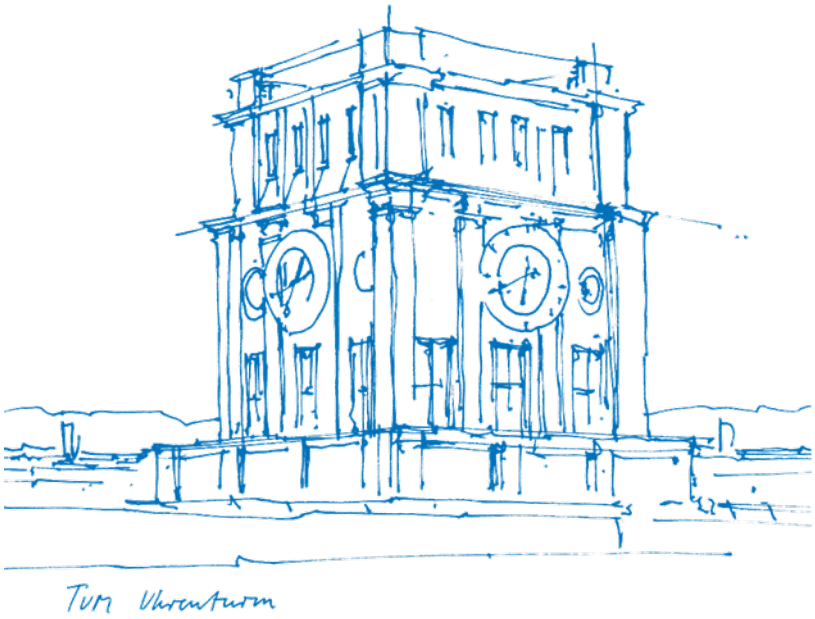
Chair of Connected Mobility
Department of Informatics
Technical University of Munich

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[31], SCTP[39], Multipath TCP [15] 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.[24]
- 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.[36] 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[3]

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

Previous work [13][8][6][26] 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.[20]

Background

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

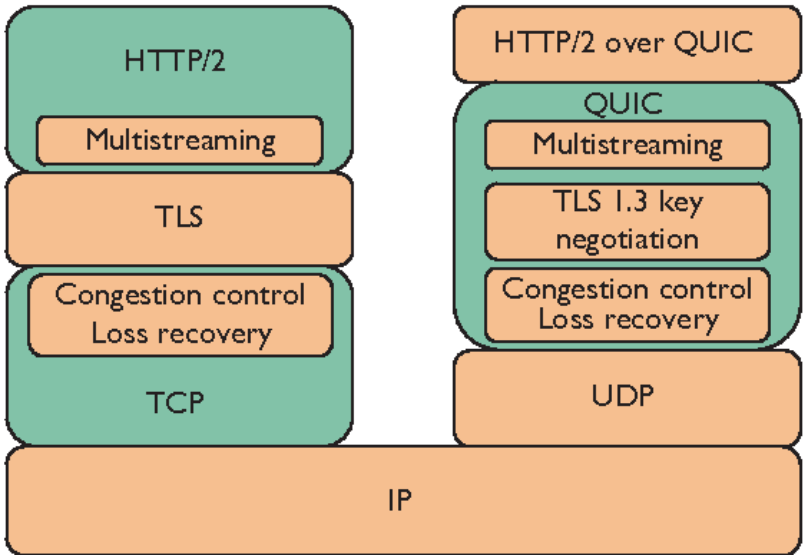


Abbildung: Layered QUIC architecture in the traditional HTTPS stack. QUIC incorporates features of congestion control, loss recovery similar to TCP, Multiple streams like HTTP2 and key negotiation like TLS.[10]

Connection Establishment[24][18][17]

