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

Robin Marx - @programmingart
PhD researcher – Hasselt University
QUIC and HTTP/3 are going to change the world!

Lucas Pardue @SimmerVigor · Mar 13
Repeating 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 will replace everything, even payment systems and 5G.
QUIC and HTTP/3 *might* change the world!

No one will need more than 637Kb of memory for a personal computer

—*Bill Gates*—
QUIC is special

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

Middlebox “Ossification” prevention

TCP

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
<th>PACKET NR</th>
<th>ACK NR</th>
<th>FLAGS</th>
<th>WINDOW</th>
<th>OPTIONS</th>
<th>ENCRYPTED</th>
<th>DATA</th>
</tr>
</thead>
</table>

QUIC

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
<th>FLAGS</th>
<th>CONNECTION ID</th>
<th>ENCRYPTED</th>
<th>PACKET NR</th>
<th>ACK NR</th>
<th>WINDOW</th>
<th>OPTIONS</th>
<th>DATA</th>
</tr>
</thead>
</table>
QUIC is end-to-end encrypted

Retransmits / Loss / Congestion

Latency / Round Trip Time

Firewall / security logic

Fake them for extra performance

https://youtu.be/TQq6Z4_HBaY?t=1357
QUIC is end-to-end encrypted

Latency / Round Trip Time
Spinbit

Retransmits / Loss / Congestion
Latency / Round Trip Time

Firewall / security logic
Fake them for extra performance

https://youtu.be/TQq6Z4_HBaY?t=1357
QUIC is end-to-end encrypted

<table>
<thead>
<tr>
<th>QUIC</th>
<th>SRC PORT</th>
<th>DEST PORT</th>
<th>FLAGS</th>
<th>CONNECTION ID</th>
<th>ENCRYPTED</th>
</tr>
</thead>
</table>

Latency / Round Trip Time

Retransmits / Loss / Congestion

Latency / Round Trip Time

Firewall / security logic

Fake them for extra performance

https://youtu.be/TQq6Z4_HBaY?t=1357
QUIC is end-to-end encrypted
QUIC is end-to-end encrypted

3 – 5% of networks
QUIC is end-to-end encrypted

3—5% of networks
20+% of networks
QUIC is end-to-end encrypted

3—5% of networks
20+% of networks

TCP fallback

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
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
QUIC is done in Userspace

Desktop

QUIC is done in Userspace

(a) Varying object size, 1% Loss
(b) Varying object size, 112 ms RTT
(e) Nexus6, 1% Loss
(f) Nexus6, 112 ms RTT

QUIC is done in Userspace

58.84%

7.05%

(a) MotoG

(b) Desktop

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**

https://docs.google.com/document/d/16SZDhfR2lspQLQ8s_-FiKBZRgp2WJO2gtDZsWYsNVJ8/edit?usp=sharing

https://datatracker.ietf.org/meeting/104/materials/slides-104-quic-offloading-quic-00
QUIC is done in Userspace: Reprise

Let's do congestion control in JavaScript

https://github.com/rmarx/quicker/tree/congestionControl

QUIC is done in Userspace: Reprise

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

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

TCP + TLS 1.2
- TCP handshake
- TLS handshake
- Data transfer

TCP + TLS 1.3 Early Data
- TCP handshake
- Data transfer

QUIC + TLS 1.3 Early Data

2 RTT
1 RTT
0 RTT

Image: https://www.computer.org/csdl/magazine/ic/2017/02/mic2017020072/13rRUxjyX8n
QUIC cuts down on latency with 0-RTT

TCP + TLS 1.2
2 RTT

TCP + TLS 1.3
1 RTT
Early Data

QUIC + TLS 1.3
0 RTT
Early Data

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

0-RTT HTTP POST
1. Pay Robin $100 for his talk
QUIC cuts down on latency with 0-RTT

0-RTT HTTP POST
1. Pay Robin $100 for his talk

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
QUIC cuts down on latency with 0-RTT

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

Angry Daniel
1.1.1.1

Deserving Robin
2.2.2.2
QUIC cuts down on latency with 0-RTT

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

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

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

UDP Amplification attack: can’t send too much
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!
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 negotiation; 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 negotiation; 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
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 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
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
“QUIC will become the major internet transport in 5 years”
- Bill Gates (probably)

bit.ly/quicsurvey