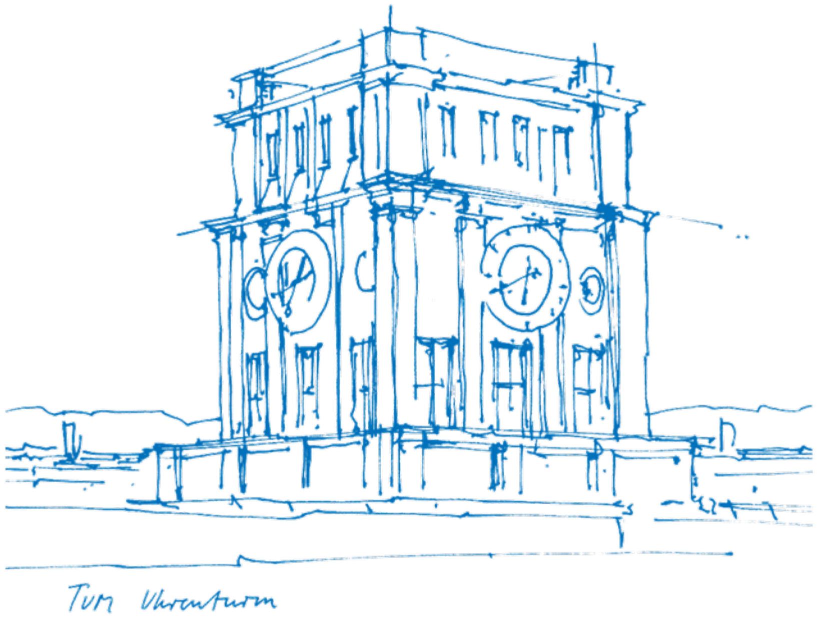
Chair of Connected Mobility
Department of Informatics
Technical University of Munich

Evaluating Web and Video Content Delivery over Google QUIC

Rohit Panda
rohit.panda@tum.de

Advisor: Dr. Vaibhav Bajpai
bajpaiv@in.tum.de

Supervisor: Prof. Dr. Jörg Ott
ott@in.tum.de



München, 19. Februar 2019

Motivation

- Web latency and security are becoming more and more important.
- Middleboxes have caused a protocol ossification. Attempted extensions to TCP like TCP Fast Open[RCC+11], SCTP[YO02], Multipath TCP [FRHB13] have not seen widespread deployment.
- QUIC is a user-space transport protocol built on top of UDP. Also QUIC provides for authen ticated, encrypted header and payload which avoids dependency on vendors and ISPs
- Google started work on QUIC in 2012.
- Faster development and deployment cycles. Already on Q044 version and accounts for more than 35% of Google's total egress traffic and 7% of global internet traffic.
- QUIC has been shown to reduce search latency by 8% for desktop users and 3.6% for mobile users.[LRW+17]
- An IETF working group has been formed to standardize QUIC which should lead to even greater adoption.

Goal and Approach

- Initial performance results from Google showed great gains but there is still a lack of repeatable studies.
- Rapid development cycle of QUIC means that many new features are introduced in each version of QUIC and support for older versions is dropped. So previous studies become obsoleted quickly.
- In this thesis we aim to evaluate the performance of QUIC(Q035, Q039, Q043, Q044) and compare it with TCP/TLS.
- To do this we conduct experiments on real Internet to investigate connection establishment time, Time to first byte, download time, YouTube QoE metrics, Throughput, fairness in competing flow scenarios and CPU utilization.

Goal and Approach

- **RQ1 : How do different versions of QUIC compare with each other and TCP/TLS 1.2 and TCP/TLS 1.3 on IPv4 and IPv6?**
- **RQ2 : How do the versions of QUIC compare with each other on different Autonomous Systems?**
- **RQ3 : How does throughput and CPU utilization of QUIC compare with TCP/TLS?**
- **RQ4 : Is QUIC fair towards TCP/TLS?**

Background

SPDY is the first attempt by Google to improve Web Latency.[spd] HTTP/2 is the successor of SPDY protocol.