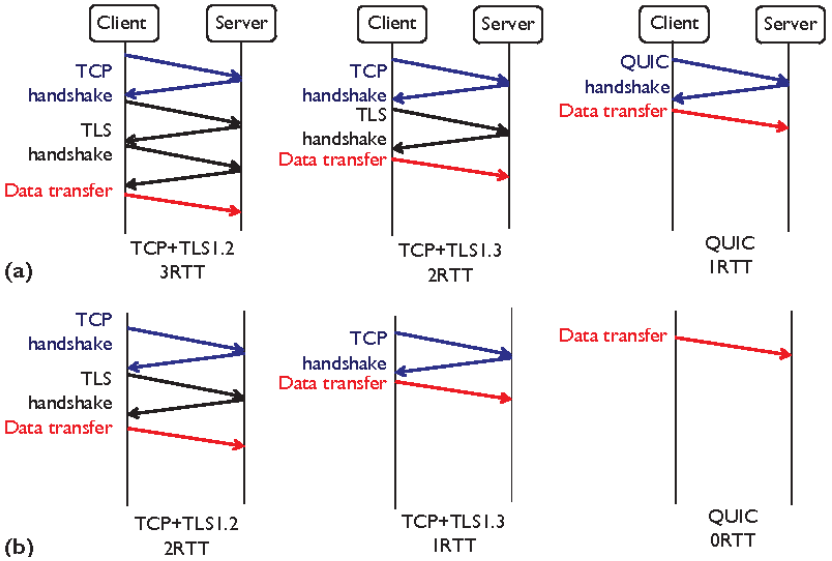


Abbildung: Handshake round-trip time (RTT) of different protocols. (a) First-time connection establishment. (b) Subsequent connections.[10]

Multiplexing

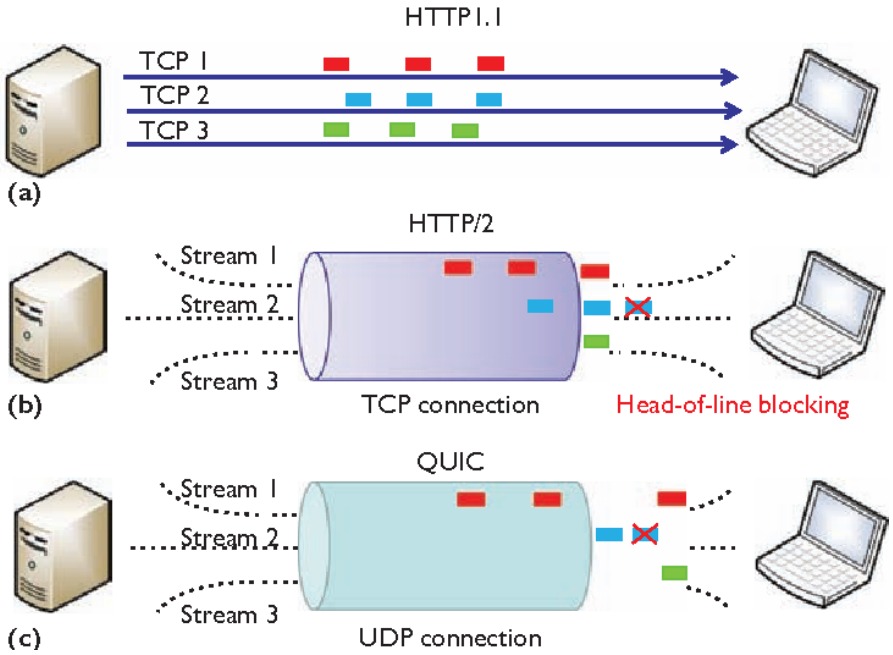


Abbildung: Multiplexing comparison. This involved sending multiple streams of data over a single transport connection using (a) HTTP1.1, (b) HTTP/2, and (c) QUIC.[10]

