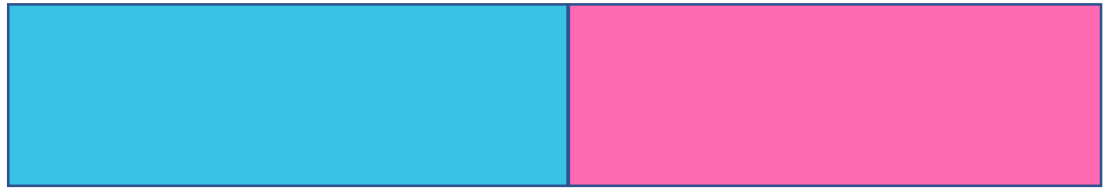
STREAM BANDWIDTH DISTRIBUTION

Stream 1

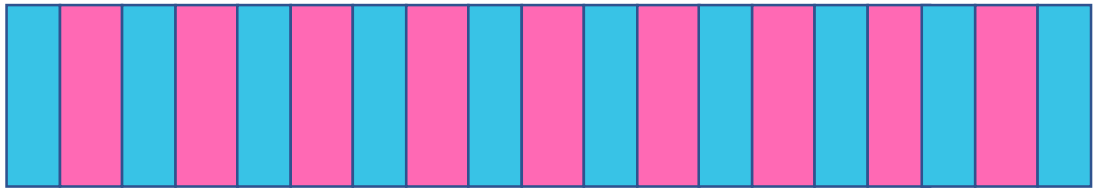
Stream 2



Sequential



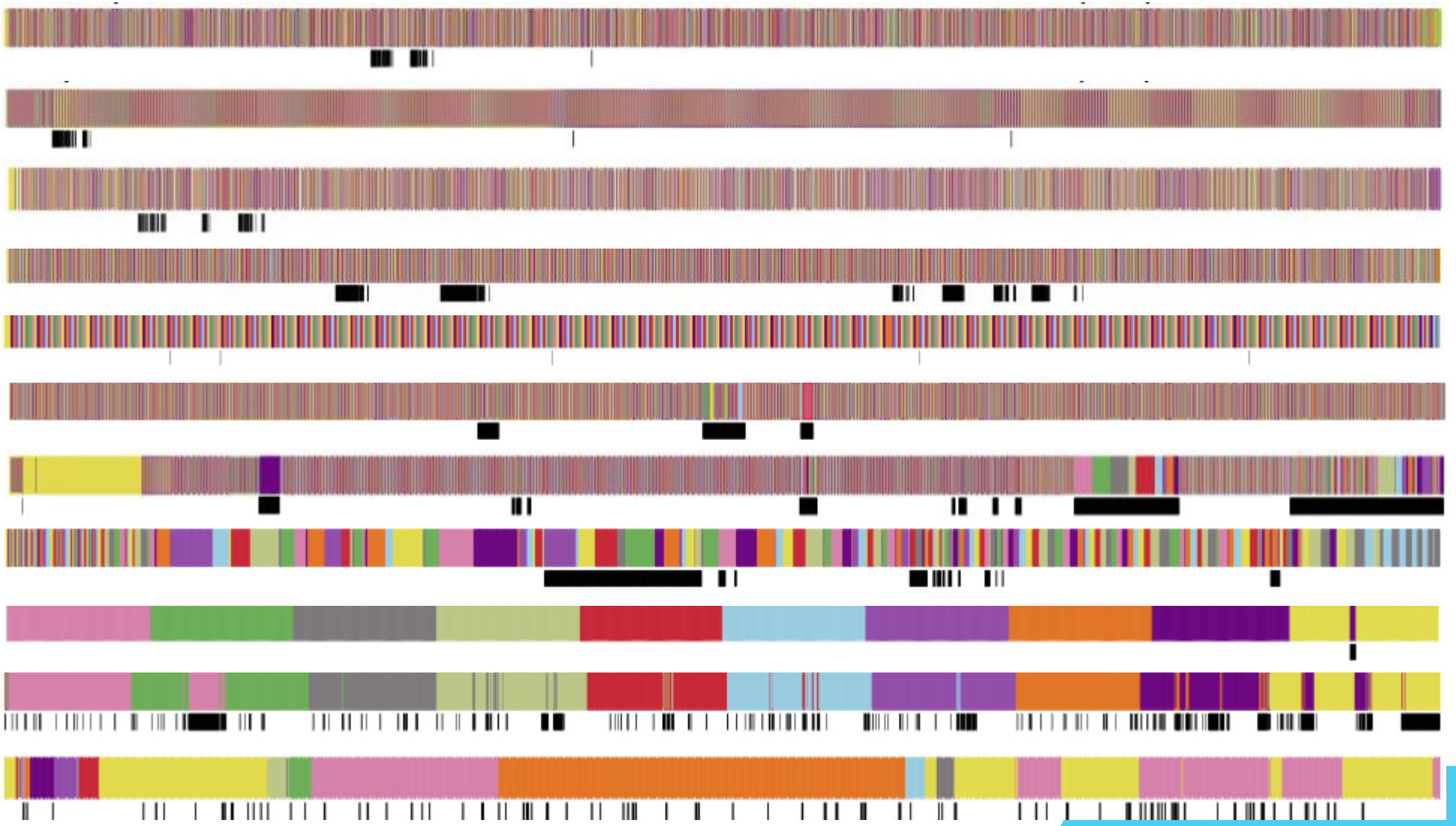
OR



Round-Robin

VARIETY IS THE SPICE OF LIFE

Round-Robin



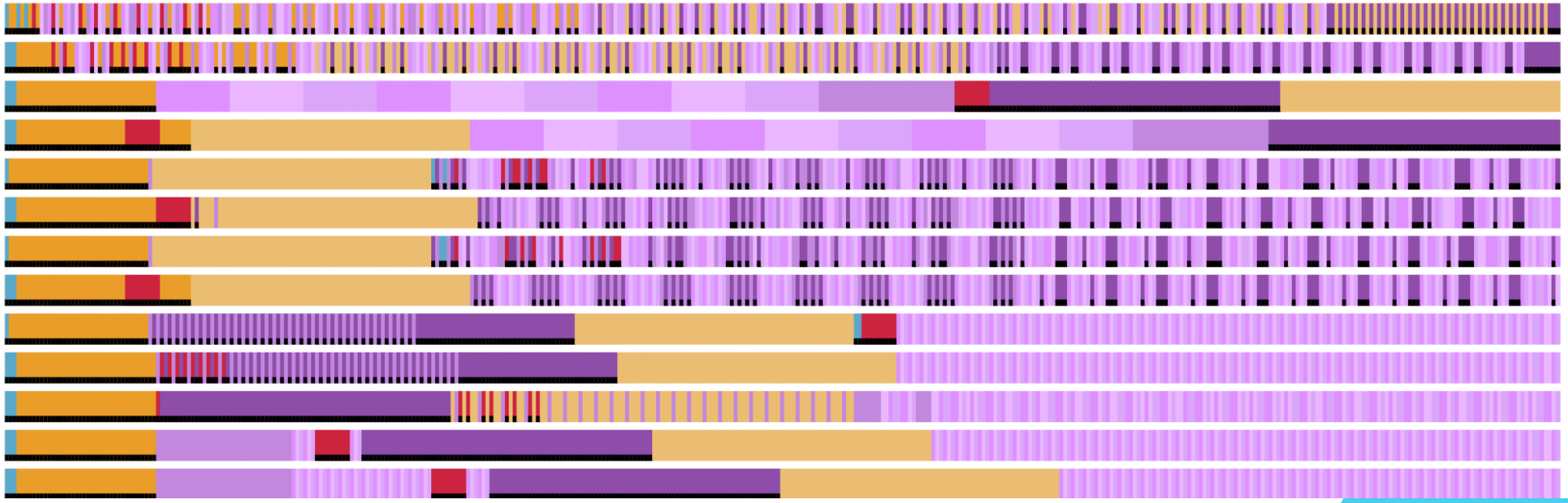
The weird ones in the middle

Sequential

VARIETY IS THE SPICE OF LIFE

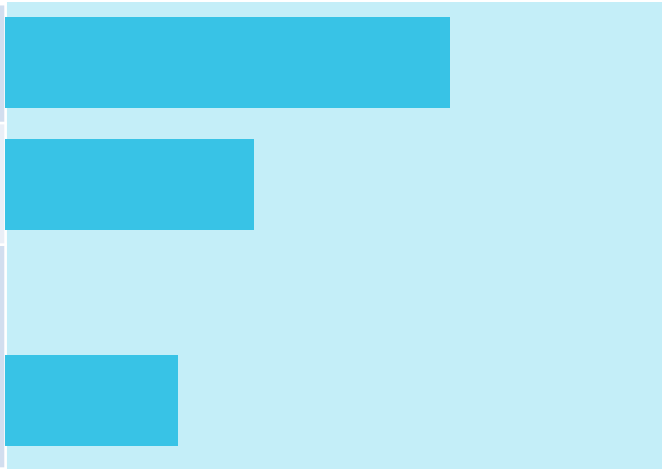
HTTP/3 INVITATIONAL

- Server Push
- Header Compression
- **Prioritization**



QUIC MULTIPLEXING

Multiplexer	Adoption
Round-Robin	9/13
Sequential	4/13
(experimental) HTTP/3 prioritization	5/18