It has features like multiplexing requests over a single TCP connection, HTTP header compression, request prioritization, server push and server hint[BPT15]

HTTP/2 suffers from the TCP head-of-line blocking issue.

Previous work [ETW14][CSO16][CCM15][MKM16] shows the improvements provided by SPDY and HTTP/2.

QUIC has many innovative features like Version Negotiation, 0-RTT connection establishment, Reduced head-of-line blocking, improved congestion control using packet pacing, Authenticated, encrypted header and payload, stream multiplexing, Connection-IDs, stream and connection level flow control, connection migration and resilience to NAT rebinding.[IT19]

Background

QUIC replaces most of the traditional HTTPS stack: TCP, TLS, HTTP/2.

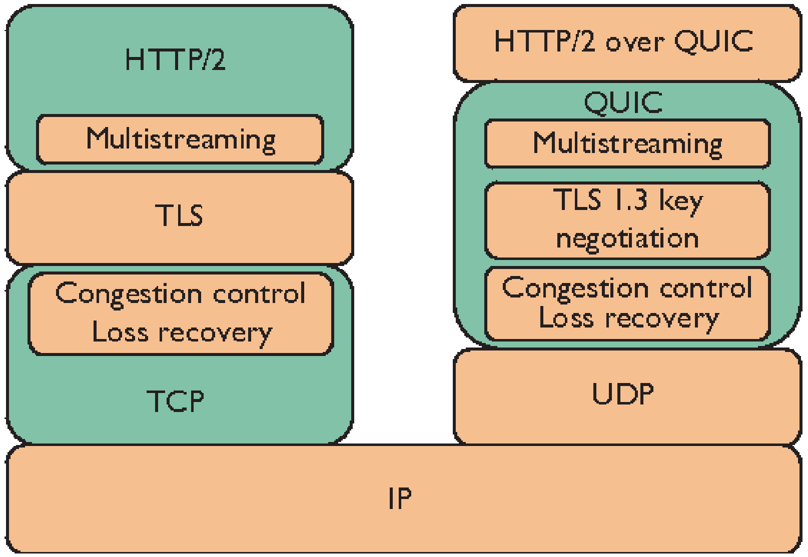


Abbildung: Layered QUIC architecture in the traditional HTTPS stack. QUIC incorporates features of congestion control, loss recovery similar to TCP, Multiple streams like HTTP2 and key negotiation like TLS.[CLL+17a]

Connection Establishment[LRW$^+$17][quib][cry]

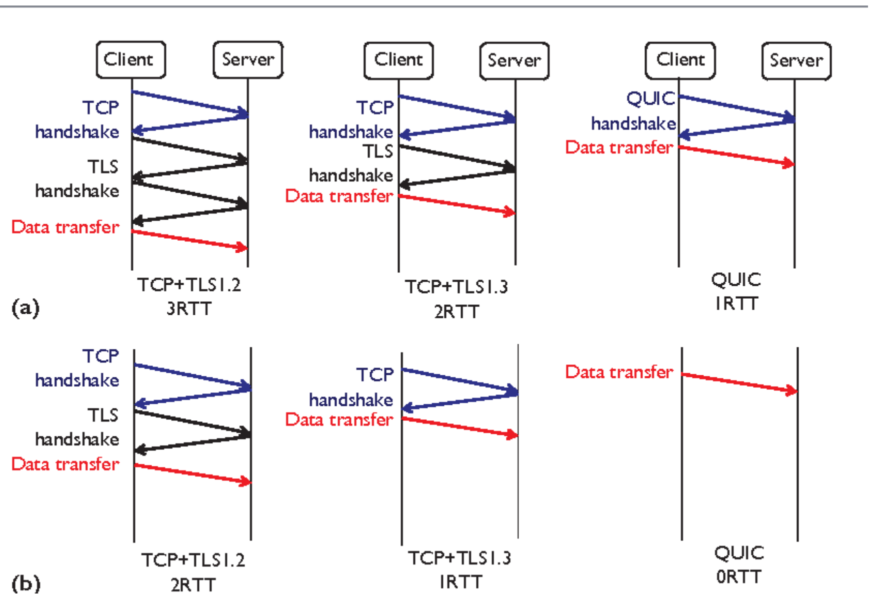


Abbildung: Handshake round-trip time (RTT) of different protocols. (a) First-time connection establishment. (b) Subsequent connections.[CLL$^+$17a]

