Validation of QUIC protocol usefulness in interactive communication

Michal Sledz

June 4, 2020

1 Abstract

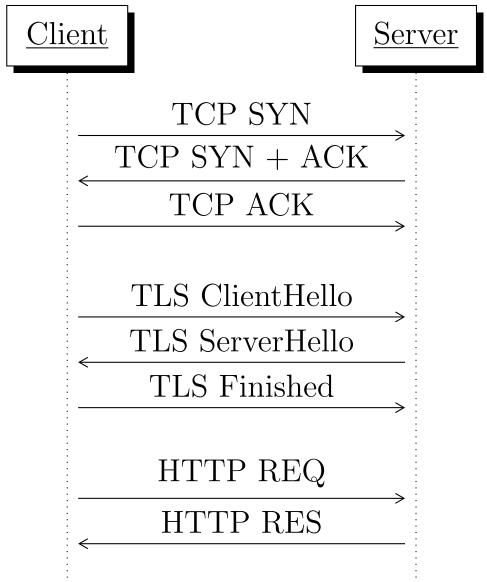
The goal of this thesis is to examine usefulness of QUIC transport protocol in an interactive communication. QUIC is a new transport protocol based on UDP protocol. With it there comes new HTTP version - HTTP/3. Both QUIC and HTTP/3 are not standardize yet but there exists some implementations of IETF draft versions which are successfully used in the production. Therefore it is a good moment to examine if QUIC suits well and gains new advantages in an interactive communication in which neither TCP nor UDP protocols perform well enough. In this purposes Rust implementation of QUIC has been chosen. Results include comparision between TCP and QUIC protocols in terms of performance e.g. connection setup time and reliability.

2 Introduction

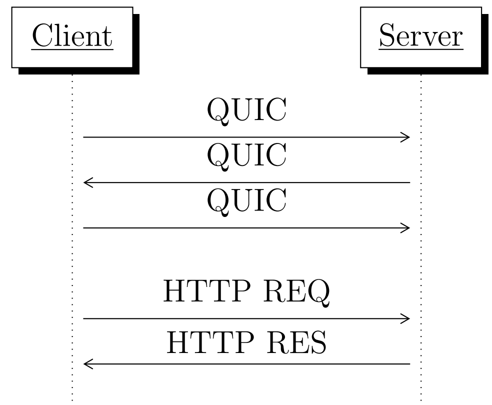
Transport layer of ISO/OSI model contains two most significant protocols in terms of Internet communication – TCP and UDP .

TCP is a connection based protocol which handles congestion control, sends confirmations when message is received or cannot be received and guarantees proper order of sending packets. Because TCP is a point to point protocol we are not allowed to realize multicast transmission using it. The biggest drawback of this protocol is its heaviness. Connection setup requires so called three way handshake which takes some time. Lack of any built-in security mechanisms forces us to use dedicated to this purpose TLS protocol which introduces its own connection handshake. This causes that connection setup becomes even less efficient. On the other hand there is a very lightweight, connectionless protocol called UDP . It does not guarantee proper order of sending packets. There are not also any confirmations or warnings sent in case of communication errors. Everything is performance oriented. UDP provides also multicast mechanism. According to this UDP is a good choice when we need very fast and lightweight communication but we can lose some packets. A good example of UDP destination is a video streaming.

We can see that these two protocols and their destinations are strongly opposite. And here comes QUIC – new transport layer protocol based on UDP which is intended to replace TCP providing higher performance and



(a) HTTP request in TCP



(b) HTTP request in QUIC

Figure 1: HTTP request comparison

even better reliability. First of all QUIC is user level protocol not system level one. It means that we can implement it without modifying operating systems kernels. Secondly QUIC is a secure protocol. It ensure data encryption and authentication which are handled by some other protocols like TLS . QUIC provides support for these protocols. Figure 1 presents comparison between HTTP request made in TCP and QUIC protocols. Typical QUIC handshake takes a single round-trip between hosts when TCP one requires two round-trips.

Another enhancement in QUIC is resistance for network changes. User can change its Wi-Fi network without performing a new client-server handshake. This is achieved by special ID's QUIC associates with each connection.

Results section describes performance differences between communication in TCP and QUIC protocols including connection time setup and reliability.

3 Related Work

Along with QUIC there come a new HTTP protocol version called HTTP/3. Both of them are not standardized yet but there are more and more IETF drafts describing these protocols. The last one was published on February and expires on August [1]. A lot of companies and scientists explores and tests QUIC and HTTP/3 features. For example in [4] QUIC was tested in terms of performance under network congestion. Therefor it is important to continuously examine new implementations and their.