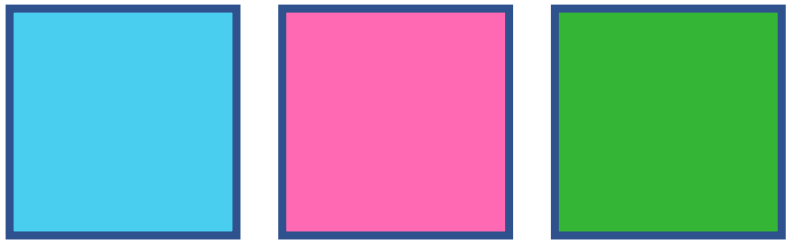
“waiting for HTTP/3 prioritization to fine-tune”

https://tools.ietf.org/html/draft-ietf-httpbis-priority-01

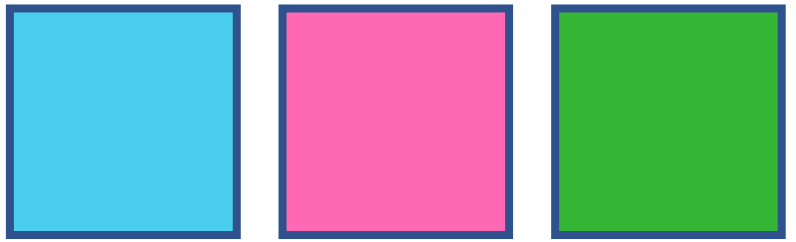
RETRANSMISSION SCHEDULING

TCP 2U2 0P0

send order



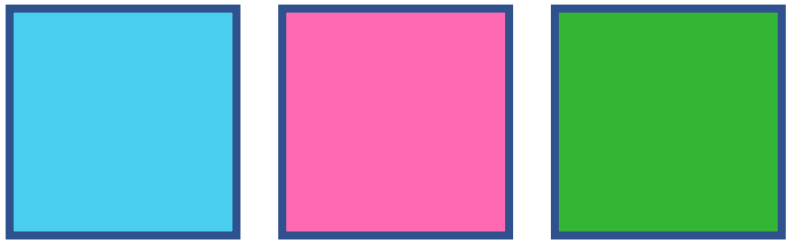
retransmission order



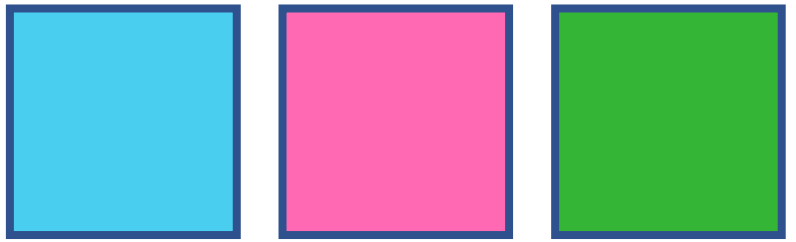
RETRANSMISSION SCHEDULING

TCP 2U 20P 20

send order

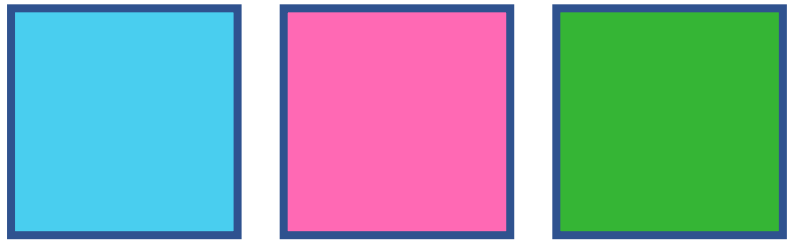


retransmission order

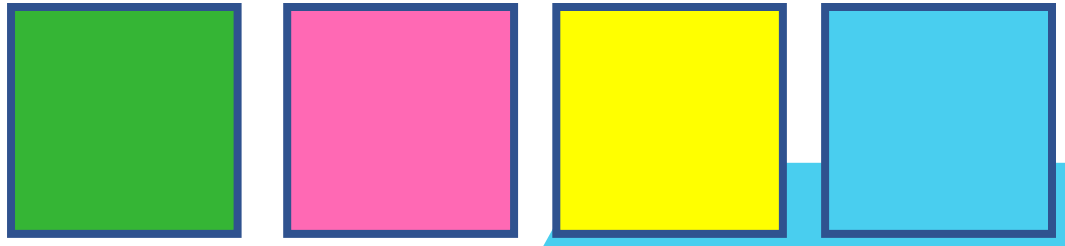


QUIC LEAGUE 2020

send order



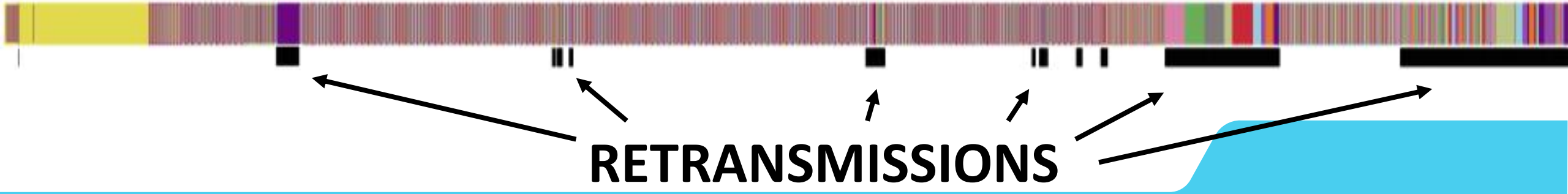
retransmission order



RETRANSMISSION SCHEDULING

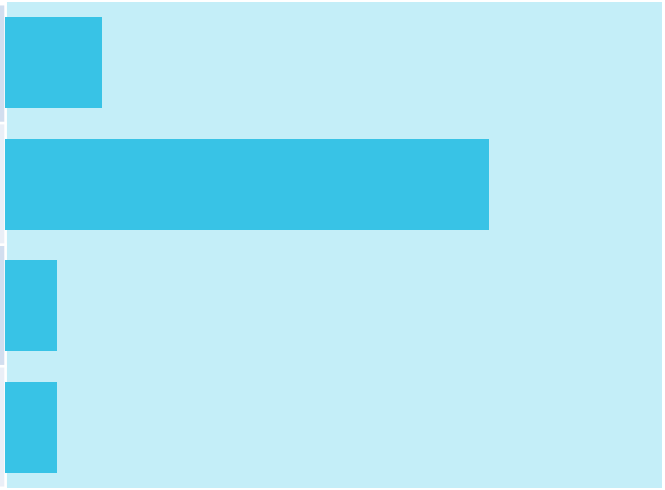
1. No special treatment
2. Highest priority, default scheduler
3. Highest priority, different scheduler
4. (HTTP/3) Prioritization-driven

Example for nr. 3:



RETRANSMISSION SCHEDULING

Retransmission approach	Adoption
1. All data is equal	2/13
2. TCP-alike	9/13
3. TCP-alike, change scheduler	1/13
4. Prioritization-driven HTTP/3	1/13



“Unclear which performs best/if it matters”