Multiplexing

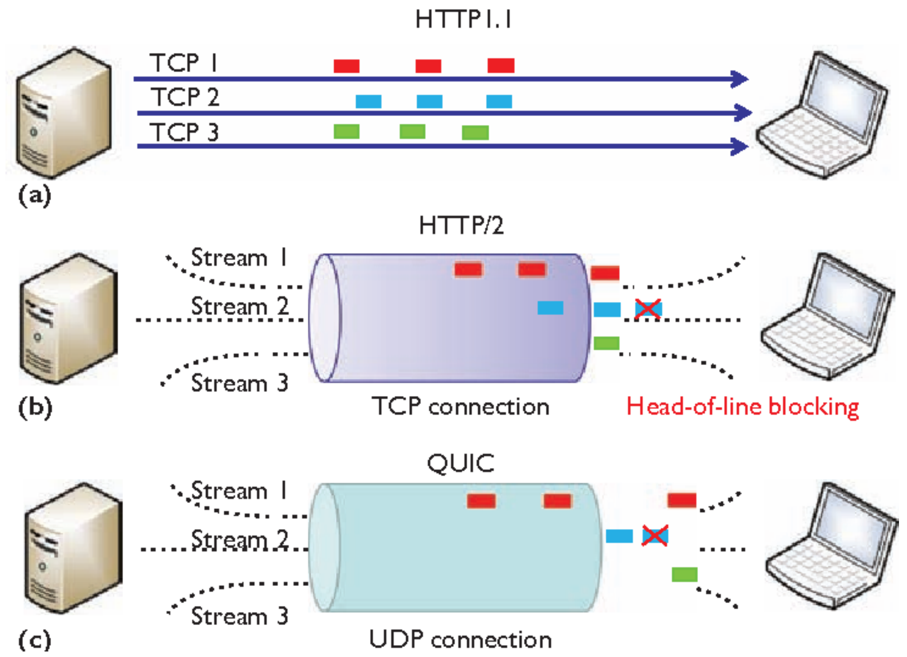


Abbildung: Multiplexing comparison. This involved sending multiple streams of data over a single transport connection using (a) HTTP1.1, (b) HTTP/2, and (c) QUIC.[CLL+17a]

QUIC also supports multiple streams in a single connection but the packets can be delivered out of order and a lost packet affects only those streams whose data it carried, not affecting other streams.

