QUIC and HTTP/3: Too big to fail!?

Robin Marx - @programmingart
PhD researcher – Hasselt University

UHASSELT EDM

https://quic.edm.uhasselt.be
Symposium on the Future of HTTP – March 2019

QUIC and HTTP/3 are going to change the world!

Lucas Pardue @SimmerVigor · Mar 13

Replying to @alagoutte

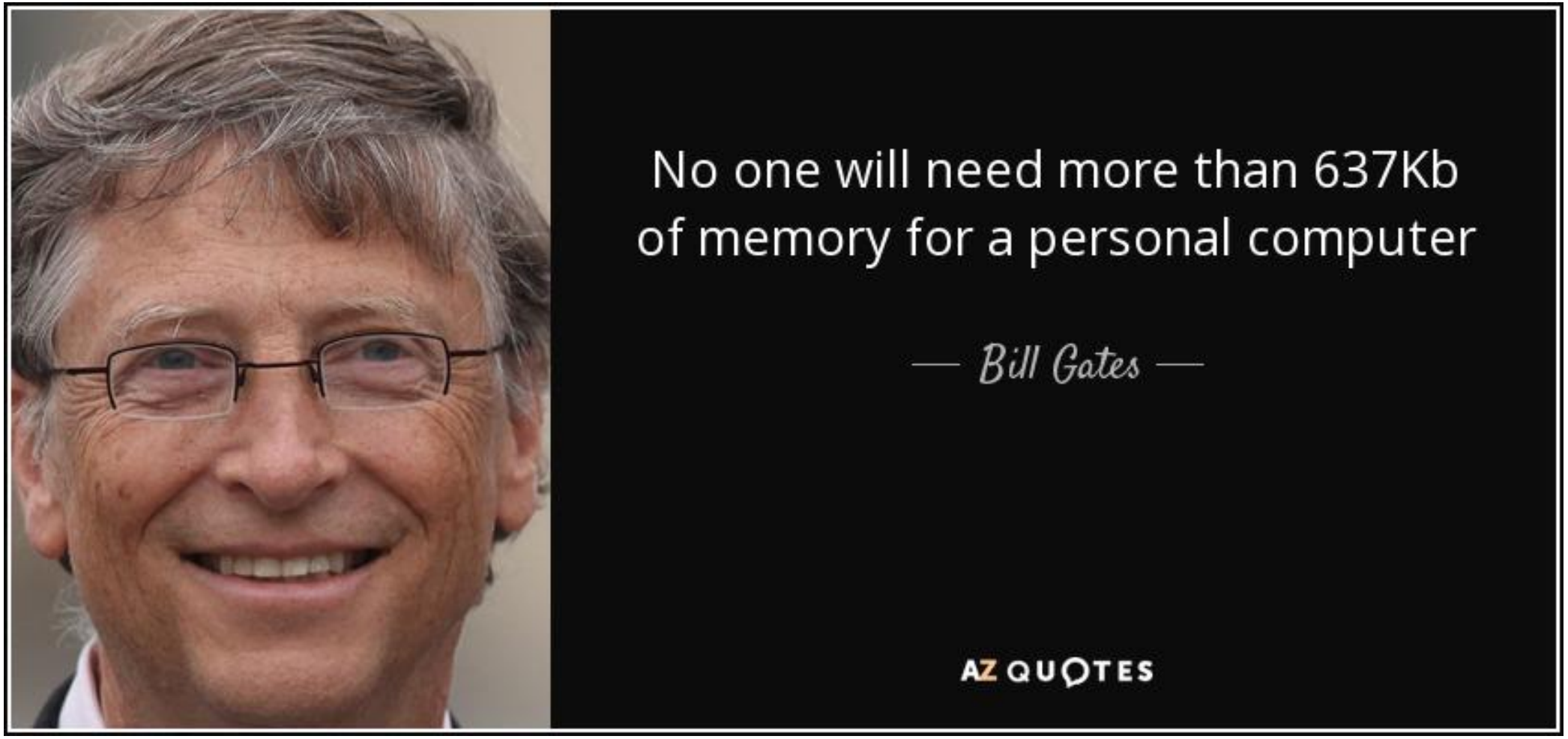
In the next 10 years:

HTTP will go to a yearly release cycle. So we will have HTTP/2019 through to HTTP/2029.

QUIC wil replace everything, even payment systems and **5G**.

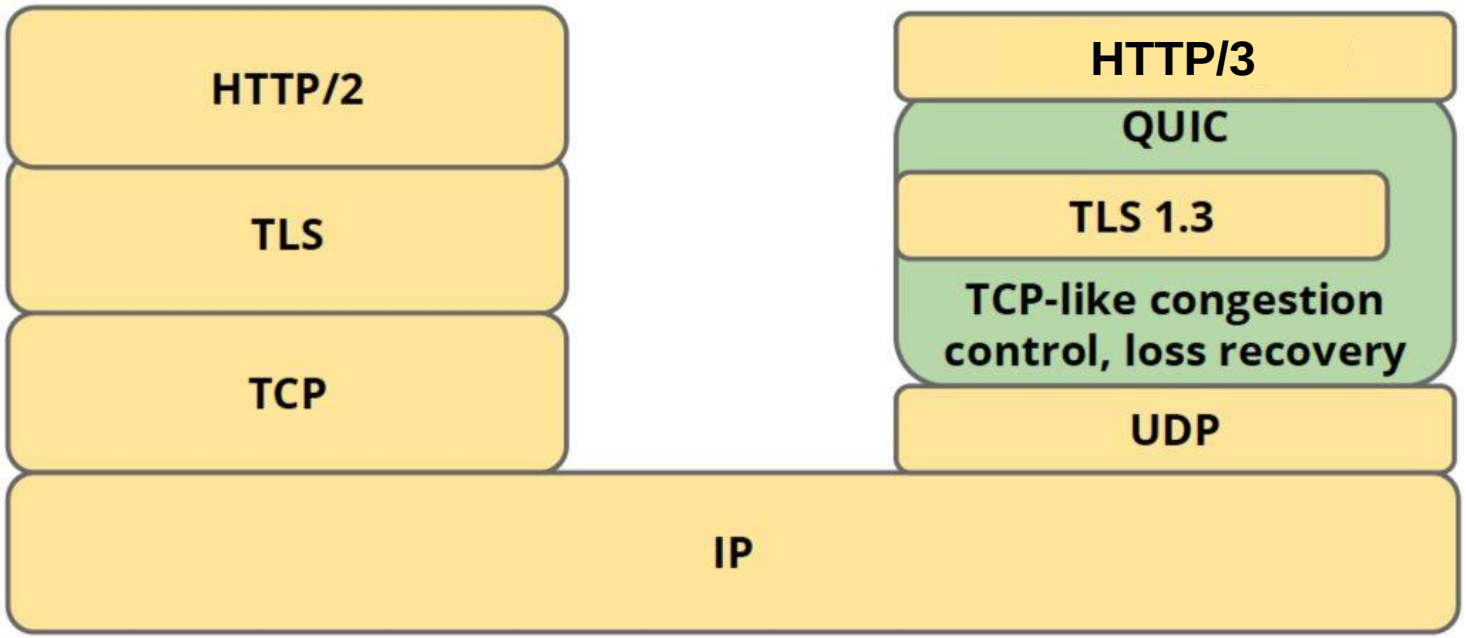
2 1

QUIC and HTTP/3 might change the world!

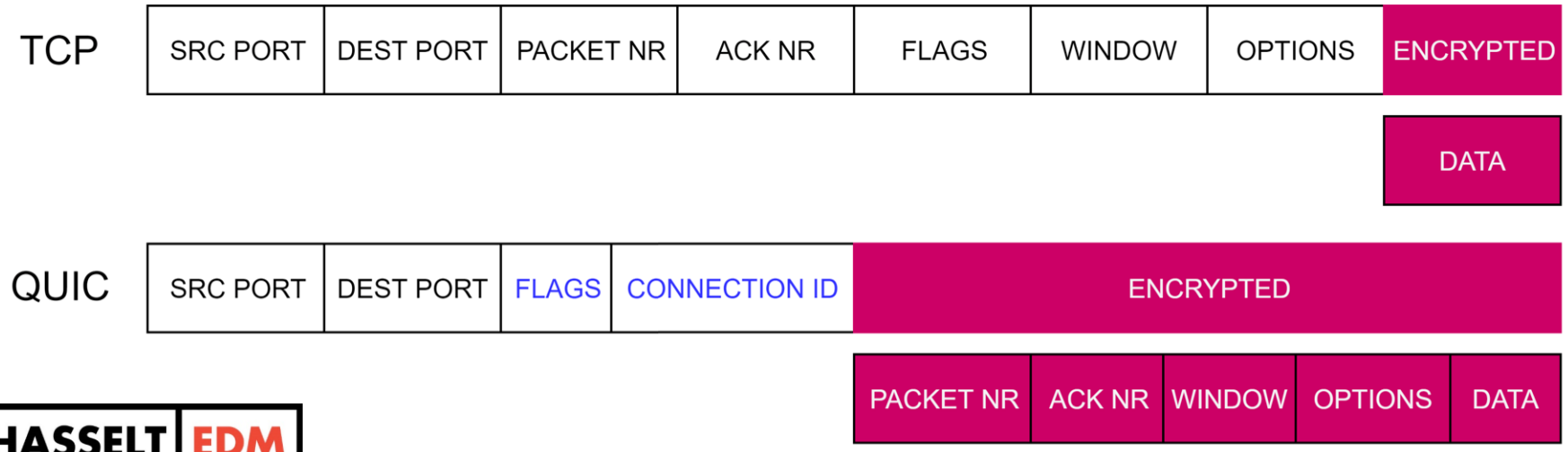


https://calendar.perfplanet.com/2018/quic-and-http-3-too-big-to-fail/
Image: https://www.azquotes.com/picture-quotes/quote-no-one-will-need-more-than-637kb-of-memory-for-a-personal-computer-bill-gates-55-85-83.jpg