03

CONGESTION CONTROL

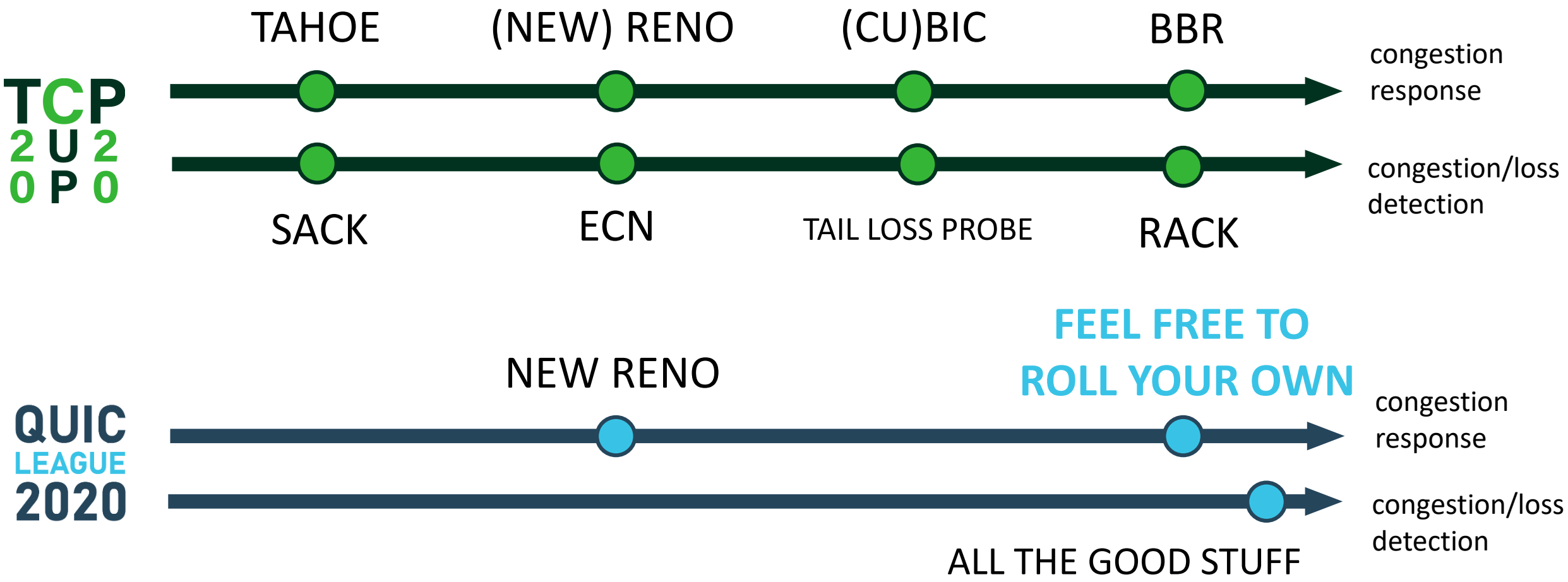
THE NEED FOR SPEED

CONGESTION CONTROL (MOST INACCURATE TIMELINE EVER)

TCP 2U2O P20

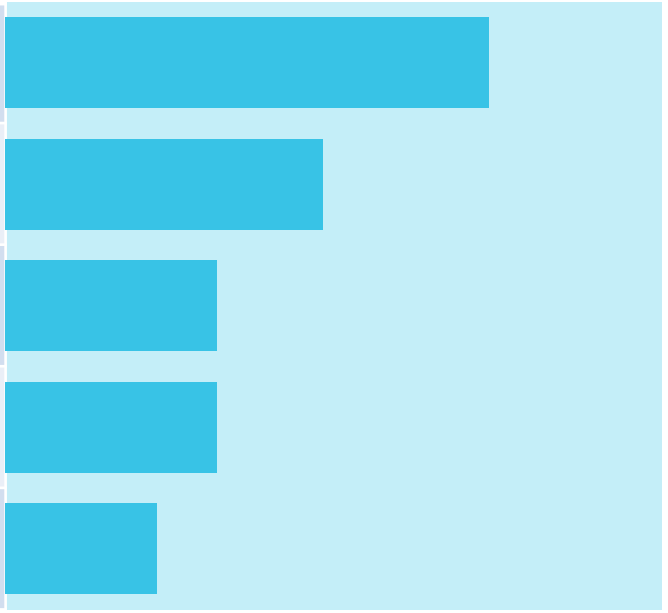
QUIC LEAGUE 2020

CONGESTION CONTROL (MOST INACCURATE TIMELINE EVER)



CONGESTION CONTROL

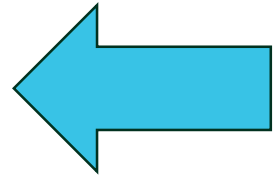
Congestion controller	Adoption
New Reno	9/15
CUBIC	6/15
→ (with hystart)	→ 4/6
BBR v1	4/15
BBR v2, COPA, …	3/15



“Often too complex to implement a new one”

CONGESTION CONTROL

```
congestion := congestion.NewCubicSender(
  congestion.DefaultClock{},
  rttStats,
  true, // use Reno
  tracer,
)
```

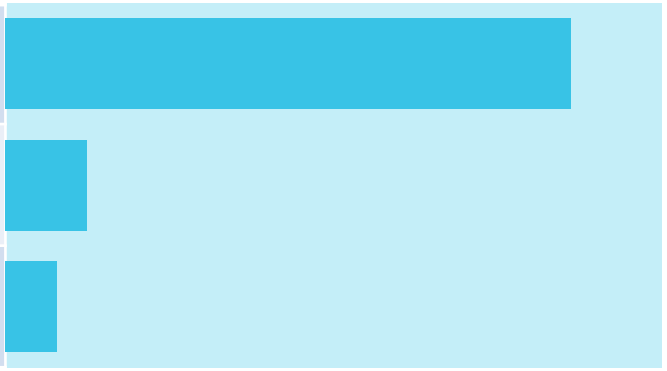


SNEAKY SNEAKY

https://github.com/lucas-clemente/quic-go/blob/a0607c8cad4493b802d7006abe9353be7ffc8ad2/internal/ackhandler/sent_packet_handler.go#L102

THE DEVIL IS IN THE DETAILS

Initial congestion window	Adoption
12-15 kB	11/14
40+ kB	3/14
smart tweaking	2/14



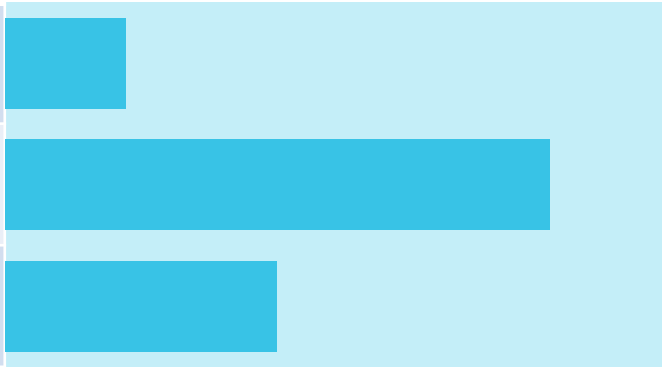
“We just looked at what Google was doing”

Pacing	Adoption
Yes	8/15
No	7/15

“Complex to get right”

THE DEVIL IS IN THE DETAILS

ACK every X packets	Adoption
2	2/12
1 - 38	10/12
ACK frequency extension	4/12



“Read from socket in large batches, ACK per batch”

“Lower ACK frequencies are better on constrained networks”

https://tools.ietf.org/html/draft-iyengar-quic-delayed-ack-01, https://tools.ietf.org/html/draft-fairhurst-quic-ack-scaling-02

0-RTT

OFF-BY-ONE ERRORS ARE THE BEST

04

ROUND TRIPS ARE THE WORST

TCP
2 U 2
0 P 0

1. SYN/ACK
2. TLS
3. (TLS)
4. HTTP

QUIC
LEAGUE
2020

1. QUIC + TLS
2. HTTP

1. **QUIC + TLS + HTTP**

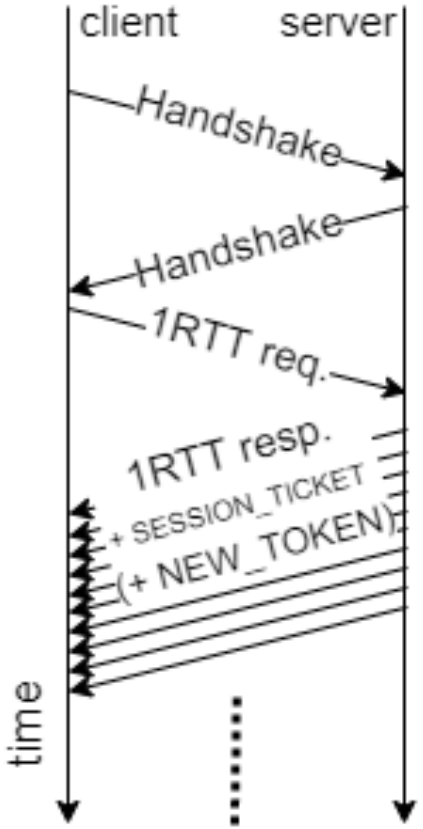
SOCCER HAS OFFSIDE, QUIC HAS 0-RTT

QUIC 0-RTT:

1. Needs to be encrypted → Only from second connection (session ticket)
2. Runs over IP + UDP → Send max **3X** as much as received (amplification limit)
3. Transports HTTP → Only idempotent requests