4 TCP and QUIC Benchmarks

We compare TCP and QUIC protocols be executing simple HTTP GET request to https://quic.tech:8443/ address. Under this URL is hosted server which can handle both TCP and QUIC requests.

4.1 Benchmarks

Following benchmarks were performed:

- overall request time
- connection establishment time
- number of TCP/UDP packets sent
- overall size of UDP datagrams and TCP segments sent

4.2 Environment

Network bandwidth:

- Download: 77 Mb/s
- Upload: 7.7 Mb/s
- Ping (Wysiadw to Warsaw): 25 ms

4.3 Code

4.3.1 TCP and HTTP/2

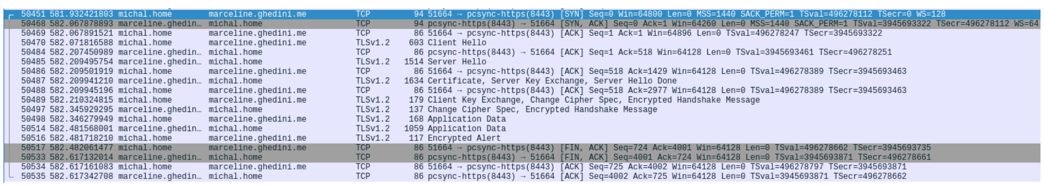
Code implementing request with TCP was written using `curl-rust` [3] which provides Rust bindings for libcurl library [2].

4.3.2 QUIC and HTTP/3

Code implementing request with QUIC was taken from examples section of cloudflare github repository [5].

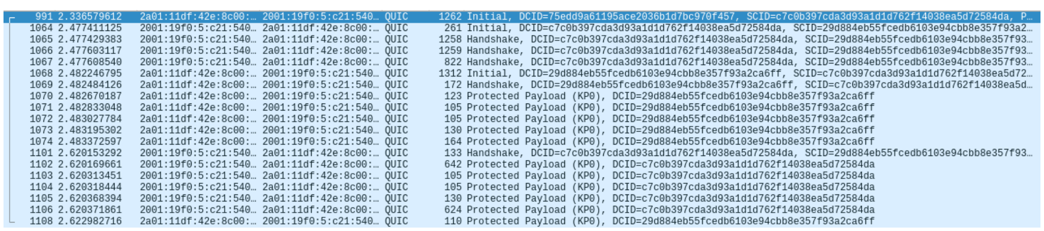
4.4 Wireshark

This subsection presents example wireshark dumps from executing request with TCP and QUIC.



50451	581.932421803	michal.home	marceline.ghedini.me	TCP	94	51664 → pcsync-https(8443) [SYN] Seq=0 Win=64800 Len=0 MSS=1440 SACK_PERM=1 TSval=496278112 TSecr=0 WS=128
50468	582.067878893	marceline.ghedin…	michal.home	TCP	94	pcsync-https(8443) → 51664 [SYN, ACK] Seq=0 Ack=1 Win=64260 Len=0 MSS=1440 SACK_PERM=1 TSval=3945693322 TSecr=496278112 WS=64
50469	582.067891521	michal.home	marceline.ghedini.me	TCP	86	51664 → pcsync-https(8443) [ACK] Seq=1 Ack=1 Win=64896 Len=0 TSval=496278247 TSecr=3945693322
50470	582.071816588	michal.home	marceline.ghedini.me	TLSv1.2	603	Client Hello
50484	582.207450989	marceline.ghedin…	michal.home	TCP	86	pcsync-https(8443) → 51664 [ACK] Seq=1 Ack=518 Win=64128 Len=0 TSval=3945693461 TSecr=496278251
50485	582.209495754	marceline.ghedin…	michal.home	TLSv1.2	1514	Server Hello
50486	582.209501919	michal.home	marceline.ghedini.me	TCP	86	51664 → pcsync-https(8443) [ACK] Seq=518 Ack=1429 Win=64128 Len=0 TSval=496278389 TSecr=3945693463
50487	582.209941210	marceline.ghedin…	michal.home	TLSv1.2	1634	Certificate, Server Key Exchange, Server Hello Done
50488	582.209945196	michal.home	marceline.ghedini.me	TCP	86	51664 → pcsync-https(8443) [ACK] Seq=518 Ack=2977 Win=64128 Len=0 TSval=496278389 TSecr=3945693463
50489	582.210324815	michal.home	marceline.ghedini.me	TLSv1.2	179	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
50497	582.345929295	marceline.ghedin…	michal.home	TLSv1.2	137	Change Cipher Spec, Encrypted Handshake Message
50498	582.346279949	michal.home	marceline.ghedini.me	TLSv1.2	168	Application Data
50514	582.481568001	marceline.ghedin…	michal.home	TLSv1.2	1059	Application Data
50516	582.481718210	michal.home	marceline.ghedini.me	TLSv1.2	117	Encrypted Alert
50517	582.482061477	michal.home	marceline.ghedini.me	TCP	86	51664 → pcsync-https(8443) [FIN, ACK] Seq=724 Ack=4001 Win=64128 Len=0 TSval=496278662 TSecr=3945693735
50533	582.617132014	marceline.ghedin…	michal.home	TCP	86	pcsync-https(8443) → 51664 [FIN, ACK] Seq=4001 Ack=724 Win=64128 Len=0 TSval=3945693871 TSecr=496278661
50534	582.617161083	michal.home	marceline.ghedini.me	TCP	86	51664 → pcsync-https(8443) [ACK] Seq=725 Ack=4002 Win=64128 Len=0 TSval=496278797 TSecr=3945693871
50535	582.617342708	marceline.ghedin…	michal.home	TCP	86	pcsync-https(8443) → 51664 [ACK] Seq=4002 Ack=725 Win=64128 Len=0 TSval=3945693871 TSecr=496278662