QUIC is special

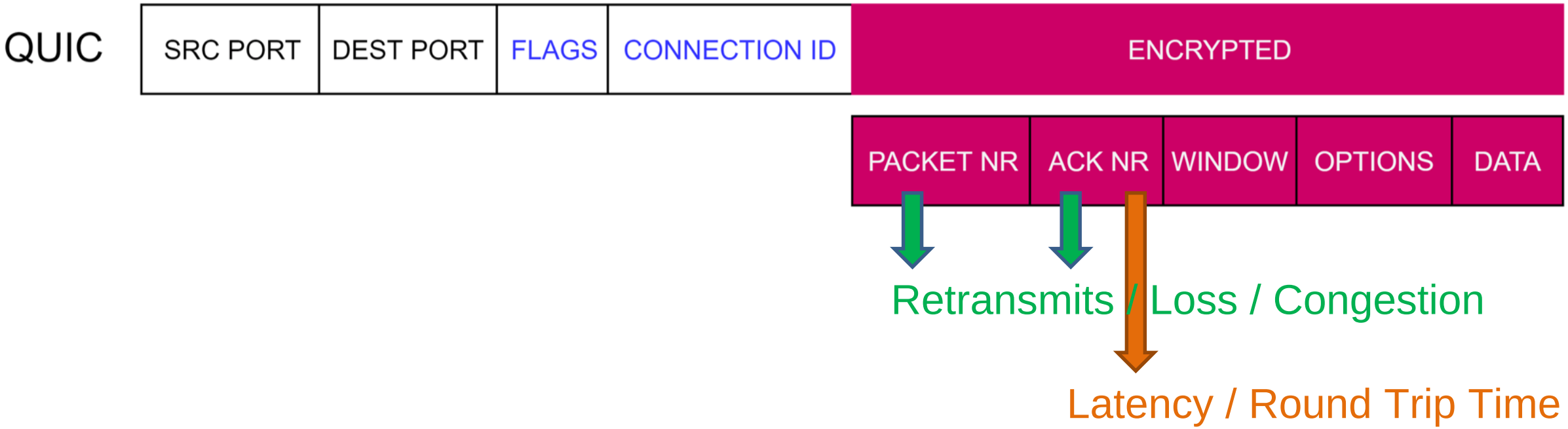


Re-implement:
- Reliability
- Ordering
- Congestion Control
- Flow Control
- …



Middlebox "Ossification" prevention

QUIC is end-to-end encrypted

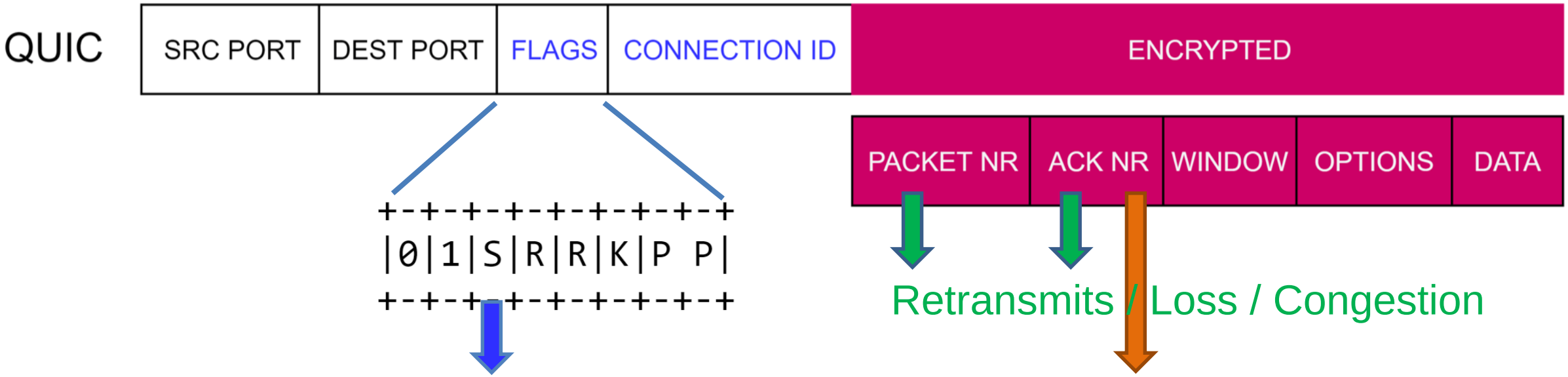


Firewall / security logic

Fake them for extra performance

https://youtu.be/TQq6Z4_HBaY?t=1357

QUIC is end-to-end encrypted

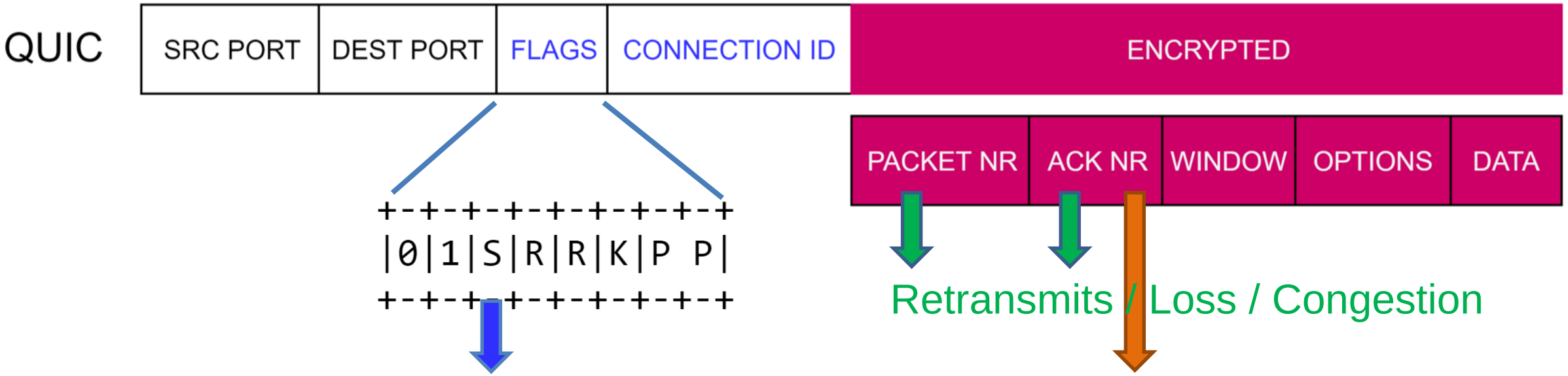


Firewall / security logic

Fake them for extra performance

https://youtu.be/TQq6Z4_HBaY?t=1357

QUIC is end-to-end encrypted



Firewall / security logic

Fake them for extra performance

https://youtu.be/TQq6Z4_HBaY?t=1357

QUIC is end-to-end encrypted

Image: https://images7.alphacoders.com/804/thumb-1920-804571.jpg
https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf

QUIC is end-to-end encrypted

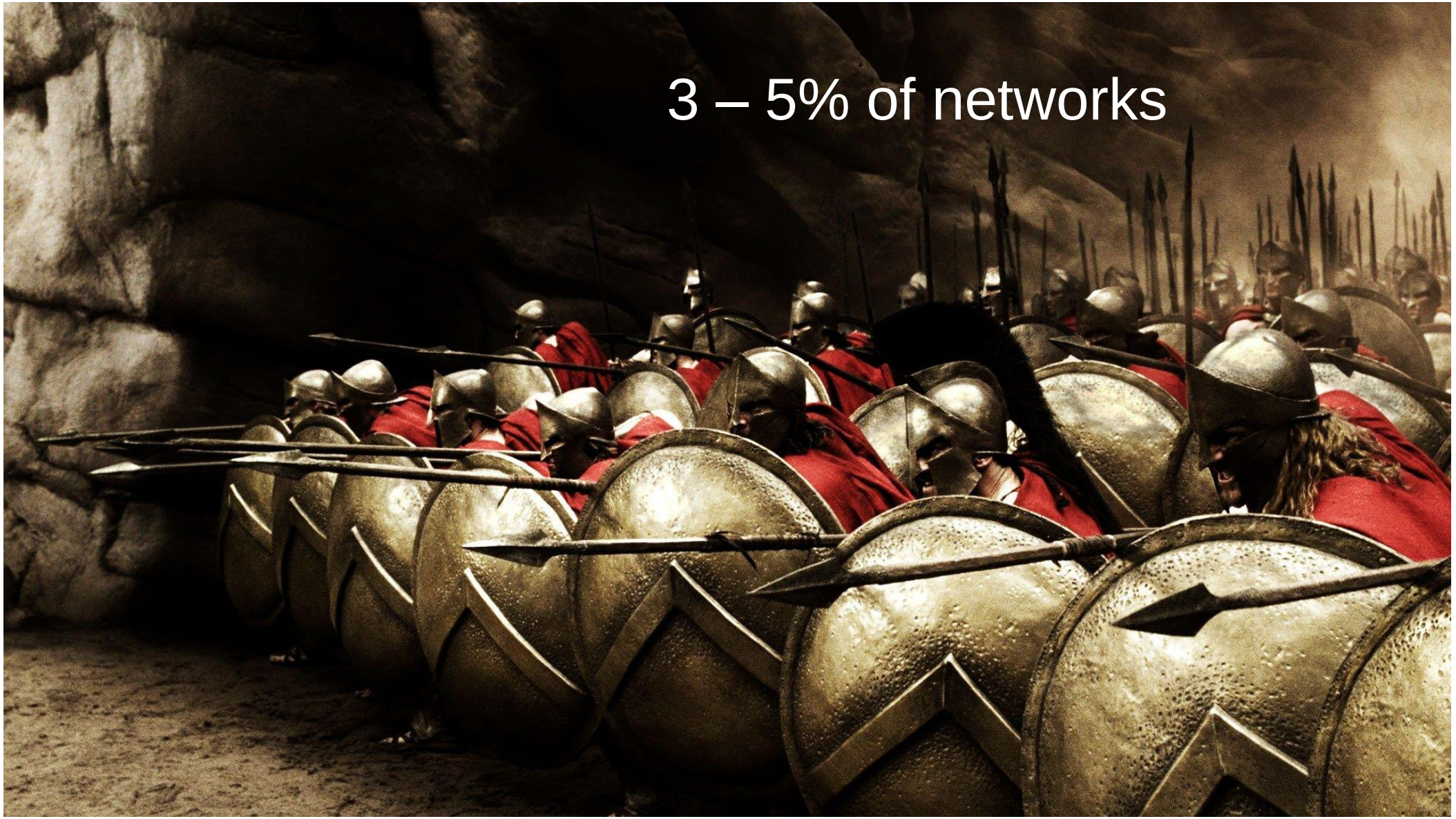


Image: https://images7.alphacoders.com/804/thumb-1920-804571.jpg
https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf