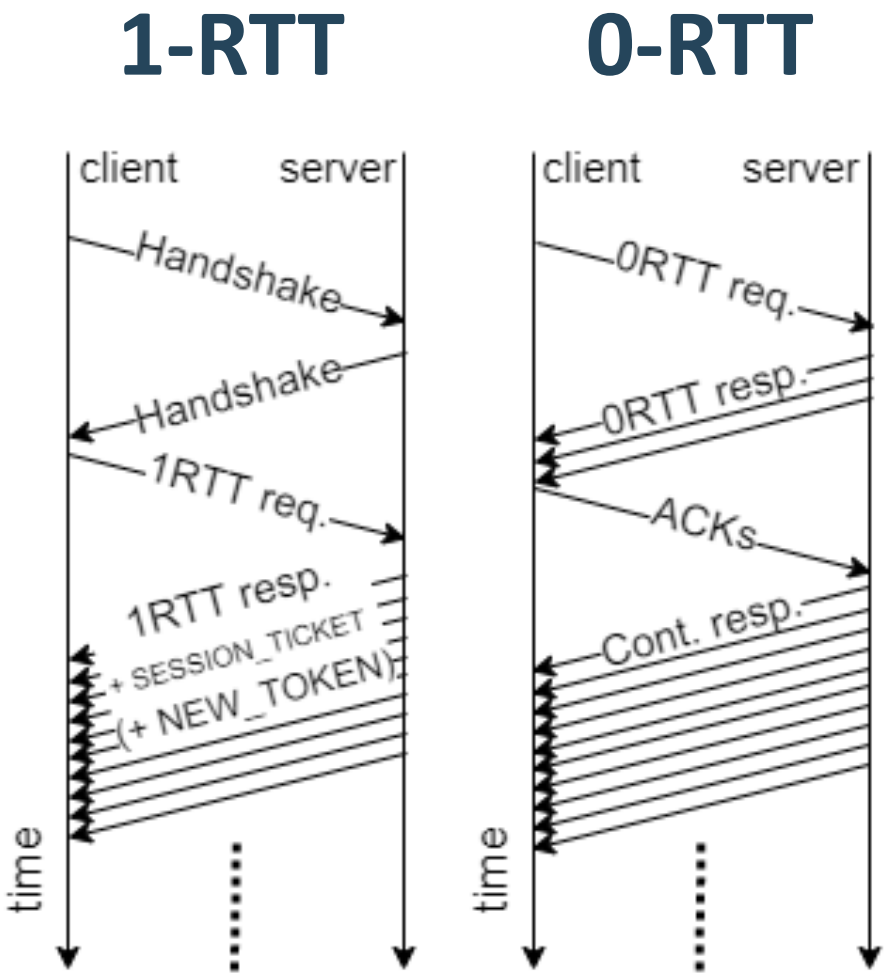
MAKING 0-RTT BETTER

1-RTT



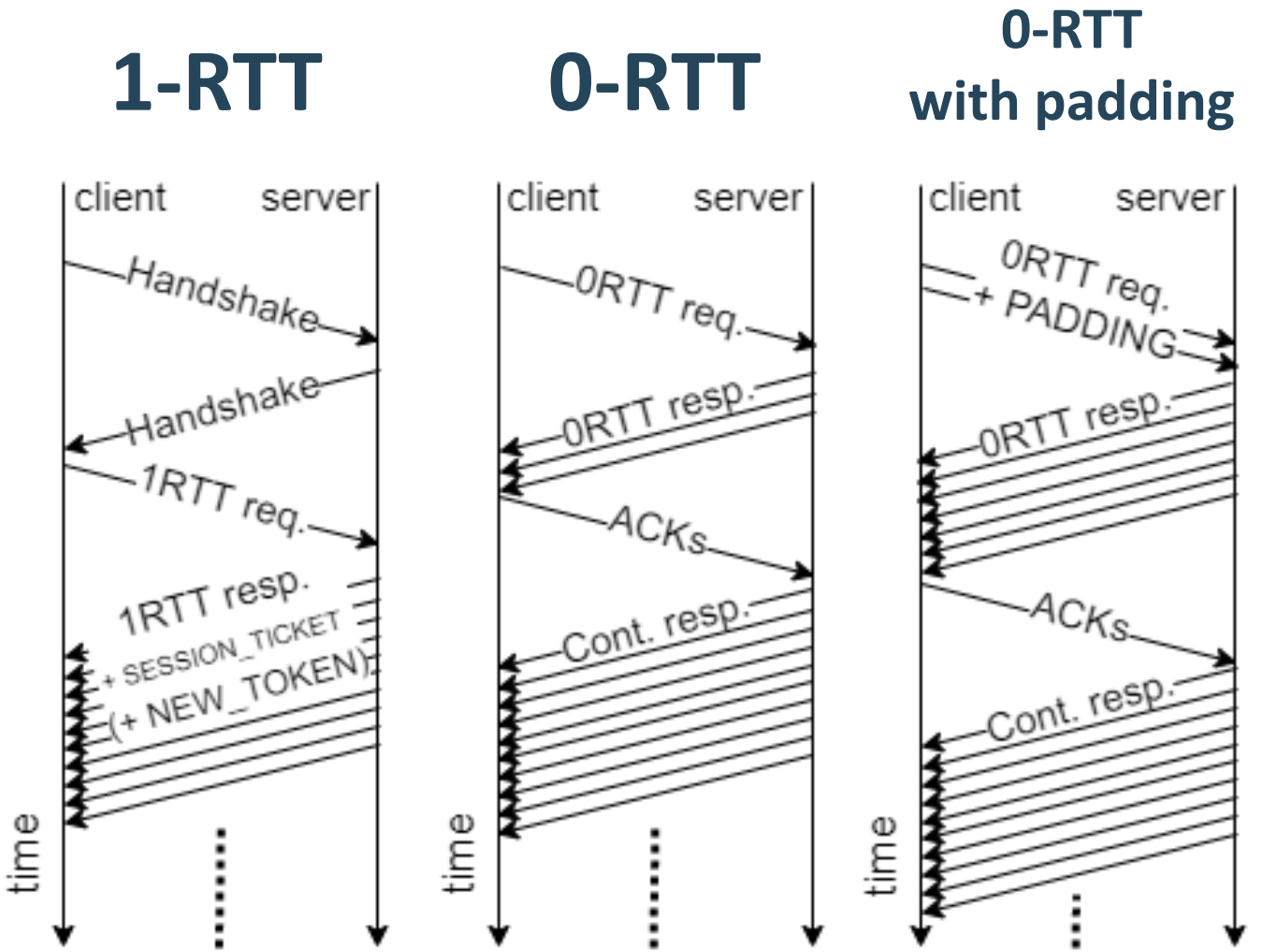
Session ticket enables 0-RTT for the **next** connection

MAKING 0-RTT BETTER



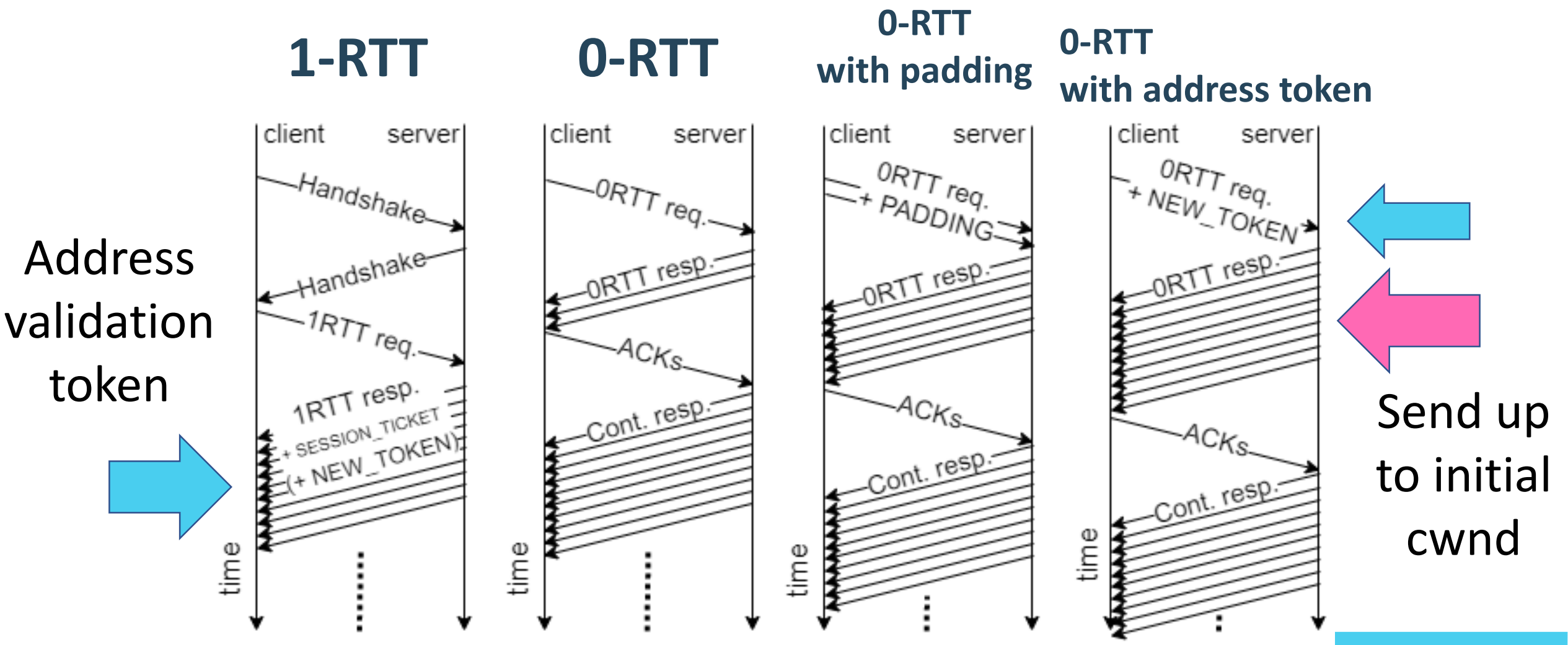
Server can only send **3X** what it received (say 4-5 kB)