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

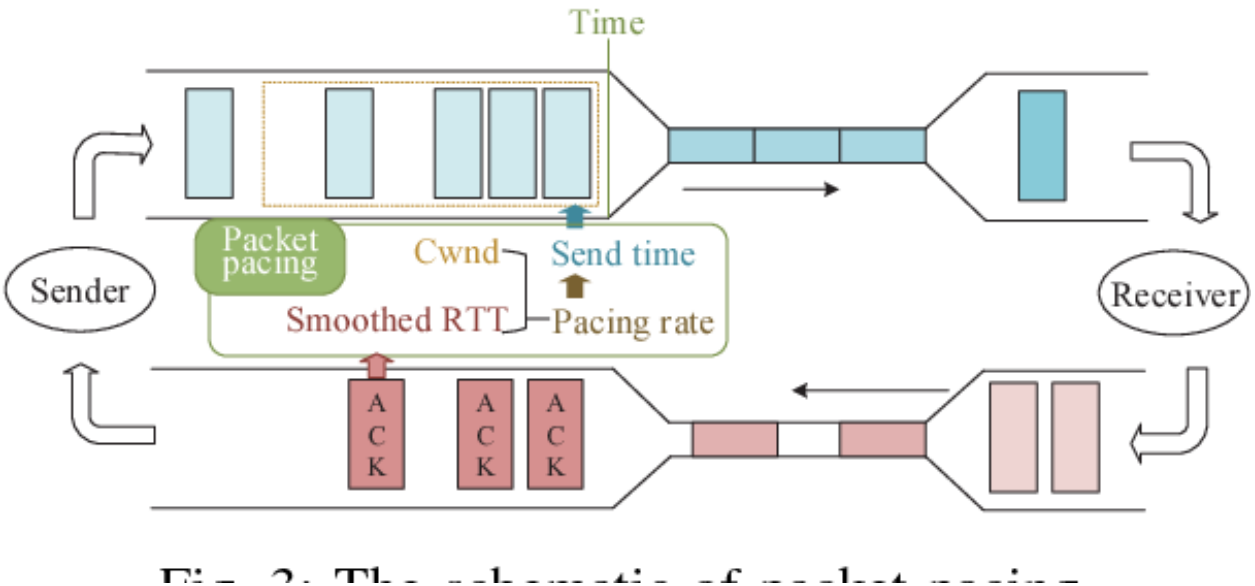


Abbildung: Packet Pacing [40]

Default congestion control algorithm is CUBIC[11] but differs from TCP CUBIC slightly.[18][19]. BBR is a new congestion control algorithm being developed by Google.[25]

Packet Pacing is done by inserting a waiting time between each UDP datagram (enabled by default) This helps lower retransmission rate [1]

Prior Work

Langley et al. [24] 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.[27], Cook et al.[7] look at page load time.

Li et al.[25], Kharat et al.[23], Qian et al. [29], Kakhki et al.[22]look at throughput, congestion control and fairness among flows.

Yu et al. [40] focused mainly on packet pacing mechanism for congestion control.

Saverimoutou et al.[35], Fischlin et al.[14] and Vaere et al.[37] look into the security aspects of QUIC.

Willem et al.[4] and Edeline et al. [12] look into how QUIC being built on top of UDP affects it.

Wang et al.[38] implement QUIC in the linux kernel and perform experiments to compare both in kernel mode.

Rüth et al.[33] 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.[32]

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.[2] which was modified to use QUIC by Sergey[28] and `tls_perf` and `quic_perf` by Bernhard[21]. 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 [16] 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[30]

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 I

When analyzing the connection establishment times for different versions of QUIC, in case of IPv4 we found that at 90th percentile Q044 performs slightly better by about 55 ms. As expected QUIC performs better than TLS 1.2 and TLS 1.3. At 50th percentile QUIC versions have similar latency but lower than TLS 1.2 by 34 ms and TLS 1.3 by 77 ms. In case of IPv6 all of the QUIC versions have quite similar values. Difference with TLS 1.2 remains similar to IPv4 but TLS 1.3 has lower connection times than TLS 1.2. But when we analyze Time to first byte we see that QUIC loses some of the improvements gained by 1-RTT connection times, it is still faster than TLS. At 50th percentile Q035, Q039, Q043 have a TTFB of around 206 ms, Q044 has a TTFB of 211 ms, TLS 1.2 has a TTFB of 261 ms and TLS 1.3 has a TTFB of 360 ms. Difference in total download time is similar to TTFB.

From the measurements done in India we find greater improvements for QUIC over TCP/TLS than in Munich which shows that benefits of QUIC is more evident in low-bandwidth, high-loss networks and high-RTT regions. QUIC provides improvements of 240 ms, 340 ms over TLS 1.2, TLS 1.3 respectively at 75th percentile in India while it provides improvements of 210 ms, 245 ms over TLS 1.2, TLS 1.3 respectively in Munich for connection establishment time. These measurements are only for IPv4.

Conclusion II

In conclusion Q044 may have slightly better Connection times but TTFB and download time between different QUIC versions is similar.

In case of Video content delivery over YouTube, QUIC provides much better improvements in India than Germany for Connect Time, with an improvement of 550 ms at 50th percentile over TLS, for Germany the value is 410 ms. For Prebuffering time, the gap between QUIC and TLS is 447 ms in Munich and 600 ms in India. For Startup delay, the gap between QUIC and TLS is 665 ms in Munich and 861 ms in India. Overall download rate for TLS 1.2 is higher than QUIC and download rate for Q035 is higher than other QUIC versions for both India and Germany. But the difference between the download speeds is lesser in India than Germany. All observations are on IPv4 only. Q035 has higher download rates and lesser stall durations than other QUIC versions.