QUIC is end-to-end encrypted

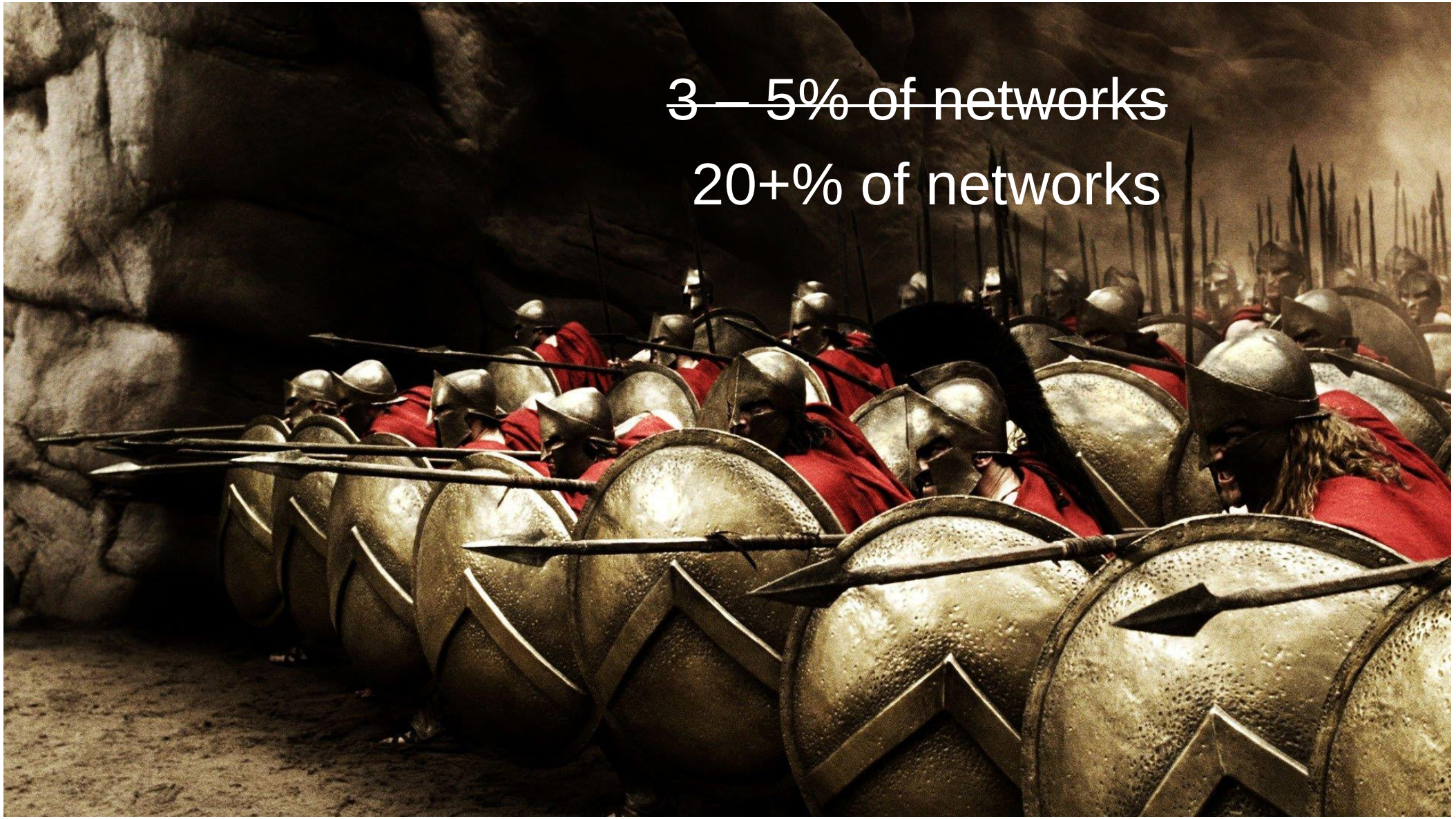


Image: https://images7.alphacoders.com/804/thumb-1920-804571.jpg
https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf

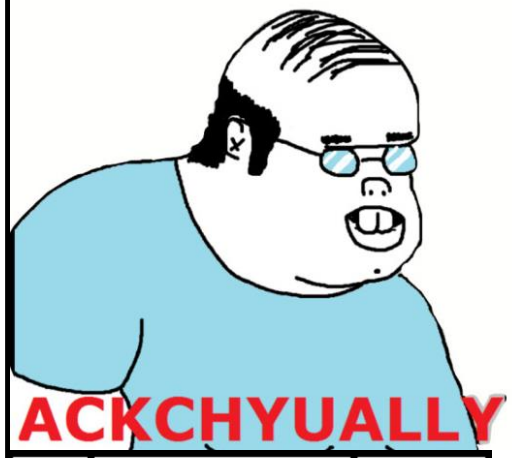
QUIC is end-to-end encrypted

Image: https://images7.alphacoders.com/804/thumb-1920-804571.jpg
https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf

QUIC is end-to-end encrypted: counterarguments

- Block QUIC = block big players (Google, FB, …)
- QUIC doesn't need performance enhancing middleboxes
  - But… satellites

- They have no reason to block QUIC



www7content.cs.fau.de/~deutschmann/NetSys2019_preprint.pdf

QUIC is done in Userspace

"QUIC uses only 2x as an equivalent TCP + TLS stack"

- Google engineers

https://dl.acm.org/citation.cfm?id=3098842

QUIC is done in Userspace

"QUIC uses only 2x as an equivalent TCP + TLS stack"

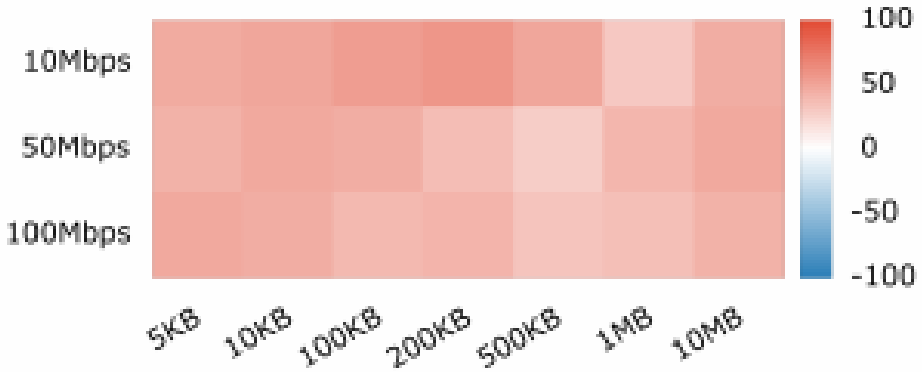
- Google engineers

"You need a sh*tload of extra servers to run QUIC"