MAKING 0-RTT BETTER



Client sends more, server can send more

MAKING 0-RTT BETTER



THE OPTIMIZATION FORMERLY KNOWN AS CRUCIAL

0-RTT support	Adoption
Yes	13/18
No	5/18

"TLS library doesn't support it yet"

Optimizations	Adoption
Extra PADDING	0/9
NEW_TOKEN	7/13

Amplification bugs	4/9

SET THE LASERS TO AMPLIFICATION

1. Ignore limit, have a 46kB init cwnd
   = **36X** amplification

2. Do not apply limit to retransmissions of 0-RTT data
   = **17X** amplification

3. Do not apply congestion control to 0-RTT data
   = **300kB burst if client sends 100kB**

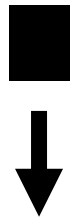
05

PACKETIZATION EFFICIENCY

THINKING OUTSIDE OF THE BOX

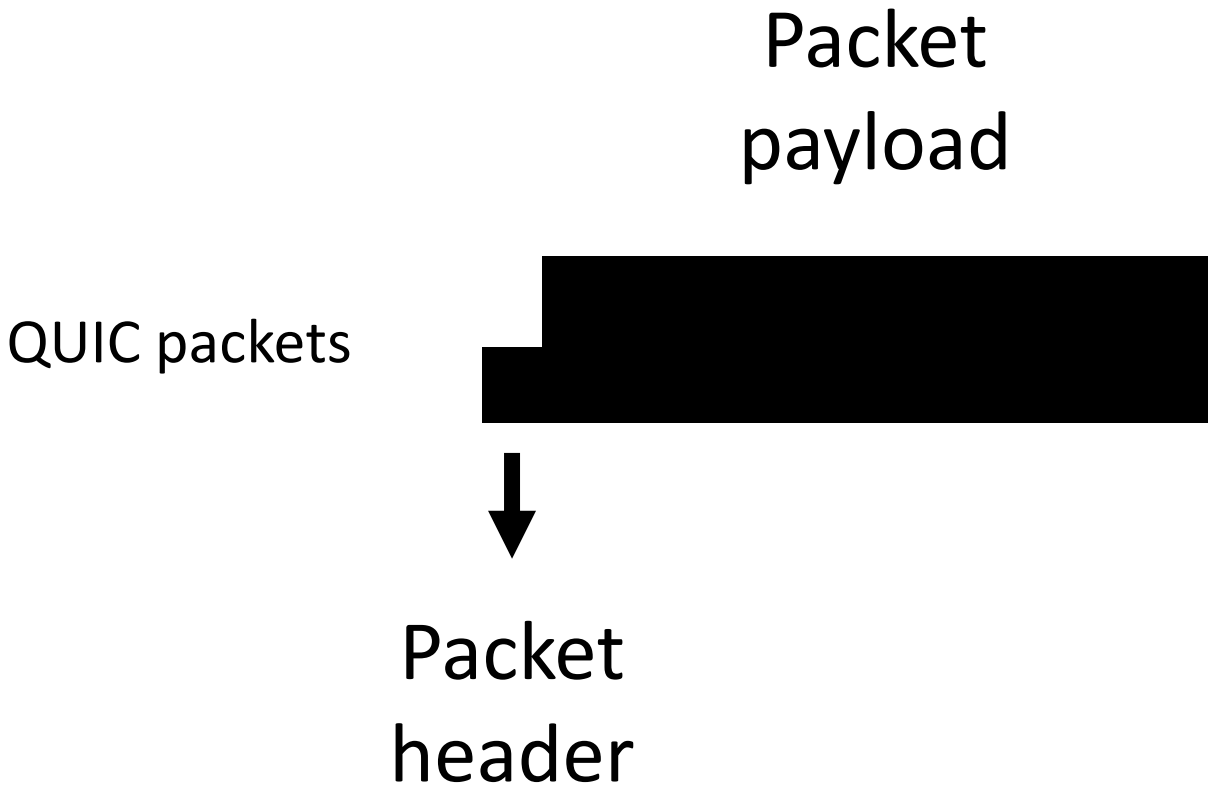
FRAMECEPTION

QUIC packets

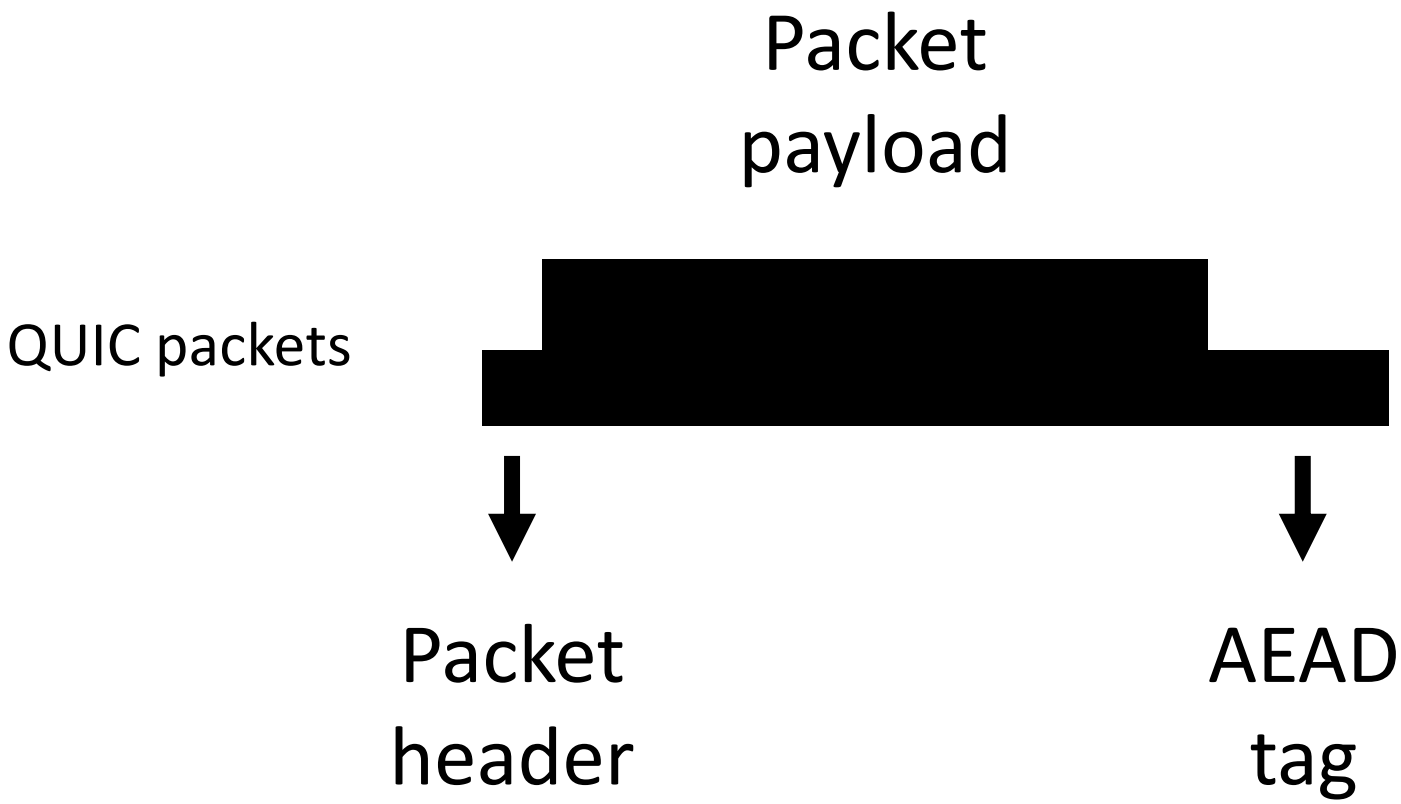


Packet header

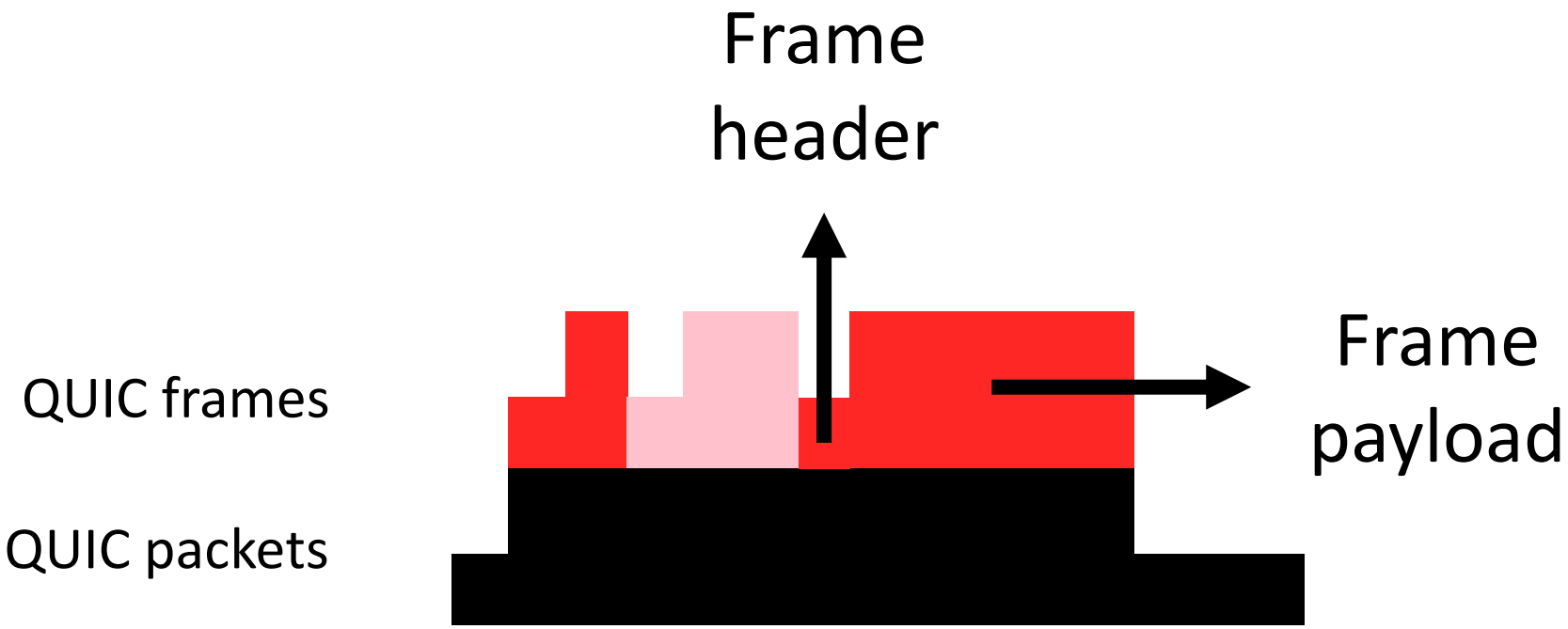
FRAMECEPTION



FRAMECEPTION



FRAMECEPTION

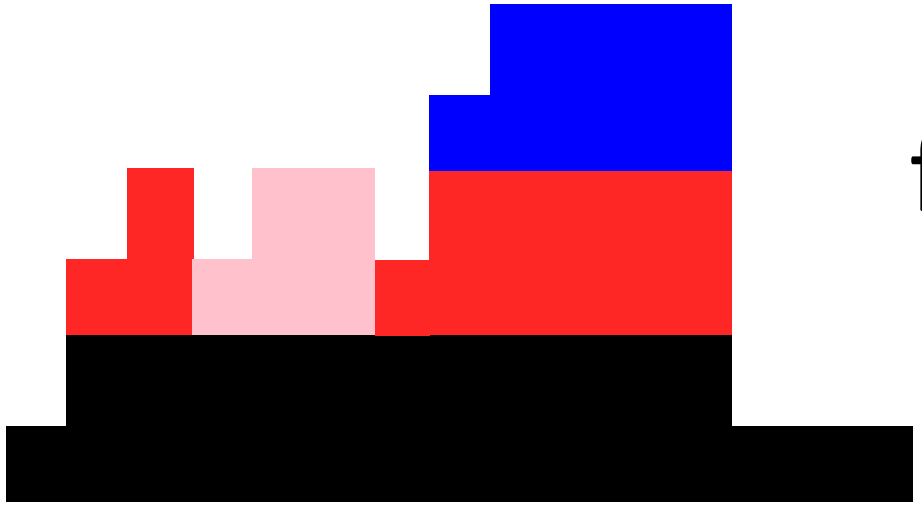


FRAMECEPTION

HTTP/3 frames

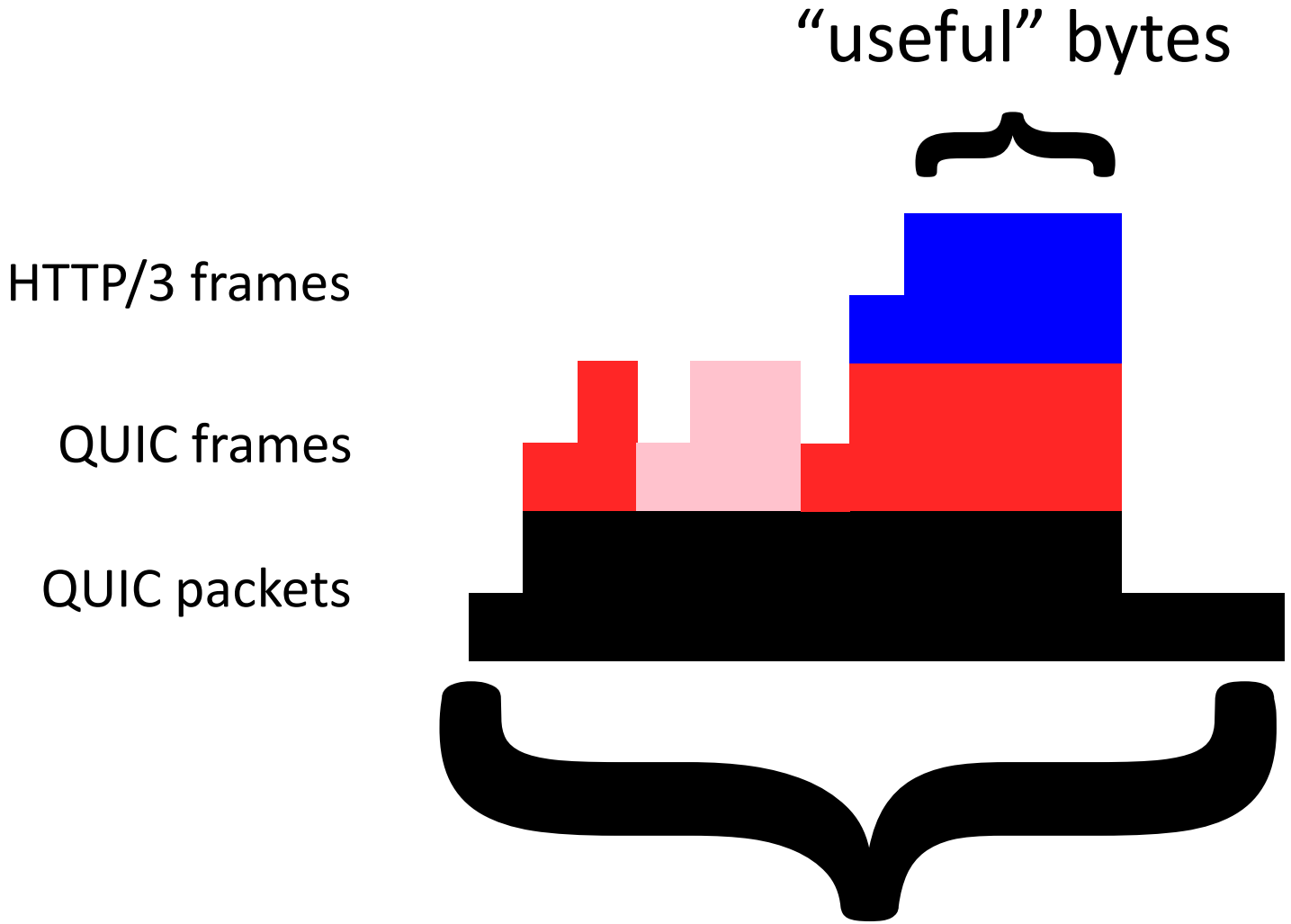
QUIC frames

QUIC packets



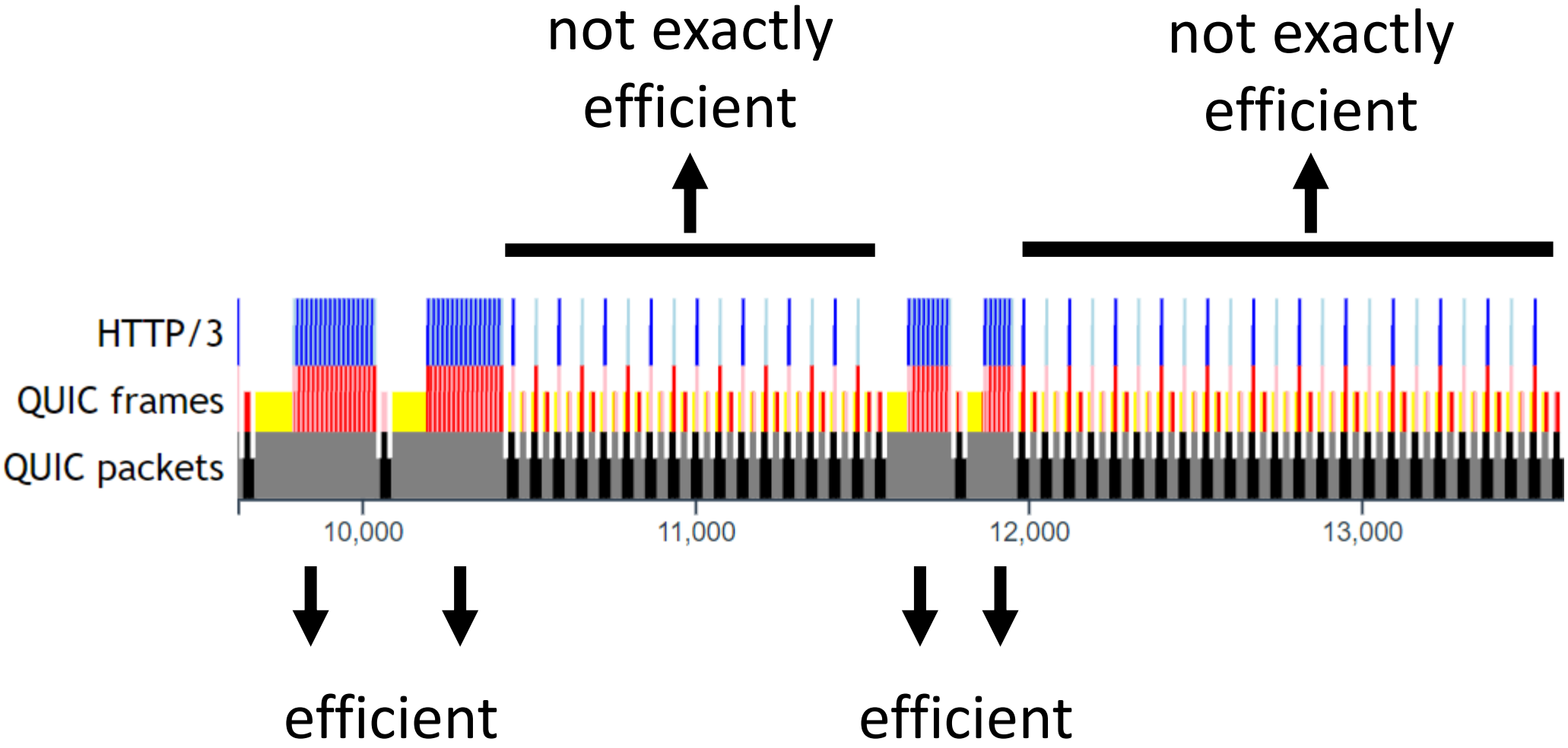
"look at the frame header of the HEADERS frame..."

FRAMECEPTION



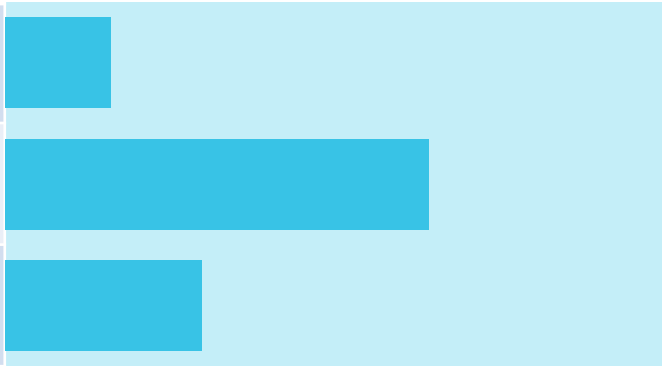
$$\frac{\text{useful}}{\text{ALL}} = \text{goodput efficiency}$$

FRAMECEPTION



EFFICIENCY

Worst-case efficiency	Stacks
95%+	1/8
90%-95%	5/8