We find Google AS serves an overwhelming majority of the websites using QUIC, almost half of the websites in our target list. Only 5 ASes account for 85% of our target websites. We also find that performance over various ASes is not the same. EDGECAST AS performs better than GOOGLE AS on connection establishment times, at 50th percentile it performs about 12 ms better. For TTFB gap between GOOGLE and EDGECAST increases to 144 ms.

Conclusion III

EDGECAST doesn't support QUIC over IPv6 where GOOGLE emerges as the clear winner. Connection times for GOOGLE - Google LLC, OVH - OVH SAS, IHCRU - Internet-Hosting Ltd are 36.556, 98.33, 201.959 respectively at the 50th percentile.

GOOGLE AS having the largest amount of websites by quite a margin in our sample means that overall plots of our metrics mimic those of GOOGLE. But on closer inspection we can see that other ASes may perform better than GOOGLE. Looking at each individual AS more minutely we find that different QUIC versions perform similarly for GOOGLE, EDGECAST and OVH whereas for ICHPU and A2 HOSTING Q044 performs slightly better in terms of connection establishment times.

Running the `YouTube tests` separately to download a video from YouTube we find that both QUIC and TCP start with the same throughput but QUIC maintains it over the entire duration of the download, TCP on the other hand has many peaks and falls. Mean throughput of QUIC is over twice that of TCP(9.70 MB/s to 4.43 MB/s).

Using `quic_perf` and `tls_perf` to download files from Google Drive we find that QUIC performs better than TCP for smaller file sizes but for larger file sizes TCP has higher throughput.

Conclusion IV

The high throughput of QUIC comes at the cost of very high CPU utilization. CPU utilization for QUIC is 90% and remains consistent throughout the progress. TCP CPU utilization is about 50%. This is because QUIC spends a lot more time in making expensive `sendmsg/recvmsg` system calls. QUIC uses a maximum packet size of 1350 for IPv4 and 1370 for IPv6 to avoid packet fragmentation. TCP because of its bytestream abstraction and kernel support for segmentation can send/receive larger packets.

From the `strace` traces QUIC spends most of its time(76.89%) on the recvmsg system call.

UDP on the other hand has to make repeated calls to send/receive system calls and also the packets have to travel all the way up the stack. This can be seen from the flamegraph produced by `perf`. We can see that the width of the __sys_recvmsg and __sys_sendmsg is quite large indicating a lot of cpu usage in these paths

After running the tests concurrently to create multiple flows of QUIC and TCP it was found that QUIC grabs more than its fair share of bandwidth when it is competing with TCP. QUIC flows are fair to other QUIC flows and also TCP flows are fair to other TCP flows. Even with 2 and 3 competing TCP flows, QUIC maintains its mean bandwidth usage. Mean bandwidth usage of QUIC is more than twice the combined mean bandwidth usage of TCP. Though both TCP and QUIC use the same CUBIC

Conclusion V

congestion control algorithm, QUIC has been known to aggressively increase its congestion window size so it is able to grab most of the bandwidth more quickly.

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 [5] measurement infrastructure and SamKnows project[34]
- Mobile devices are a major source of QUIC traffic so tests need to be adopted for mobile devices using cronet[9].
- 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 I

[1] A. Aggarwal, S. Savage und T. E. Anderson. “Understanding the Performance of TCP Pacing”. In: *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, S. 1157–1165. DOI: 10.1109/INFCOM.2000.832483.

[2] S. Ahsan, V. Bajpai, J. Ott und J. Schönwälder. “Measuring YouTube from Dual-Stacked Hosts”. In: *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings*. Hrsg. von J. Mirkovic und Y. Liu. Bd. 8995. Lecture Notes in Computer Science. Springer, 2015, S. 249–261. DOI: 10.1007/978-3-319-15509-8_19.

[3] M. Belshe, R. Peon und M. Thomson. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC 7540. Mai 2015. DOI: 10.17487/RFC7540.

[4] W. de Bruijn. “Optimizing UDP for content delivery: GSO, pacing and zerocopy”.

[5] *CAIDA Ark*. URL: http://www.caida.org/projects/ark/.

[6] G. Carlucci, L. D. Cicco und S. Mascolo. “HTTP over UDP: an experimental investigation of QUIC”. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*. Hrsg. von R. L. Wainwright, J. M. Corchado, A. Bechini und J. Hong. ACM, 2015, S. 609–614. DOI: 10.1145/2695664.2695706.