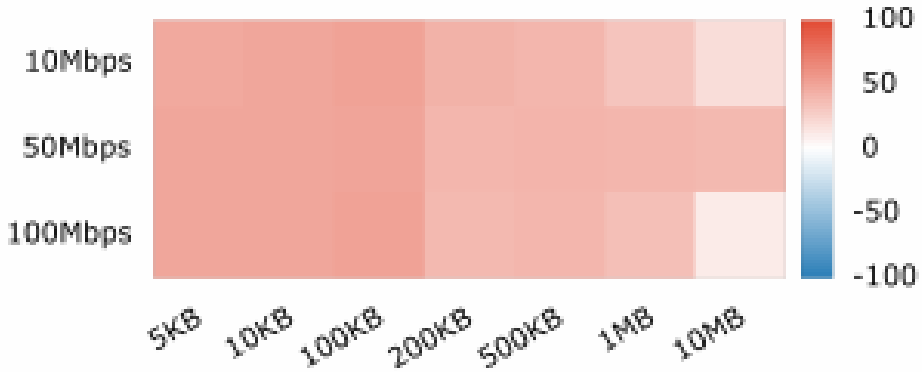
- What I'm reading

https://dl.acm.org/citation.cfm?id=3098842

QUIC is done in Userspace



(a) Varying object size,1% Loss

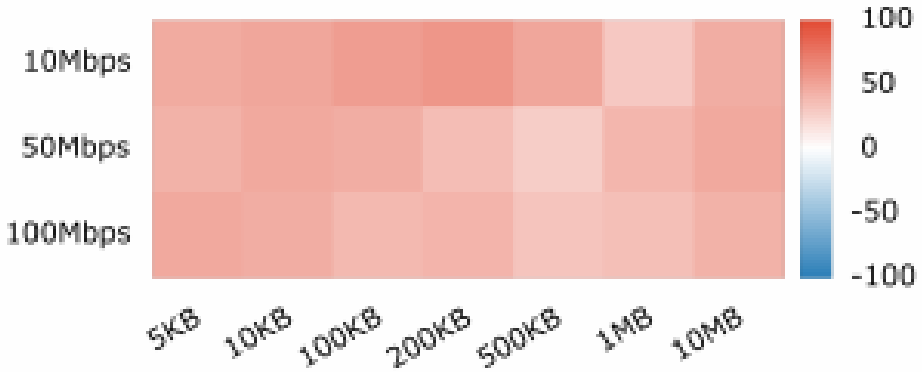


(b) Varying object size, 112 ms RTT

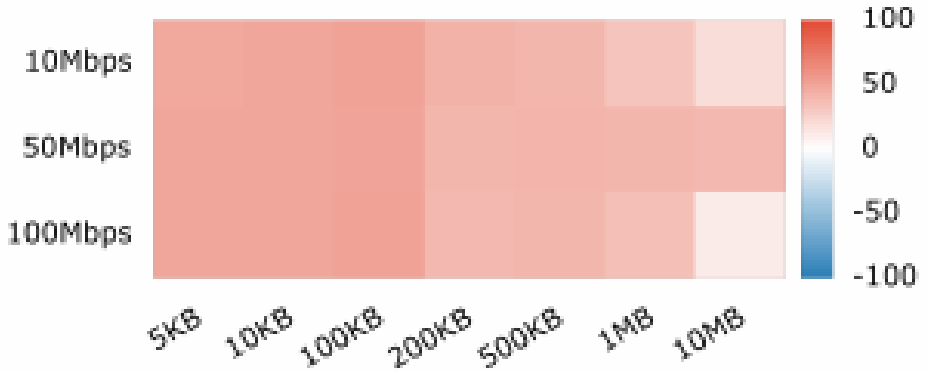
Desktop

http://david.choffnes.com/pubs/long-look-at-quic-imc17.pdf

QUIC is done in Userspace

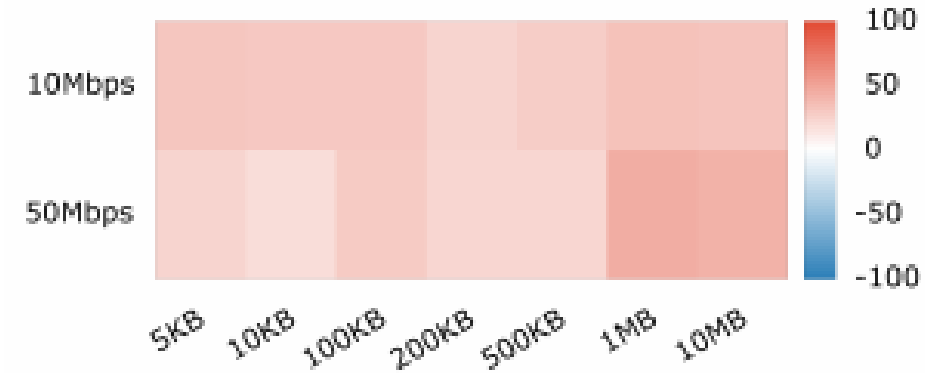


(a) Varying object size,1% Loss

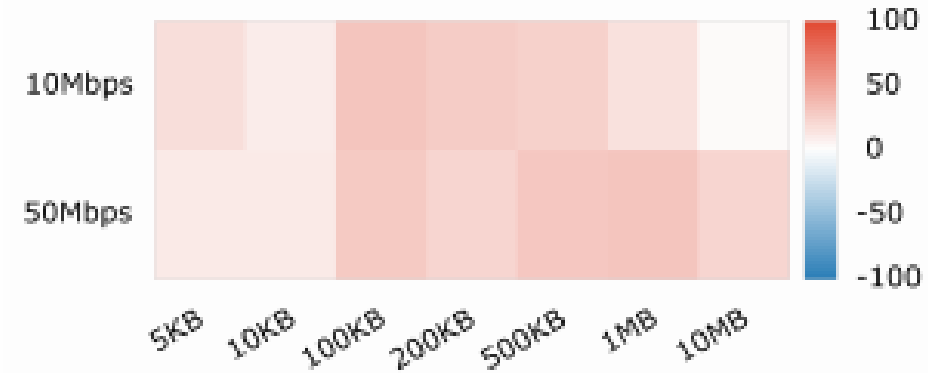


(b) Varying object size, 112 ms RTT

Desktop



(e) Nexus6, 1% Loss



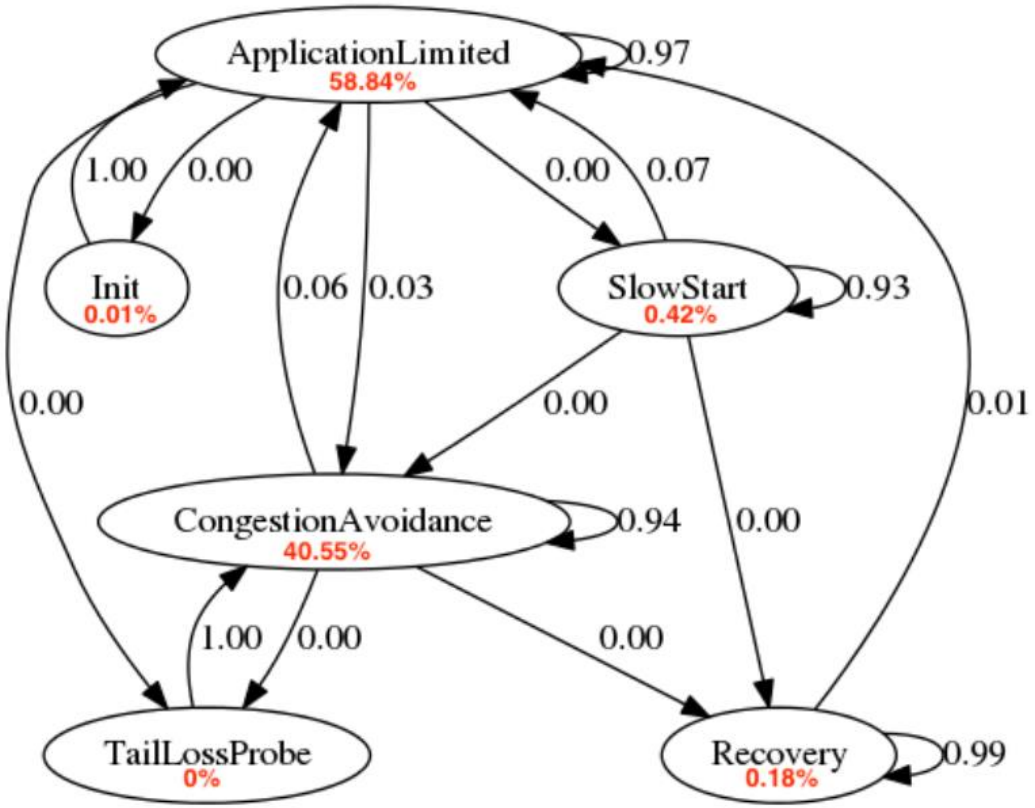
(f) Nexus6, 112 ms RTT

Mobile

http://david.choffnes.com/pubs/long-look-at-quic-imc17.pdf

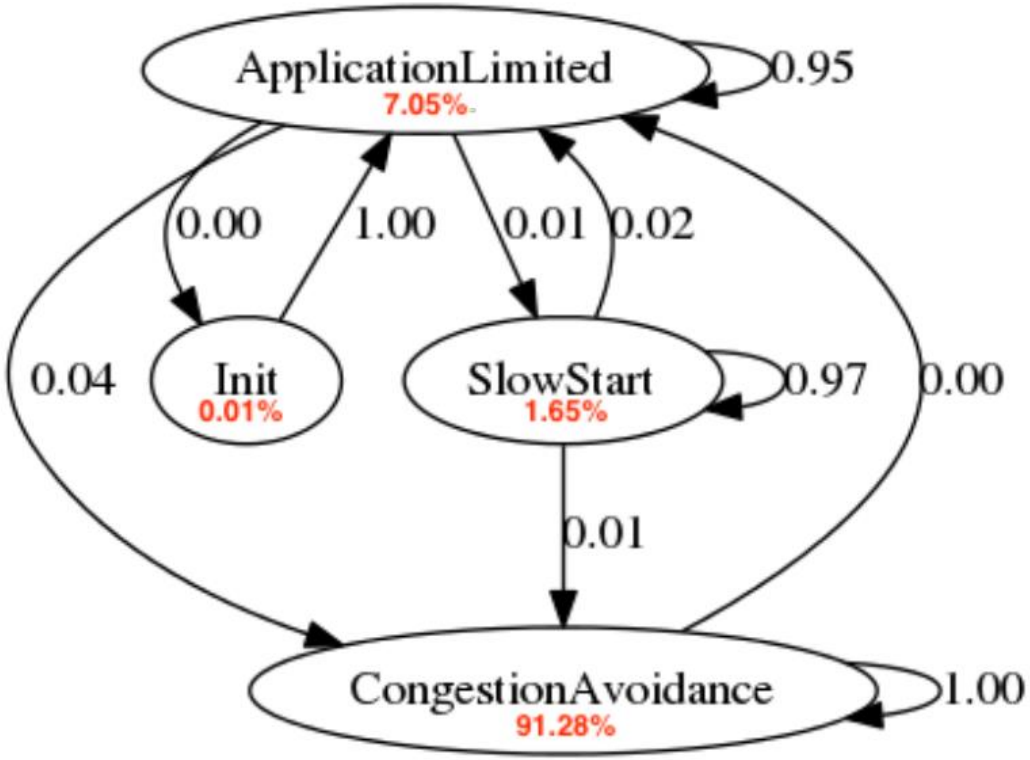
QUIC is done in Userspace

58.84%



(a) MotoG

7.05%



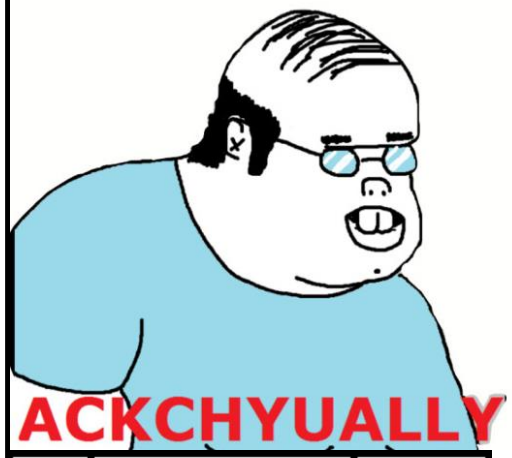
(b) Desktop

http://david.choffnes.com/pubs/long-look-at-quic-imc17.pdf

QUIC is done in Userspace: Counterarguments

- QUIC will get hardware offload / move to kernel
  - But… Variable-length encoding
  - But… ACK length ;)