83%-90%	2/8



“Haven’t optimized this part yet”

HTTP/3 DATA frame size	Stacks
> 1MB	6/13
100kB – 1MB	2/13
< 100kB	5/13

“We just use the full send buffer, driven by the cwnd”

PACKET SIZE

Bigger packets = more payload = less overhead

QUIC minimum packet size = **1200 bytes**

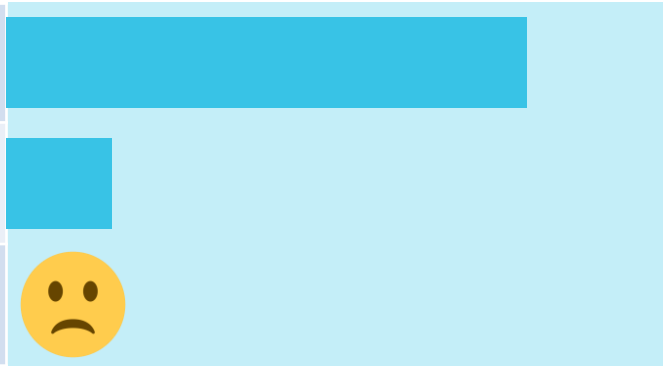
Networks typically support larger, e.g., **1500 bytes**

Needs to be discovered by

- **PMTUD**
- **DPLPMTUD**

PACKET SIZE

Packet size discovery	Stacks
None	11/14
Naïve (1x 1450+ byte)	3/14
Actual DPLPMTUD	0/14

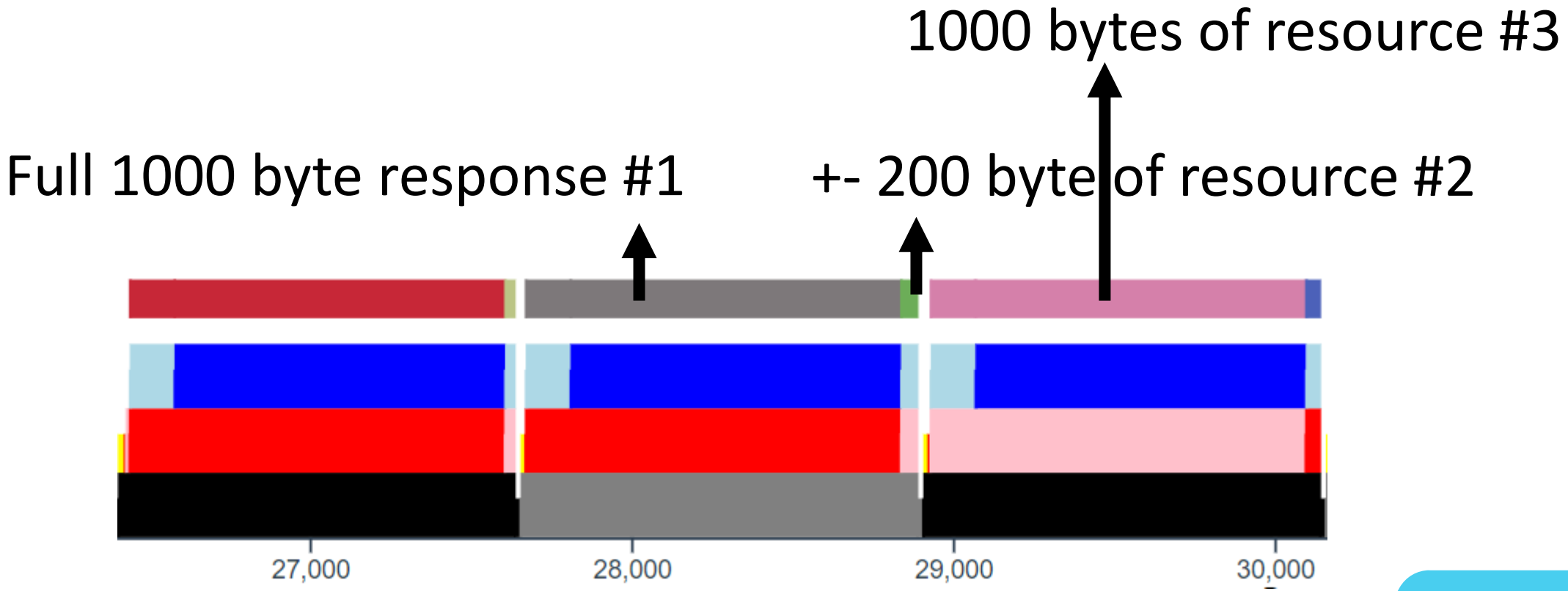


“Useful, but not a priority”

“I can’t handle acronyms of more than 5 letters”