Congestion Control + Packet Pacing

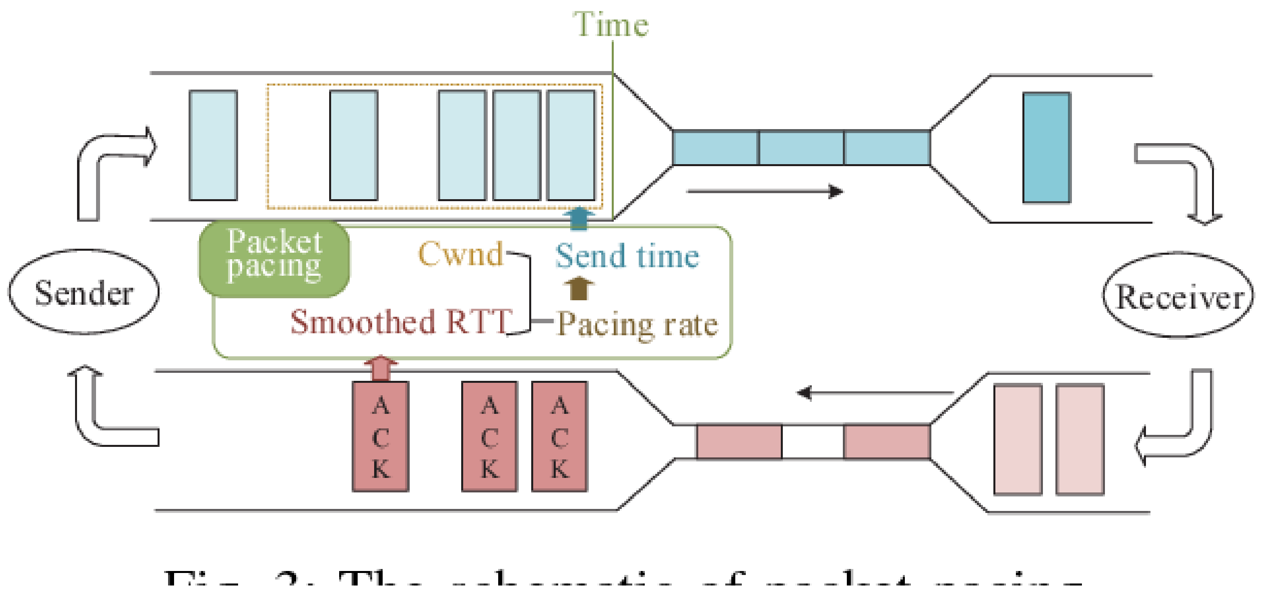


Abbildung: Packet Pacing [YXY17]

Default congestion control algorithm is CUBIC[CLL$^+$17b] but differs from TCP CUBIC slightly.[quib][IS19]. BBR is a new congestion control algorithm being developed by Google.[LCJC18]

Packet Pacing is done by inserting a waiting time between each UDP datagram (enabled by default)
This helps lower retransmission rate [ASA00]

Prior Work

Langley et al. [LRW$^+$17] discuss about motivations in various design decisions of QUIC, development and testing performed over various iterations, performance improvements. They provide us with the first large-scale measurements of QUIC across various versions.

Nepomuceno et al.[NdOA$^+$18], Cook et al.[CMTH17] look at page load time.

Li et al.[LCJC18], Kharat et al.[KRGK18], Qian et al. [QWT18], Kakhki et al.[KJC$^+$17]look at throughput, congestion control and fairness among flows.

Yu et al. [YXY17] focused mainly on packet pacing mechanism for congestion control.

Saverimoutou et al.[SMV17], Fischlin et al.[FG14] and Vaere et al.[VBKT18] look into the security aspects of QUIC.

Willem et al.[dB] and Edeline et al. [EKT$^+$16] look into how QUIC being built on top of UDP affects it.

Wang et al.[WBRP18] implement QUIC in the linux kernel and perform experiments to compare both in kernel mode.

Ruth et al.[RPDH18] look into QUIC usage in the wild.

Methodology

We look at metrics like connection establishment time(For QUIC this will be 1-RTT Handshake, for TCP/TLS1.3 2-RTT and for TCP/TLS1.2 3-RTT), time to first byte and page download time using `quic_perf` and `tls_perf`.

We analyze these metrics on AS level(announces prefix of our target websites) using RIPE Database.[RIP]

YouTube QoE metrics like like Connect Time to media CDNs, Prebuffering time, Startup delay and Overall download rate using `YouTube tests`

Throughput and CPU utilization using `YouTube tests`, `quic_perf` and `tls_perf`.

CUBIC congestion control fairness in cases of competing flows by running the tests concurrently.

We use `perf` and `strace` to profile QUIC and explain its high CPU utilization.

We investigate QUIC Discovery using Alt-Svc Header

Methodology

Reused the `YouTube test` for TCP developed by S Ahsan et al.[ABOS15] which was modified to use QUIC by Sergey[Pod] and `tls_perf` and `quic_perf` by Bernhard[Jae]. Several additions and improvements were made to their tests like adding support for Q044, improving logging and error handling, adding HTTP/2 support.

Used the latest version of `litespeed lsquic client` and `Boringssl` library for QUIC implementation.

`libcurl` was used for TCP implementation.

For measurements over IETF QUIC we used the QUIC Tracker [quia] test suite which supports QUIC(draft-17) and is also TLS1.3 compatible.