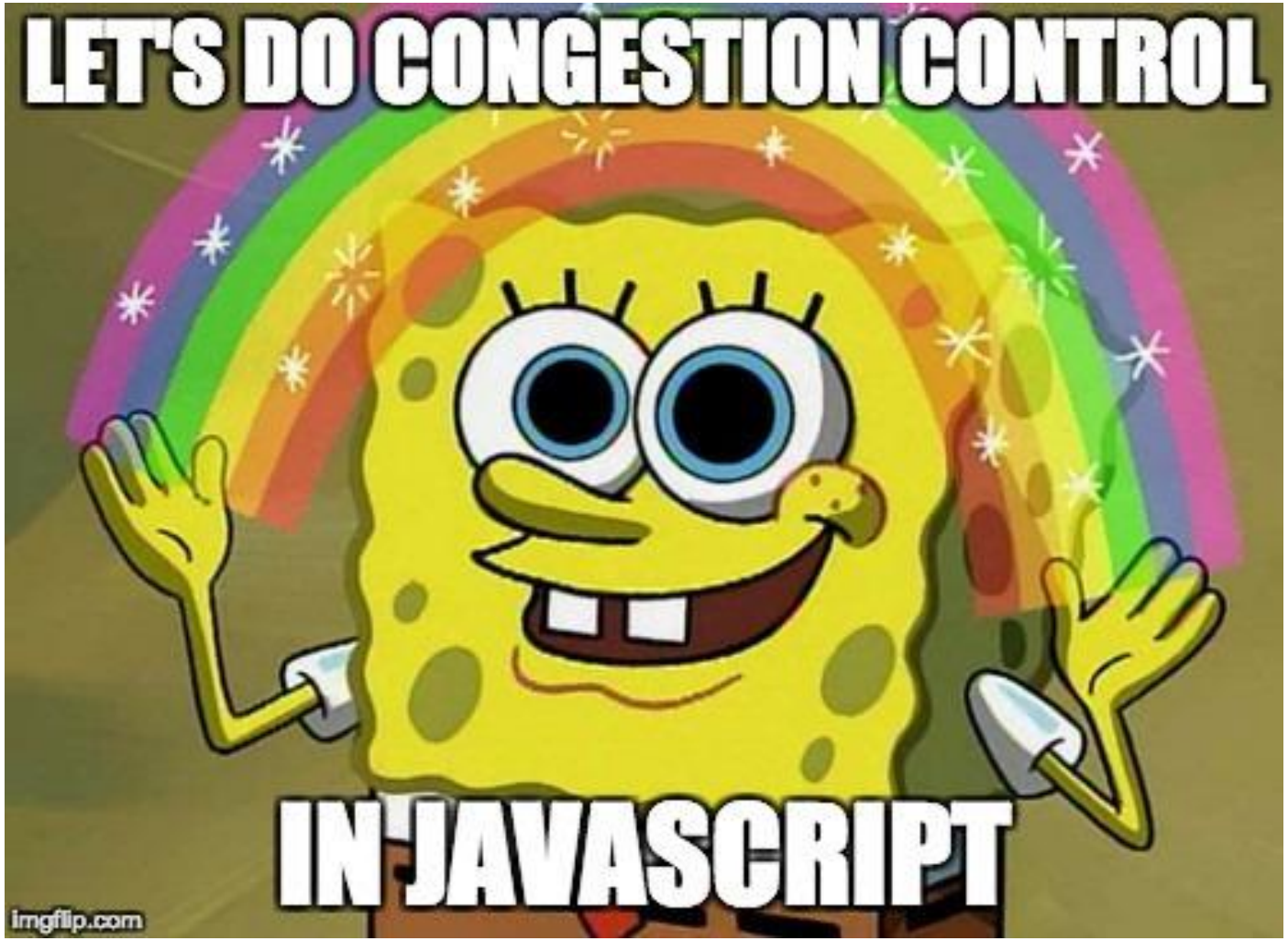
- Even with this overhead, **Google runs QUIC at scale**



UHASSELT EDM

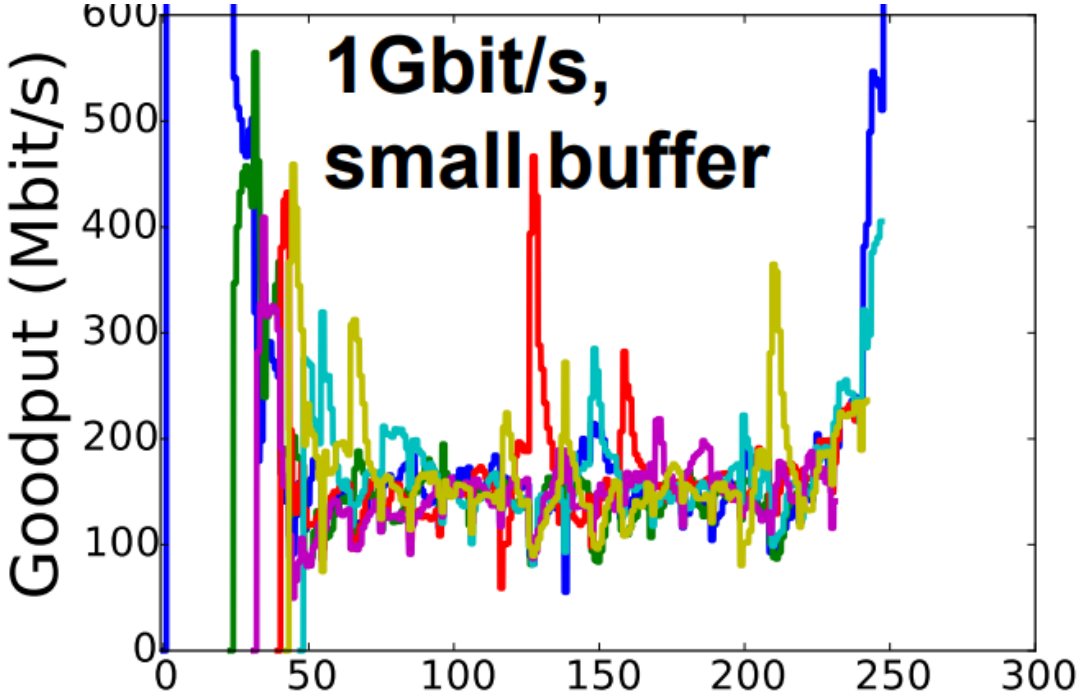
https://docs.google.com/document/d/16SZDhfR2IspQLQ8s_-FiKBZRgp2WJ02gtDZsWYsNVJ8/edit?usp=sharing
https://datatracker.ietf.org/meeting/104/materials/slides-104-quic-offloading-quic-00

QUIC is done in Userspace: Reprise



https://github.com/rmarx/quicker/tree/congestionControl
Image: https://w3cmemes.wordpress.com/2017/11/07/lets-just-do-everything-in-js

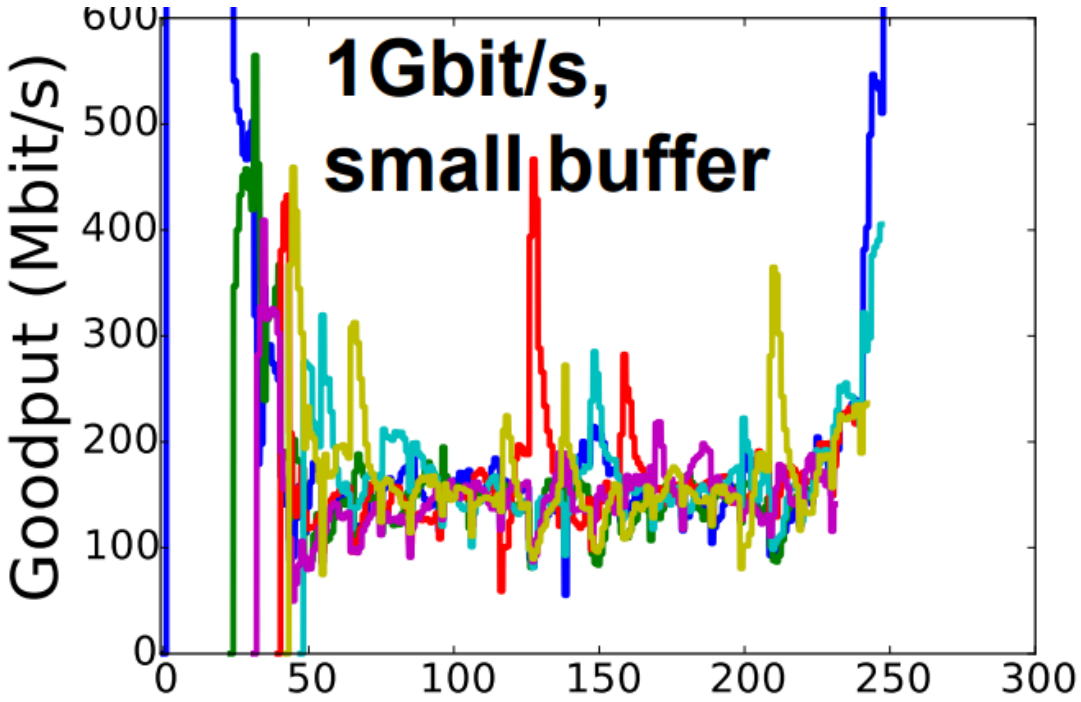
QUIC is done in Userspace: Reprise



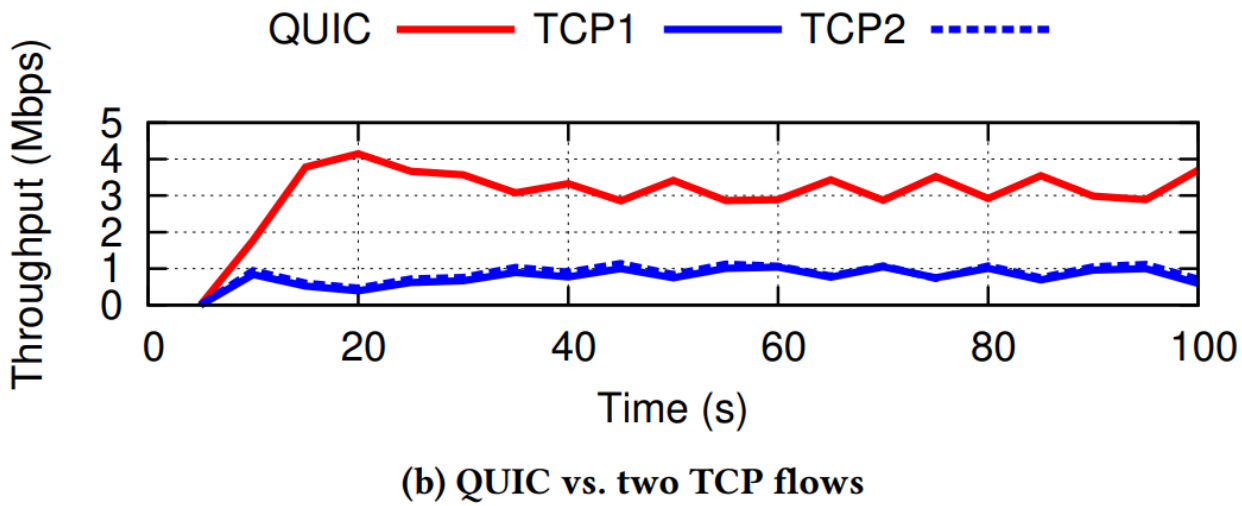
Fair

Image: http://david.choffnes.com/pubs/long-look-at-quic-imc17.pdf
Image: https://datatracker.ietf.org/meeting/100/materials/slides-100-iccrg-an-experimental-evaluation-of-bbr-congestion-control-00
https://huitema.wordpress.com/2019/01/12/will-transport-innovation-collapse-the-internet