[7] S. Cook, B. Mathieu, P. Truong und I. Hamchaoui. “QUIC: Better for what and for whom?” In: *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*. IEEE, 2017, S. 1–6. DOI: 10.1109/ICC.2017.7997281.

[8] R. Corbel, E. Stephan und N. Omnes. “HTTP/1.1 pipelining vs HTTP2 in-the-clear: Performance comparison”. In: *2016 13th International Conference on New Technologies for Distributed Systems (NOTERE)*. Juli 2016, S. 1–6. DOI: 10.1109/NOTERE.2016.7745823.

[9] *cronet*. URL: https://developer.android.com/guide/topics/connectivity/cronet/.

References II

[10] Y. Cui, T. Li, C. Liu, X. Wang und M. Kuhlewind. “Innovating transport with QUIC: Design approaches and research challenges”. In: *IEEE Internet Computing* 21.2 (2017), S. 72–76. ISSN: 10897801. DOI: 10.1109/MIC.2017.44.

[11] Y. Cui, T. Li, C. Liu, X. Wang und M. Kühlewind. “Innovating Transport with QUIC: Design Approaches and Research Challenges”. In: *IEEE Internet Computing* 21.2 (2017), S. 72–76. DOI: 10.1109/MIC.2017.44.

[12] K. Edeline, M. Kühlewind, B. Trammell, E. Aben und B. Donnet. “Using UDP for Internet Transport Evolution”. In: *CoRR* abs/1612.07816 (2016). arXiv: 1612.07816.

[13] Y. El-khatib, G. Tyson und M. Welzl. “Can SPDY really make the web faster?” In: *2014 IFIP Networking Conference, Trondheim, Norway, June 2-4, 2014*. IEEE, 2014, S. 1–9. DOI: 10.1109/IFIPNetworking.2014.6857089.

[14] M. Fischlin und F. Günther. “Multi-Stage Key Exchange and the Case of Google’s QUIC Protocol”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. Hrsg. von G. Ahn, M. Yung und N. Li. ACM, 2014, S. 1193–1204. DOI: 10.1145/2660267.2660308.

[15] A. Ford, C. Raiciu, M. J. Handley und O. Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses*. RFC 6824. Jan. 2013. DOI: 10.17487/RFC6824.

[16] *github repository. The QUIC Tracker*. URL: https://github.com/QUIC-Tracker/quic-tracker.

[17] *Google design document. QUIC Crypto*. URL: https://docs.google.com/document/d/1g5nIXAIkN_Y-7XJW5K45IblHd_L2f5LTaDUDwvZ5L6g/.

[18] *Google design document. QUIC from 10,000 feet*. URL: https://docs.google.com/document/d/1gY9-YNDNAB1eip-RTPbqphgySwSNSDHLq9D5Bty4FSU/.

[19] J. Iyengar und I. Swett. *QUIC Loss Detection and Congestion Control*. Internet-Draft draft-ietf-quic-recovery-18. Work in Progress. Internet Engineering Task Force, Jan. 2019. 36 S.

[20] J. Iyengar und M. Thomson. *QUIC: A UDP-Based Multiplexed and Secure Transport*. Internet-Draft draft-ietf-quic-transport-18. Work in Progress. Internet Engineering Task Force, Jan. 2019. 141 S.

[21] B. Jaeger. “Measuring Google QUIC Connection Establishment Times”. Bachelor’s Thesis. Technische Universität München.

References III

[22] A. M. Kakhki, S. Jero, D. R. Choffnes, C. Nita-Rotaru und A. Mislove. “Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols”. In: *Proceedings of the 2017 Internet Measurement Conference, IMC 2017, London, United Kingdom, November 1-3, 2017*. Hrsg. von S. Uhlig und O. Maennel. ACM, 2017, S. 290–303. DOI: 10.1145/3131365.3131368.

[23] P. K. Kharat, A. Rege, A. Goel und M. Kulkarni. “QUIC Protocol Performance in Wireless Networks”. In: *2018 International Conference on Communication and Signal Processing (ICCSP)*. Apr. 2018, S. 0472–0476. DOI: 10.1109/ICCSP.2018.8524247.

[24] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. R. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang und Z. Shi. “The QUIC Transport Protocol: Design and Internet-Scale Deployment”. In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*. ACM, 2017, S. 183–196. DOI: 10.1145/3098822.3098842.