Created debian packages using cpack for easier deployment over Raspberry PI.

List of target websites for our measurements was taken from QUIC Research@COMSYS[com]

Top 50 most popular videos for a region were selected for measurements over YouTube and files of different sizes were uploaded to Google Drive for performance tests

Measurement Setup

Measurements were performed from 3 vantage Points.

We used the VM `vmott16.cm.in.tum.de` with linux OS. This machine has over 400 Mbits/s UDP connection speed with Microsoft Azure virtual server and 1.20 Gbits/s via TCP. It supported both IPv4 and IPv6.

We used a Raspberry Pi 3 Model B with 1 GB RAM and 32 GB microSDHC Card. This PI was installed at a residencial location connected to a LAN network run by the Leibniz-datacenter(LRZ). It was running on a 100 Mbits/s line.

Another Raspberry PI was installed in Bhubaneswar, India. It was running on 20 Mbits/s line but peak bandwidth was measured at 14 Mbit/s during testing.

Only IPv4 could be tested at the above two locations because of lack of IPv6 support

The tests were coded in C as libraries used provide us with an easy-to-use C API. The IETF test were coded in Go.

Overview of Thesis Results

Conclusion

Future work and limitations

- Geographical bias due to location of probes.For more representative results the tests should be deployed at multiple locations with the help of CAIDA Ark [CAI] measurement infrastructure and SamKnows project[Sam]
- Mobile devices are a major source of QUIC traffic so tests need to be adopted for mobile devices using cronet[cro].
- No support for 0-RTT connections in lsquic at the time of test creation. Support has been introduced in latest release.
- The QUIC `YouTube tests` have a higher failure rate than TCP `YouTube tests` which can be looked into.
- Other features of QUIC like Congestion control window size, Packet pacing, Flow control can be evaluated.
- QUIC tests can be performed under constrained conditions like low buffer size, introducing random bandwidth changes, random packet loss and different RTTs.
- Other metrics like no. of retransmissions, no. of packets lost, congestion window size can be measured.
- IETF QUIC may be standardized soon, it may also be considered for comparison with Google QUIC and TCP/TLS.

References

[ABOS15] Saba Ahsan, Vaibhav Bajpai, Jörg Ott, and Jürgen Schönwälder, *Measuring youtube from dual-stacked hosts*, Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings (Jelena Mirkovic and Yong Liu, eds.), Lecture Notes in Computer Science, vol. 8995, Springer, 2015, pp. 249–261.

[ASA00] Amit Aggarwal, Stefan Savage, and Thomas E. Anderson, *Understanding the performance of TCP pacing*, Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv, Israel, March 26-30, 2000, IEEE, 2000, pp. 1157–1165.

[BPT15] Mike Belshe, Roberto Peon, and Martin Thomson, *Hypertext Transfer Protocol Version 2 (HTTP/2)*, RFC 7540, May 2015.

[CAI] *Caida ark*.

[CCM15] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo, *HTTP over UDP: an experimental investigation of QUIC*, Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015 (Roger L. Wainwright, Juan Manuel Corchado, Alessio Bechini, and Jiman Hong, eds.), ACM, 2015, pp. 609–614.

[CLL$^+$17a] Yong Cui, Tianxiang Li, Cong Liu, Xingwei Wang, and Mirja Kuhlewind, *Innovating transport with QUIC: Design approaches and research challenges*, IEEE Internet Computing **21** (2017), no. 2, 72–76.

[CLL$^+$17b] Yong Cui, Tianxiang Li, Cong Liu, Xingwei Wang, and Mirja Kühlewind, *Innovating transport with QUIC: design approaches and research challenges*, IEEE Internet Computing **21** (2017), no. 2, 72–76.

[CMTH17] Sarah Cook, Bertrand Mathieu, Patrick Truong, and Isabelle Hamchaoui, *QUIC: better for what and for whom?*, IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017, IEEE, 2017, pp. 1–6.

[com] *Quic research @ comsys*.

[cro] *cronet*.

Questions?