QUIC is done in Userspace: Reprise



Fair



(b) QUIC vs. two TCP flows

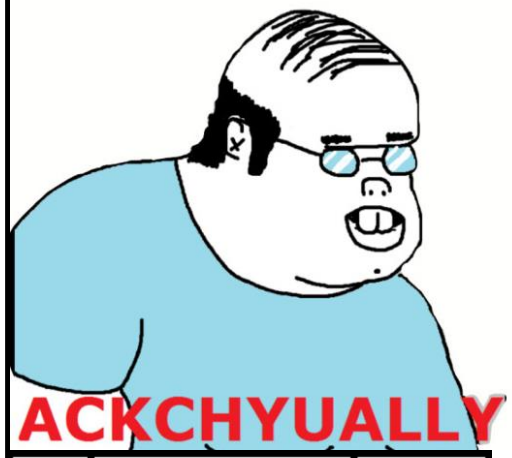
Not so fair

Image: http://david.choffnes.com/pubs/long-look-at-quic-imc17.pdf
Image: https://datatracker.ietf.org/meeting/100/materials/slides-100-iccrg-an-experimental-evaluation-of-bbr-congestion-control-00
https://huitema.wordpress.com/2019/01/12/will-transport-innovation-collapse-the-internet

QUIC is done in Userspace: Reprise: Counterarguments

- Networks themselves will prevent abuse (AQM)

- Has been possible for ages, no real-world abuse noticed
  - But… BBR
  - But… 6 parallel TCP connections in HTTP/1.1



QUIC cuts down on latency with 0-RTT

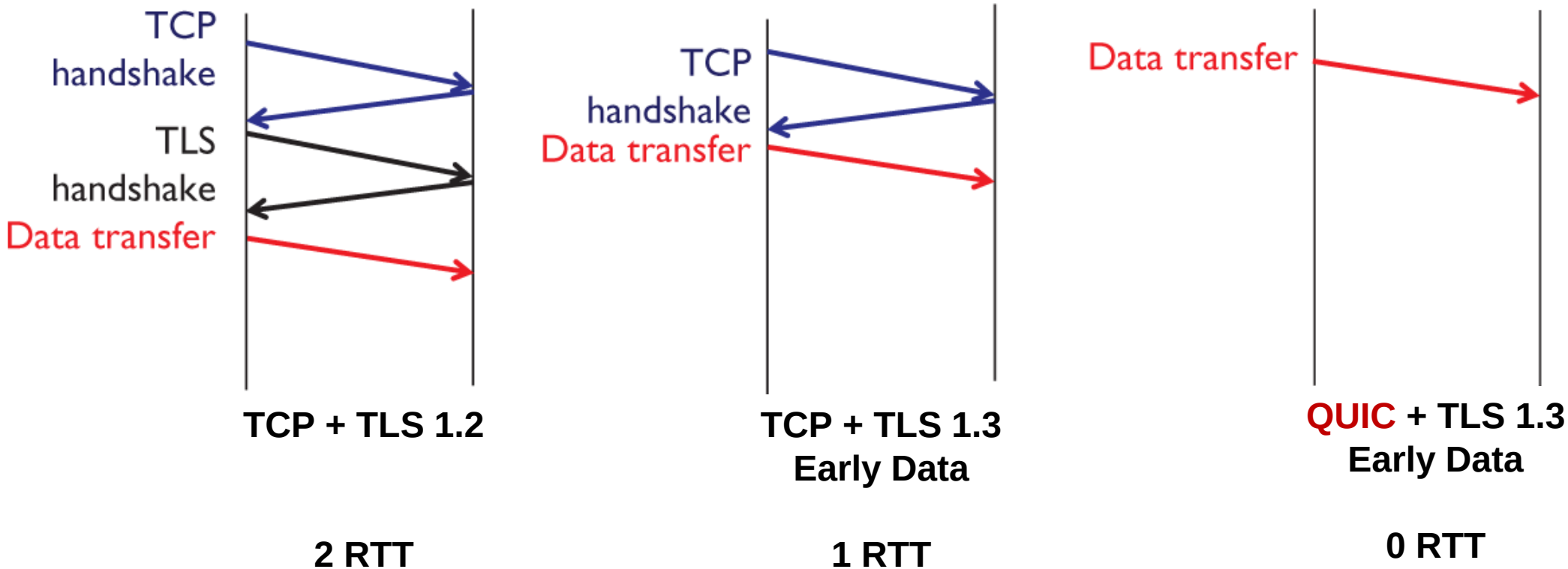
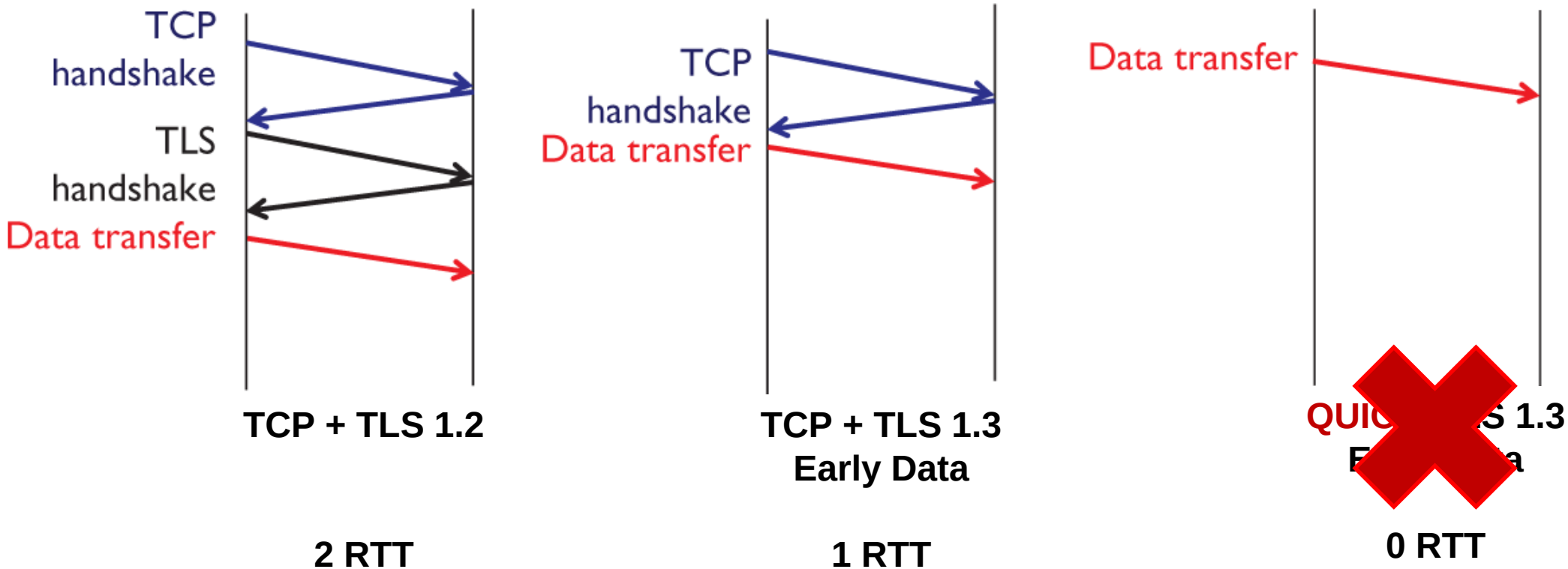


Image: https://www.computer.org/csdl/magazine/ic/2017/02/mic2017020072/13rRUxjyX8n

QUIC cuts down on latency with 0-RTT



TCP FAST OPEN +
TLS 1.3 Early Data

Image: https://www.computer.org/csdl/magazine/ic/2017/02/mic2017020072/13rRUxjyX8n

QUIC cuts down on latency with 0-RTT

Poor Daniel

0-RTT HTTP POST

1. Pay Robin $100 for his talk

Crafty Robin

Image: http://www.iconarchive.com/show/noto-emoji-travel-places-icons-by-google/42492-bank-icon.html

QUIC cuts down on latency with 0-RTT

Poor Daniel

0-RTT HTTP POST

1. Pay Robin $100 for his talk

Crafty Robin

2. Pay Robin $100

3. Pay Robin $100

4. Pay Robin $100

5. Pay Robin $100

6. Pay Robin $100

7. Pay Robin $100

8. Pay Robin $100

Replay attack: can’t just send anything

Image: http://www.iconarchive.com/show/noto-emoji-travel-places-icons-by-google/42492-bank-icon.html

QUIC cuts down on latency with 0-RTT

Angry Daniel
1.1.1.1

Deserving Robin
2.2.2.2

0-RTT HTTP GET
“I am Robin at 2.2.2.2”
Send me one-gigabyte-file.json

Image: https://www.kisspng.com/free/web-server.html
https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/