Figure 2: Example wireshark dump from TCP GET request



991	2.336579612	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	1262	Initial, DCID=75edd9a61195ace2036b1d7bc970f457, SCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da, P…
1064	2.477411125	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	261	Initial, DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da, SCID=29d884eb55fcedb6103e94cbb8e357f93a2…
1065	2.477429383	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	1258	Handshake, DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da, SCID=29d884eb55fcedb6103e94cbb8e357f93…
1066	2.477603117	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	1259	Handshake, DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da, SCID=29d884eb55fcedb6103e94cbb8e357f93…
1067	2.477608540	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	822	Handshake, DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da, SCID=29d884eb55fcedb6103e94cbb8e357f93…
1068	2.482246795	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	1312	Initial, DCID=29d884eb55fcedb6103e94cbb8e357f93a2ca6ff, SCID=c7c0b397cda3d93a1d1d762f14038ea5d72…
1069	2.482484126	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	172	Handshake, DCID=29d884eb55fcedb6103e94cbb8e357f93a2ca6ff, SCID=c7c0b397cda3d93a1d1d762f14038ea5d…
1070	2.482670187	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	123	Protected Payload (KP0), DCID=29d884eb55fcedb6103e94cbb8e357f93a2ca6ff
1071	2.482833048	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	105	Protected Payload (KP0), DCID=29d884eb55fcedb6103e94cbb8e357f93a2ca6ff
1072	2.483027784	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	105	Protected Payload (KP0), DCID=29d884eb55fcedb6103e94cbb8e357f93a2ca6ff
1073	2.483195302	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	130	Protected Payload (KP0), DCID=29d884eb55fcedb6103e94cbb8e357f93a2ca6ff
1074	2.483372597	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	164	Protected Payload (KP0), DCID=29d884eb55fcedb6103e94cbb8e357f93a2ca6ff
1101	2.620153292	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	133	Handshake, DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da, SCID=29d884eb55fcedb6103e94cbb8e357f93…
1102	2.620169661	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	642	Protected Payload (KP0), DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da
1103	2.620313451	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	105	Protected Payload (KP0), DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da
1104	2.620318444	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	105	Protected Payload (KP0), DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da
1105	2.620368394	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	130	Protected Payload (KP0), DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da
1106	2.620371861	2001:19f0:5:c21:540…	2a01:11df:42e:8c00:…	QUIC	624	Protected Payload (KP0), DCID=c7c0b397cda3d93a1d1d762f14038ea5d72584da
1108	2.622982716	2a01:11df:42e:8c00:…	2001:19f0:5:c21:540…	QUIC	110	Protected Payload (KP0), DCID=29d884eb55fcedb6103e94cbb8e357f93a2ca6ff

