Measuring QUIC Dynamics over a High Delay Path
IETF-105, Montreal
MAP-RG Meeting
r09

G Fairhurst, Ana Custura, Tom Jones
University of Aberdeen
QUIC vs HTTP/TLS & OpenVPN

QUIC: Debian Linux, quicly v20, reno
TCP: Debian Linux, cubic, SACK, IW10
TCP: Debian Linux, OpenVPN, cubic, SACK, IW10

Time to download a 100k file, n=50

Time to download a 1M file, n=20
QUIC vs HTTP/TLS & OpenVPN

QUIC: Debian Linux, quickly v20, reno
TCP: Debian Linux, cubic, SACK, IW10
TCP: Debian Linux, OpenVPN, cubic, SACK, IW10

Time to download a 100k file, n=50

Time to download a 1M file, n=20
Satellite Residential Broadband
(SLA varies by customer)

- Shared: MF-TDMA network
- Ave. Forward Capacity: 8.5 Mbit/s
- Ave. Return Capacity: 1.4 Mbit/s
- Large BDP: RTT > 550 ms

Measured with `iperf3` using TCP
Hourly RTT Measurements

RTT 580 - 2900 ms, ave. 639 ms

Measured with ping
Packet Number v Time - QUIC vs HTTP+TLS1.3

- **100KB**
  - **QUIC**
  - **TCP/PEP**

- **1 MB**
  - **QUIC**
  - **TCP/PEP**

- **TCP/OpenVPN TLS 1.3**
  - **TCP**
  - **TCP/PEP**
  - **TCP/OpenVPN TLS 1.3**

Times plotted in seconds.
Not always the case: Same download, different behaviour, e.g. the second download takes 10s to complete
Conclusions

• Using quicly was a good experience

• TLS 1.3 has a 2 RTT advantage, more noticeable in small transfers

• Performance of QUIC over satellite not as good as for TCP (with PEP)

• Down-grading to TCP is not a long-term solution

• Need to understand the root causes of performance issues

  • Could be implementation details or need small changes to spec

  • Likely to benefit from new (maybe simple) mechanisms…
Future Plans

• We will continue measurements

  • Happy to talk about logging and tracing!

  • Aware there are many different satellite systems!
Extra Slides
Test setup

- QUIC on Debian Linux
  - QUIC 20
  - RENO
  - IW10 (12800 B)
  - MSS 1460

- HTTP over TLS 1.2/1.3 on Debian Linux
  - SACK, W Scaling
  - CUBIC
  - IW 20 (29200 B) / IW 10 (14600 B)
  - MSS 1460

- HTTP over TLS 1.2/1.3 over OpenVPN
  - SACK, W Scaling
  - CUBIC
  - IW 20 (27160 B) / IW 10 (13580)
  - MSS 1358
Measurements

- We use vagrant to configure and start a test virtual machine,
- This compiles and builds quicly from source, and also installs openvpn-client
- The machine is bridged on the satellite network and runs scripts that:
  1. start tcpdump, writing pcap files to a shared results folder
  2. perform a wget request to a webserver hosted by the University of Aberdeen
  3. stop tcpdump
  4. performs a quicly download from a quicly webserver hosted by the University of Aberdeen, saving json logs to a shared result folder
  5. sets up openvpn
  6. performs steps 1 -> 3
  7. stop openvpn
  //Rinse and repeat every hour and with different file sizes etc.
  8. In parallel, the webservers continuously capture packets and log quicly webserver interactions.
- The pcap traces were analyzed with python3-libtrace, which allows access to the IP and TCP layers. The quicly logs are json and therefore easily parsed - they provide fields for PNs, ACKs, CWND and timestamps.
- We use python3-matplotlib to plot the data extracted from traces and logs.
Forward path
Packet Number v Time - QUIC vs HTTP+TLS1.2

100KB

QUIC

TCP SN over time

Time (seconds)

0.0 1.0 2.0 3.0 4.0 5.0 6.0

SN/MSS

0 20 40 60

1 MB

QUIC

TCP SN over time

Time (seconds)

0.0 5.0 10.0 15.0 20.0

SN/MSS

0 200 400 600

TCP/PEP

OpenVPN SN over time

Time (seconds)

0.0 1.0 2.0 3.0 4.0 5.0 6.0

SN/MSS

0 20 40 60

TCP

OpenVPN SN over time

Time (seconds)

0.0 5.0 10.0 15.0 20.0

SN/MSS

0 200 400 600

TCP/OpenVPN
TCP and TCP over OpenVPN - 1MB downloads

- Connection setup with TLS 1.2 - adds 2x RTT
TCP and TCP over OpenVPN
- 100KB downloads

- Connection setup with TLS 1.2 - adds 2x RTT
QUIC - 1MB downloads

- Not always the case: Same download, different behaviour, e.g. the second download takes 10s to complete
QUIC RTT and CWND

**100KB**
QUIC client CWND over time, 100k download

- Server
- Client

**1M**
QUIC client CWND over time, 1M download

Queue delay?

CWND just grows forever

QUIC - RTT over time, 100k

QUIC - RTT over time, 1M
TCP RTT and CWND

TCP CWND over time, 100k download

TCP CWND over time, 1M download

TCP RTT over time
OpenVPN RTT and CWND

100KB
TCP CWND over time, 100k download

1M
TCP CWND over time, 1M download

TCP/OpenVPN RTT over time

TCP RTT over time
Return path
TCP and TCP over OpenVPN - 1M Requests

- The horizontal lines were ACKs all along
QUIC 1M Requests

- ACKs for every packet, closely following data received from server
- Cannot pinpoint where the GET request and initial crypto happen