QUIC cuts down on latency with 0-RTT

Angry Daniel
1.1.1.1

0-RTT HTTP GET
“I am Robin at 2.2.2.2”
Send me one-gigabyte-file.json

one-gigabyte-file.json

Deserving Robin
2.2.2.2

UDP Amplification attack: can’t send too much

Image: https://www.kisspng.com/free/web-server.html
https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/

QUIC cuts down on latency with 0-RTT

Angry Daniel
1.1.1.1

0-RTT HTTP GET
“I am Robin at 2.2.2.2”
Send me one-gigabyte-file.json

First 3-6 packets of
one-gigabyte-file.json

Deserving Robin
2.2.2.2

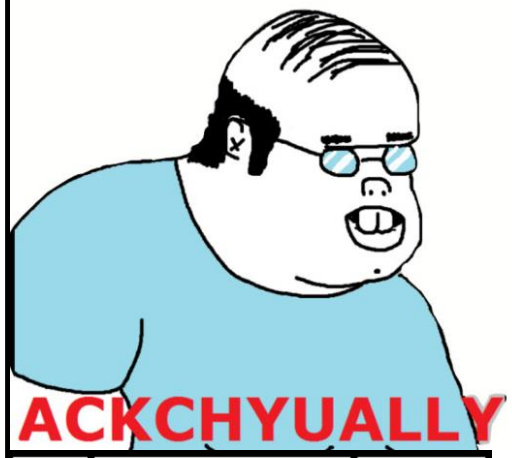
UDP Amplification attack: can’t send too much

Image: https://www.kisspng.com/free/web-server.html
https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/

QUIC cuts down on latency with 0-RTT: counterarguments

- TCP Fast Open isn’t feasible on real networks
  - But… just right now

- Clients can send 9000+ 0-RTT packets filled with padding
  - 1 0-RTT GET + 29 filled with zeroes => 90 packets response data!



https://blog.cloudflare.com/introducing-0-rtt/

QUIC has version negotiation

QUIC v3.5.66.6.8.55-Facebook

QUIC has version negotiation

QUIC v3.5.66.6.8.55-Facebook

QUIC used to have version negotiation

B.1. Since draft-ietf-quic-transport-18

- Removed version negotation; version negotiation, including authentication of the result, will be addressed in the next version of QUIC (#1773, #2313)

QUIC used to have version negotiation

B.1. Since draft-ietf-quic-transport-18

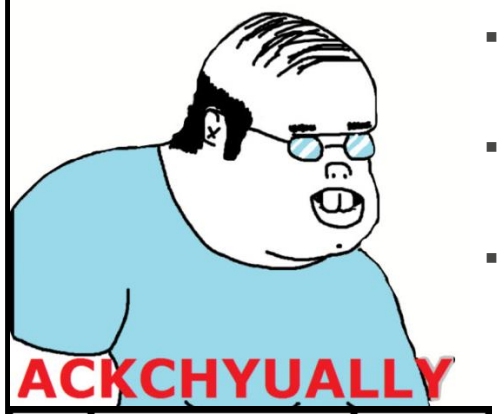
- Removed version negotation; version negotiation, including authentication of the result, will be addressed in the next version of QUIC (#1773, #2313)

QUIC used to have version negotiation: counterarguments

- We still have transport parameters and extension frames
- v2 will become main version and v1 will disappear quickly
- Clients will cache versions
  - But… Caching is 1 of the 3 big problems in CS

QUIC used to have version negotiation: counterarguments

- We still have transport parameters and extension frames
- v2 will become main version and v1 will disappear quickly
- Clients will cache versions
  - But… Caching is 1 of the 3 big problems in CS



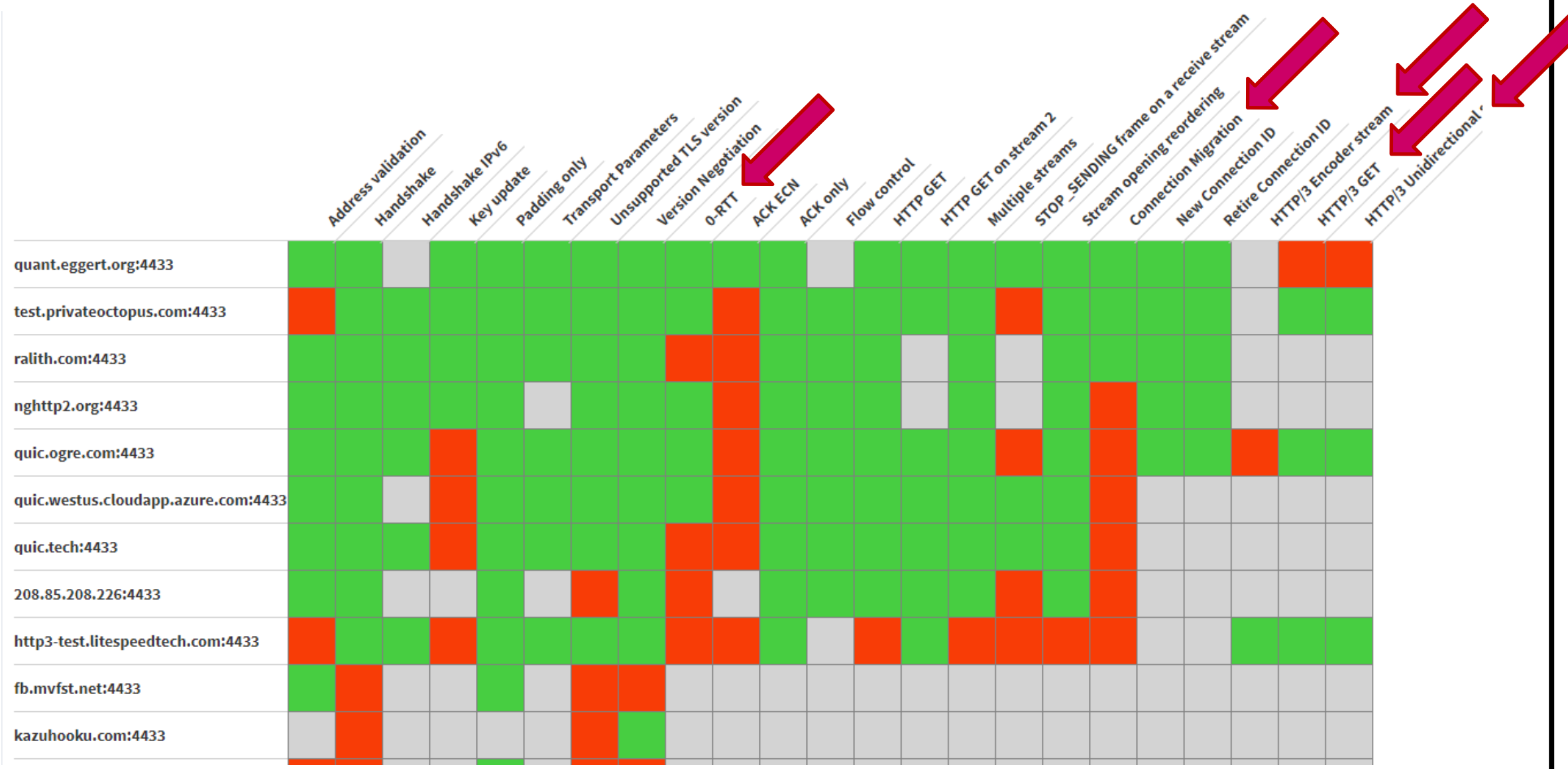
2. Agreeing on the Spinbit
3. Not logging plaintext passwords
4. Off-by-one errors

QUIC is at exactly the right complexity

- V1
  - Congestion control + loss detection
  - Flow control
  - Encryption and integrity protection
  - Connection migration
  - 0-RTT support
  - Independent streams
  - Low overhead
  - DoS prevention
  - Stateless Retry
  - …
  - Not even talking about HTTP/3 features here

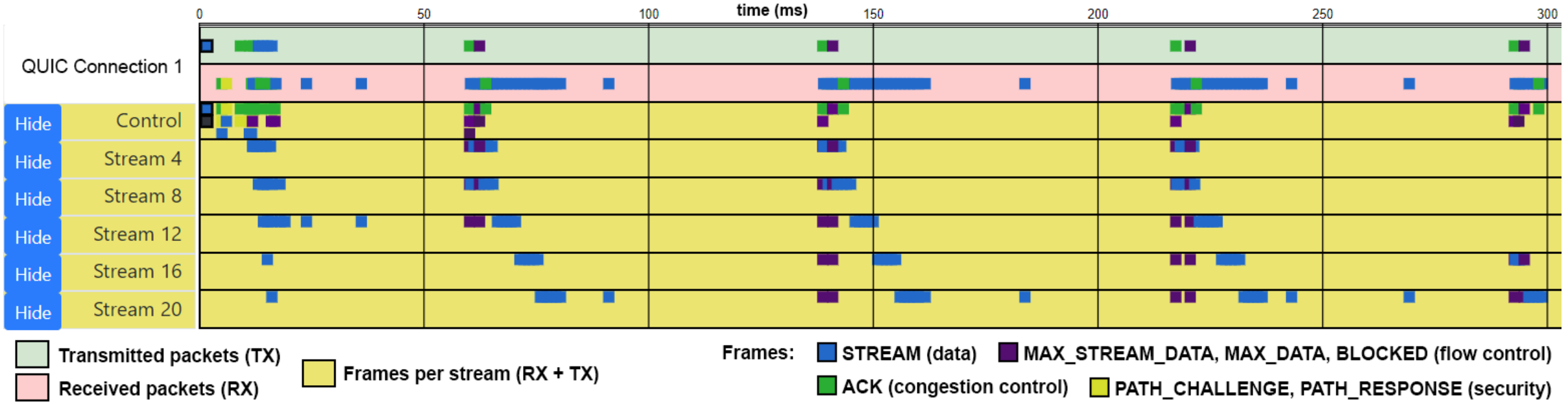
- Postponed to V2
  - Multipath
  - Forward error correction
  - Unreliable data transfer
  - Support for other crypto besides TLS 1.3
  - …
  - Most non-HTTP/3 applications are being postponed to V2
  - IoT, realtime media, …