PUTTING IT ALL TOGETHER

Test: request 100 resources of 1000 bytes each

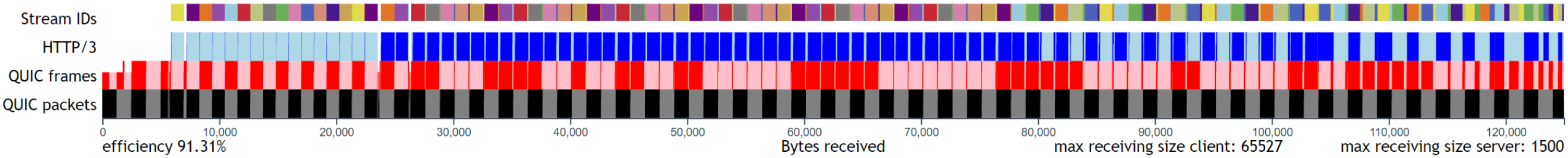


PUTTING IT ALL TOGETHER

HOL-blocked for 2 seconds

First 200 bytes

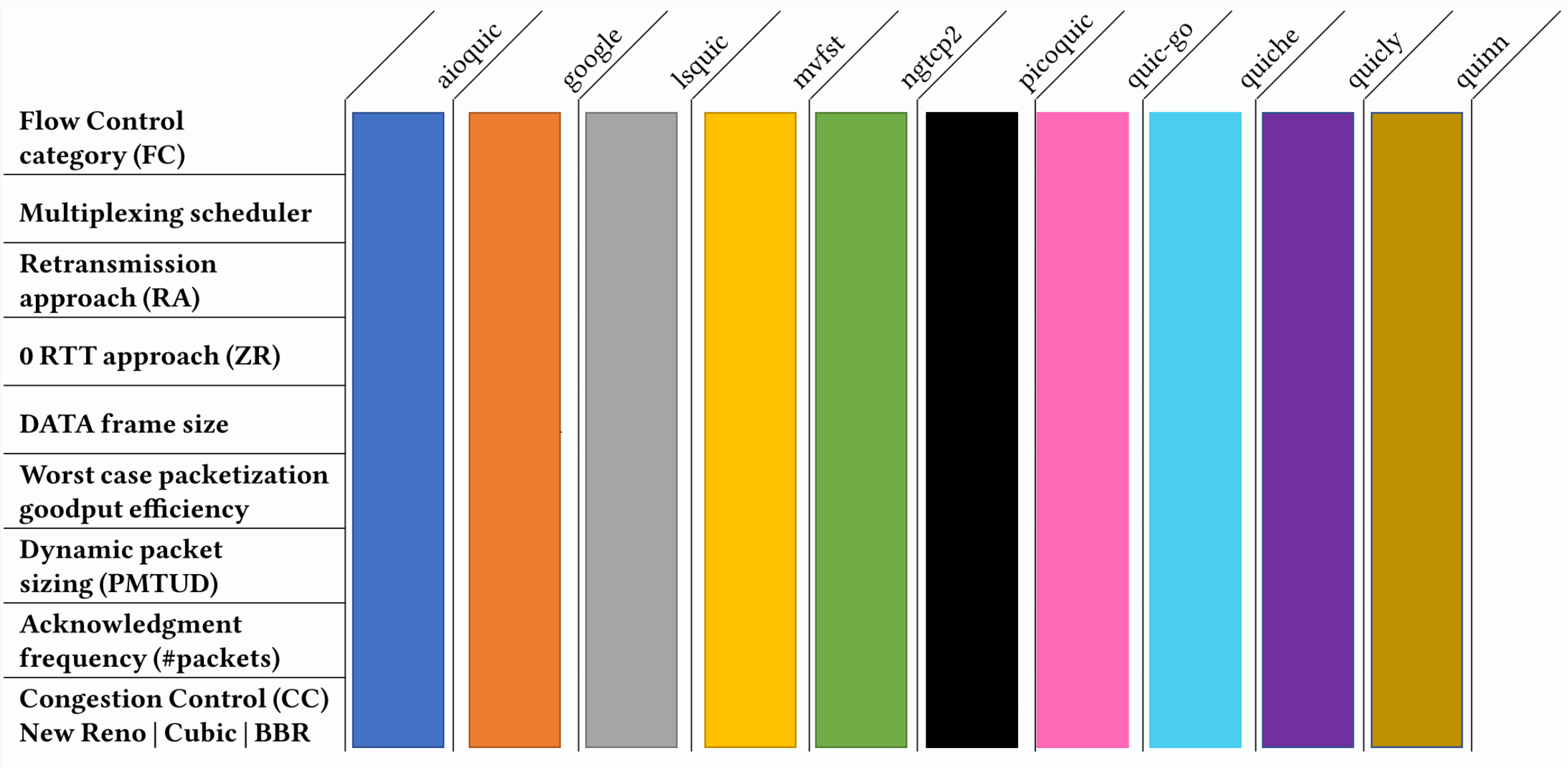
Final 800 bytes



WHAT ARE YOU SAYING EXACTLY?

	aioquic	google	lsquic	mvfst	ngtcp2	picoquic	quic-go	quiche	quicly	quinn
Flow Control category (FC)	2	1	1	1	1	2	1	1	1	1
Multiplexing scheduler	SEQ	RR	RR	RR	SEQ	SEQ	RR	RR	RR	RR
Retransmission approach (RA)	2	1	2	3	2	2	2	1	2	2
0 RTT approach (ZR)	1	1	2	3	1	2	2	1	2	1
DATA frame size	large	medium	small	large	small	large	large	small	large	small
Worst case packetization goodput efficiency	90.34%	95.02%	92.54%		90.88%	87.94%			91.52%	83.92%
Dynamic packet sizing (PMTUD)	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Acknowledgment frequency (#packets)	1	2-10	2-8	10	2-4	2-6	2-9	1	2	
Congestion Control (CC) New Reno | Cubic | BBR	✓|✗|✗	✗|✓|✓	✗|✓|✓	✓|✓|✓	✓|✗|✗	✓|✓|✓	✗|✓|✗	✓|✓|✗	✓|✗|✗	✓|✗|✗

WHAT ARE YOU SAYING EXACTLY?



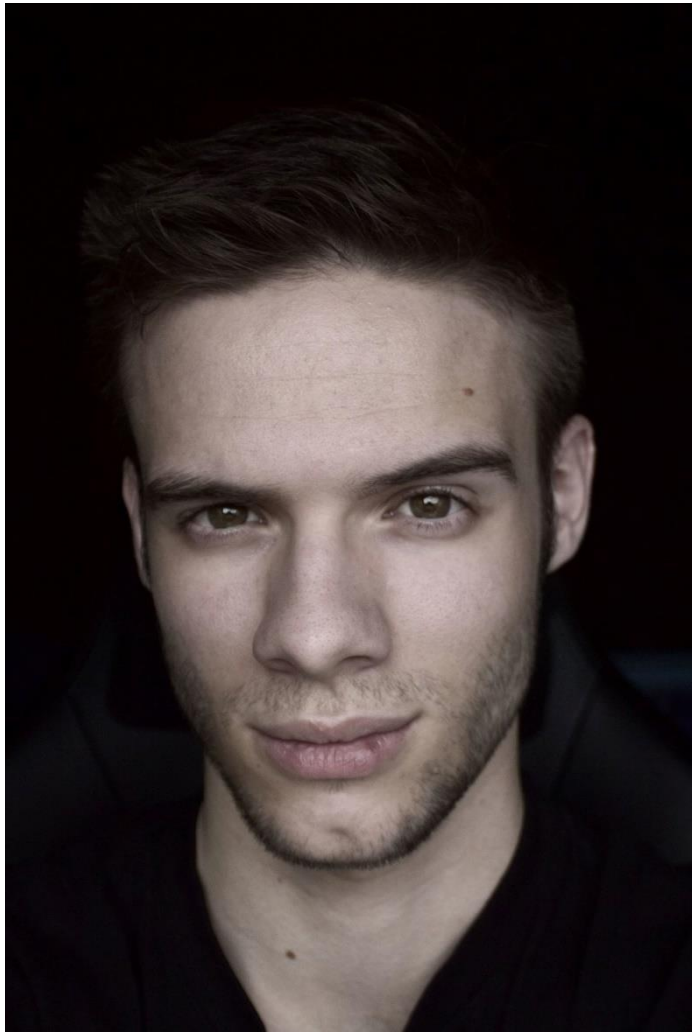
WHAT ARE YOU SAYING EXACTLY?

1. Test multiple implementations and compare results
2. Verify the implementation you use and its settings
3. Ideally: do both (maybe our tools can make this easier)

In my opinion:

Mature, but not mature enough for (most) performance testing

THE INDOMITABLE JORIS



First year PhD student

Mainly working on Video Streaming

THE INDOMITABLE JORIS

joris.herbots@uhasselt.be

QUIC
LEAGUE
2020

CONCLUSION

IT’S JUST A GAME

LOOK BEFORE YOU LEAP

1. QUIC is complex:

   many knobs to turn, easy to make it slower/faster

2. QUIC and HTTP/3 implementations aren't finished:

   don't trust, always verify

   QUIC 1 != QUIC 2 != QUIC 3 != QUIC 4

   test different implementations

3. You might want to look at our methodology

[qlog] <qvis>

IMAGE SOURCES

- https://seeklogo.com/images/V/versus-logo-97EAA46E88-seeklogo.com.png

CREDITS & COPYRIGHTS

- Template created by Showeet.com
- **Free font used:**
  - Calibri (Microsoft font)
- **Copyrights:**
  - Free with Attribution
  - Cannot be resold or redistributed under any circumstances
  - Cf. https://www.showeet.com/terms-of-use/

THANK YOU!

Hope you like this template :)

showeet

Free creative templates, charts, diagrams and maps for your outstanding presentations

https://www.showeet.com/