[25] F. Li, J. W. Chung, X. Jiang und M. Claypool. “TCP CUBIC versus BBR on the Highway”. In: *Passive and Active Measurement - 19th International Conference, PAM 2018, Berlin, Germany, March 26-27, 2018, Proceedings*. Hrsg. von R. Beverly, G. Smaragdakis und A. Feldmann. Bd. 10771. Lecture Notes in Computer Science. Springer, 2018, S. 269–280. DOI: 10.1007/978-3-319-76481-8_20.

[26] P. Megyesi, Z. Kramer und S. Molnár. “How quick is QUIC?” In: *2016 IEEE International Conference on Communications, ICC 2016, Kuala Lumpur, Malaysia, May 22-27, 2016*. IEEE, 2016, S. 1–6. DOI: 10.1109/ICC.2016.7510788.

[27] K. Nepomuceno, I. N. d. Oliveira, R. R. Aschoff, D. Bezerra, M. S. Ito, W. Melo, D. Sadok und G. Szabó. “QUIC and TCP: A Performance Evaluation”. In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. Juni 2018, S. 00045–00051. DOI: 10.1109/ISCC.2018.8538687.

[28] S. Podanev. “Measuring YouTube Video Content Delivery over QUIC Protocol”. Master’s Thesis. Technische Universität München.

References IV

[29] F. Qian, V. K. Wang und R. Yafazolli. “Achieving Robust Mobile Web Content Delivery Performance Based on Multiple Coordinated QUIC Connections”. In: *IEEE Access* 6 (2018), S. 11313–11328. DOI: 10.1109/ACCESS.2018.2804222.

[30] *QUIC Research @ COMSYS*. URL: https://quic.comsys.rwth-aachen.de/data.php.

[31] S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain und B. Raghavan. “TCP fast open”. In: *Proceedings of the 2011 Conference on Emerging Networking Experiments and Technologies, Co-NEXT ’11, Tokyo, Japan, December 6-9, 2011*. Hrsg. von K. Cho und M. Crovella. ACM, 2011, S. 21. DOI: 10.1145/2079296.2079317.

[32] *RIPE DB*. URL: https://stat.ripe.net/.

[33] J. Rüth, I. Poese, C. Dietzel und O. Hohlfeld. “A First Look at QUIC in the Wild”. In: *Passive and Active Measurement - 19th International Conference, PAM 2018, Berlin, Germany, March 26-27, 2018, Proceedings*. Hrsg. von R. Beverly, G. Smaragdakis und A. Feldmann. Bd. 10771. Lecture Notes in Computer Science. Springer, 2018, S. 255–268. DOI: 10.1007/978-3-319-76481-8_19.

[34] *SamKnows project*. URL: https://samknows.com/.

[35] A. Saverimoutou, B. Mathieu und S. Vaton. “Which secure transport protocol for a reliable HTTP/2-based web service: TLS or QUIC?” In: *2017 IEEE Symposium on Computers and Communications, ISCC 2017, Heraklion, Greece, July 3-6, 2017*. IEEE Computer Society, 2017, S. 879–884. DOI: 10.1109/ISCC.2017.8024637.

[36] *SPDY Protoocl draft-3*. URL: https://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft3.

[37] P. D. Vaere, T. Bühler, M. Kühlewind und B. Trammell. “Three Bits Suffice: Explicit Support for Passive Measurement of Internet Latency in QUIC and TCP”. In: *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*. ACM, 2018, S. 22–28.

[38] P. Wang, C. Bianco, J. Riihijärvi und M. Petrova. “Implementation and Performance Evaluation of the QUIC Protocol in Linux Kernel”. In: *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM 2018, Montreal, QC, Canada, October 28 - November 02, 2018*. Hrsg. von A. Boukerche, S. S. Kanhere und P. Bellavista. ACM, 2018, S. 227–234. DOI: 10.1145/3242102.3242106.

References V

[39] J. Yoakum und L. Ong. *An Introduction to the Stream Control Transmission Protocol (SCTP)*. RFC 3286. Mai 2002. DOI: 10.17487/RFC3286.

[40] Y. Yu, M. Xu und Y. Yang. “When QUIC meets TCP: An experimental study”. In: *36th IEEE International Performance Computing and Communications Conference, IPCCC 2017, San Diego, CA, USA, December 10-12, 2017*. IEEE, 2017, S. 1–8. DOI: 10.1109/PCCC.2017.8280429.


Questions?