QUIC is at exactly the right complexity



https://quic-tracker.info.ucl.ac.be/grid

QUIC is at exactly the right complexity



QUIC is at exactly the right complexity

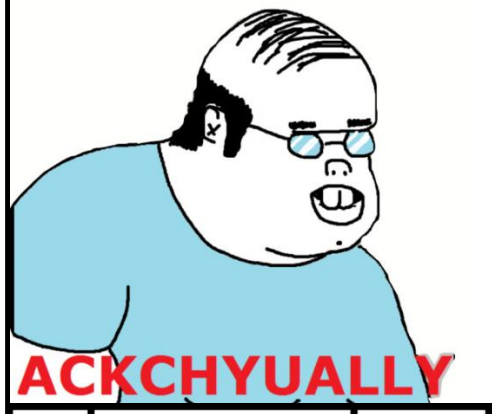
- V1 is too complex
  - Will have deployment issues and bugs for a long time
  - Could lead to people holding off on usage

- V1 is not complex enough
  - Tougher to convince things like IoT/games to switch later on

QUIC is at exactly the right complexity: counterarguments

- HTTP/2 has been buggy for years, still used
- QUIC can evolve very rapidly: V2 will be here soon

- QUIC is meant for the long run
  - But… uptake momentum is important too



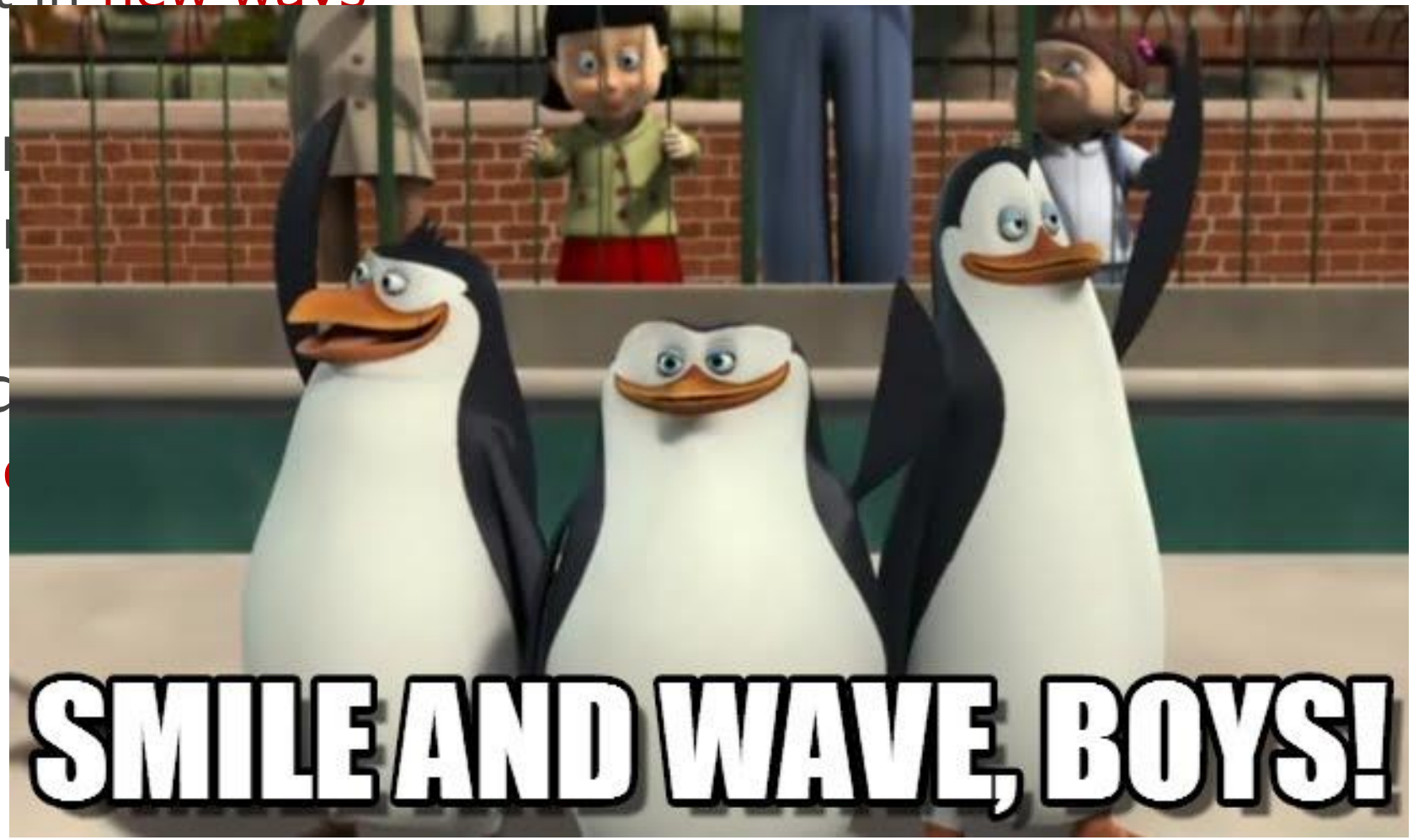
https://github.com/andydavies/http2-prioritization-issues
https://twitter.com/AndyDavies/status/1065916677408346112
https://blog.cloudflare.com/http-2-prioritization-with-nginx/
https://speeder.edm.uhasselt.be/www18/

QUIC uses TLS 1.3, so it's secure

- TLS 1.3 in itself seems valid enough
  - But QUIC uses it in new ways
- Lots of discussion at the IETF this week
  - Key updates, version negotiation, amplification prevention, …
- If attack is found, might need to disable QUIC completely
  - Luckily: easy and fast to update

QUIC uses TLS 1.3, so it's secure

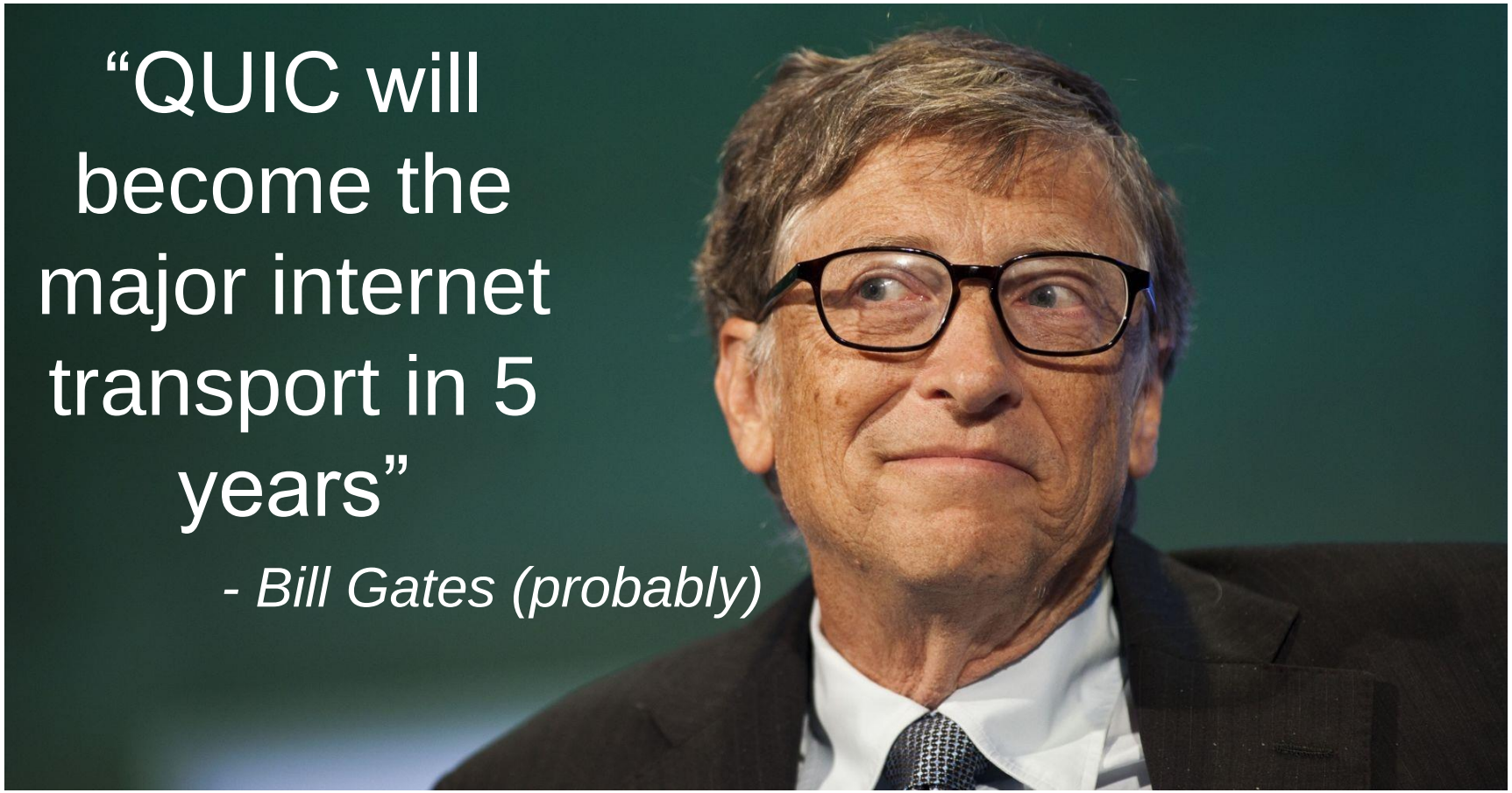
- TLS 1.3 in itself seems valid enough
  - But QUIC uses it in new ways
- Lots of discussion
  - Key updates, ver
- If attack is found
  - Luckily: easy and



Summary of CONFIRMED QUIC FACTS

- QUIC pisses off network and firewall operators
- QUIC is slow and destroys batteries
- QUIC traffic will drown out all TCP flows
- QUIC's 0-RTT is completely useless
- QUIC will incur version negotiation every single time
- QUIC is too complex and not complex enough at the same time.
- QUIC is unsafe and will lead to Trump's re-election

Prediction



bit.ly/quicsurvey

https://calendar.perfplanet.com/2018/quic-and-http-3-too-big-to-fail