Figure 3: Example wireshark dump from QUIC GET request

4.5 Results

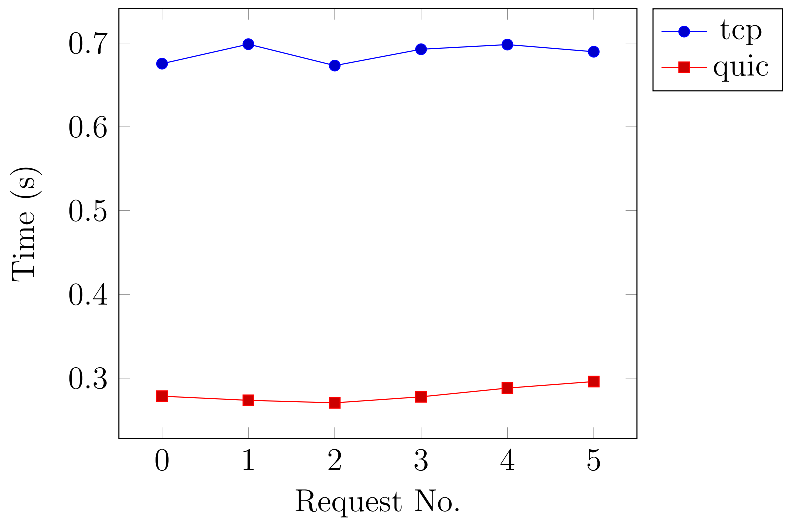


Figure 4: Overall http GET request time with connection fin in case of TCP

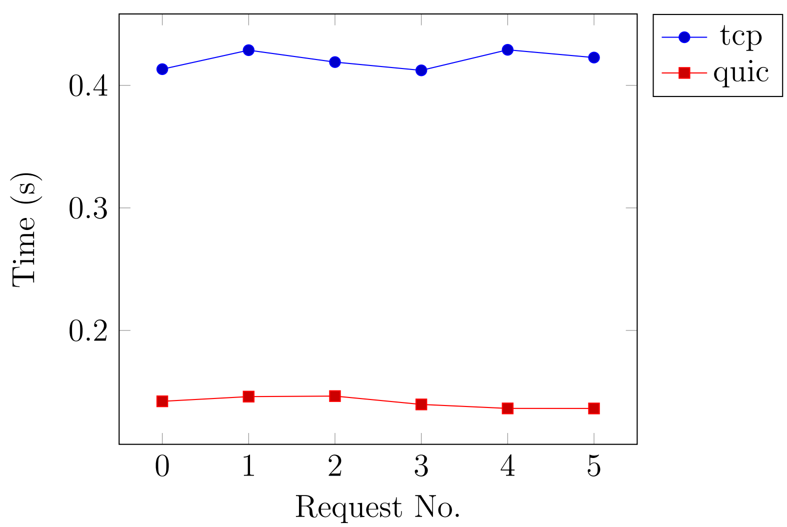


Figure 5: Connection establishing time

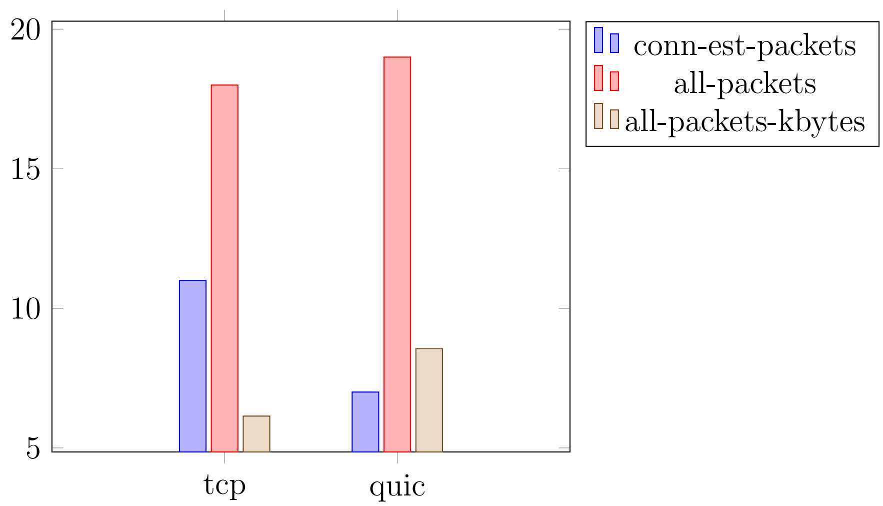


Figure 6: Packets summary

References

[1] Hypertext transfer protocol version 3 (http/3).

[2] libcurl - the multiprotocol file transfer library.

[3] libcurl bindings for rust.

[4] Performance analysis of quic protocol under networkcongestion.

[5] Savoury implementation of the quic transport protocol and http/3.