An in-depth technical overview of HTTP/3 and the QUIC protocol

IK1552
Internetworking

Emil Ståhl

An in-depth technical overview of HTTP/3 and the QUIC protocol

Emil Ståhl

April 2022

Abstract

The distribution of information on the Internet heavily depends on the HTTP protocol. The last decade has seen ever-increasing demands for performance, flexibility, and stability in data transfers. Due to the use of TCP as the underlying transport protocol, the current version of HTTP faces several limitations including overhead for encryption and connection establishments, poor migration flow, and insufficient congestion control leading to successfully delivered bytes being re-transmitted. The next version of HTTP, named HTTP/3, aims to solve these issues by using QUIC as its underlying protocol for delivering packets. This work aims to research how the next version of HTTP solves these issues. The obtained results show that HTTP/3 offers performance, stability, and flexibility improvements mainly for mobile users accessing the Internet from low latency networks. This has the potential of having positive implications for developing nations and society as a whole with contributions to improved social sustainability.

1 Introduction

This section provides an introduction to the work including its goal and purpose.

1.1 Background

Since the year of 1989, the distribution of information and multimedia on the Internet has relied on the HTTP protocol. HTTP presumes the use of a reliable transport protocol such as TCP. However, the modern Internet introduces new challenges that these protocols were not designed to handle efficiently[1, 2]. To improve the functionality of the Internet suite, the Internet community has introduced a new version of HTTP and an improved connection-oriented transportation layer protocol. Given the large scale of the Internet network, understanding the functionality and implications of these new protocols is important[3, 4, 5].

1.2 Problem

Analyzing the performance and functionality of the Internet protocol suite is key to tackling the scalability challenges associated with the large scale in which many Internet services operate. As of January 2021, there were 4.66 billion active Internet users worldwide or 59.5 percent of the global population. Of this total, 92.6 percent (4.32 billion) accessed the internet via mobile devices[3, 4]. Since HTTP together with its transport layer of choice are the underlying protocols of the Internet on which society relies it is important that these protocols are highly functioning and ensure maximum performance, flexibility, and stability[6, 7, 1, 2].

1.3 Purpose

The purpose of this work is to through a literature study explain and analyze how the HTTP/3 and QUIC protocols can improve the performance and functionality of the modern-day Internet. An aspect of sustainability exists when considering the development and usage of the Internet. All forms of digitization inevitably lead to a further dependency on electricity. However, one could also argue that a well-functioning Internet is a keystone for increasing social sustainability and achieving all of the sustainable development goals that have been established by the United Nations[8].

1.4 Goal

The goal of this work is to better understand the limitations of current Internet protocols and how HTTP/3 and QUIC may solve these issues.

1.5 Benefits and Sustainability

Many possible benefits may result from understanding the performance of Internet protocols. Since HTTP and transport layer protocols are part of the Internet's core infrastructure, which most of the modern society relies on, it is crucial to understand the underlying technology used to optimize for performance, up-time, and security[6, 9].

1.6 Methodology

The decision of methodology is important in a scientific study. In this work, the research method used is defined as a structured quantitative analysis[10].

2 Methodology

In this chapter, the different parts of the theoretical method are explained.

2.1 Literature study

The main methodology of this work is a literature study where we read through, analyze, and sort literature to identify the essential attribute of material to gain enough knowledge in the area to be able to discuss the implications of the area and draw relevant conclusions[10]. Specifically, the literature review methodology includes the following steps:

1. formulating the research question(s) and objective(s),
2. searching the extant literature,
3. screening for inclusion,
4. assessing the quality of primary studies, and
5. extracting data

Although these steps are presented here in sequential order, one must keep in mind that the review process can be iterative and that many activities can be initiated during the planning stage and later refined during subsequent phases[10]. The knowledge gained from this literature study is to be summarized in Section 3 and later discussed in Section 4.

3 Results

This section presents and analyzes the results found during the literature review.

3.1 Limitations of TCP

The current version of HTTP, version 2, makes use of TCP for delivering packets between interfaces on a network. Since TCP has been the most widely used connection-oriented transportation protocol during the last few decades, it has undergone a lot of research and analysis to improve its throughput and functionality[5]. The following sections present examples of such research[1].

3.1.1 Connection overhead

The connection overhead of TCP especially affects areas such as Internet-of-Things or other industries that establishes many short-lived connections to exchange a small amount of data[11]. Establishing a new TCP connection each time data is to be exchanged results in a lot of connection overhead. The connection overhead of establishment is especially bad for lossy networks such as wireless devices[12]. For data payloads that exceeds the default TCP MSS of 1480 bytes, the bandwidth overhead is around 2.8% which results in a efficiency of 97.33%. However, for applications sending smaller amounts of data, the overhead can far exceed the 4% mark[11, 1, 12]. Furthermore, when making use of technologies such as Transport Layer Security (TLS) the connection overhead

increases even more by adding 1-2 RTTs bringing the total number of RTTs for connection establishment to 3-4. The efficiency of the connection drops to 87.7% by using the AES protocol that is the preferred standard for TLS[12]. If every packet sent is not fully utilized the overall performance of the transmission of information is going to be very poor. When TCP is used as the underlying protocol to transmit HTTP traffic, 25% of the bandwidth can be consumed by headers and various large cookie values[5, 12, 6, 9, 13, 14, 15].

3.1.2 Migrating flows

In addition to connection overhead, migrating flows is also a limitation of current TCP implementations. Changing applications or devices may require a connection tear down and re-establishment of the TCP connection. This affects areas such as session information loss and NAT rebinding. For connection migrations, a routers NAT-table can be inconsistent leading to security issues[12, 9, 6, 16].

3.1.3 Half-Open Connections

Moreover, a half-open connection is something that can be improved in TCP, this means that a connection and ports are being used even if one end-point has crashed meaning that the connection is left in use until a timer expires[12, 11].

3.1.4 Other limitations

Another limitation of TCP is the need for multiple connections to the same host which HTTP in many cases make use of. However, establishing multiple connections results in overhead[11]. Currently, only SPDY supports multiplexing but is not widely supported[12, 17]. Lastly, Head-of-Line (HOL) blocking is a phenomena where resources are being locked up by some other entity in order to complete an action which leads to additional resources and latency are being used affecting performance[12, 18, 19, 14, 15]. This is further discussed in section 3.1.5.

3.1.5 How TCP affects HTTP connections

When HTTP/2 was introduced, it solved some problems that its predecessor HTTP/1.1 had, this included multiplexing HTTP exchanges in one TCP connection[2]. However, this does not solve the issue that all requests and responses are equally affected by packet loss since TCP is unaware of the HTTP abstraction and only sees a stream of bytes being sent, even if some bytes are successfully delivered this results in the whole packet being re-transmitted[20]. As briefly mentioned in section 3.1.1, TLS greatly increases the overhead of connections by adding 1-2 RTTs before the client can start to request data with HTTP affecting website load times[17, 14, 15]. Finally, a limitation of HTTP/2 is that it bounds to an IP address which leads to that clients are not able to change networks without re-establishing the connection[21]. As we now have learned,

the performance of HTTP/2 connections seems to be affected by limitations that originate in how TCP is designed and functions. Therefore, a successor to TCP for use together with HTTP is needed[17, 12, 7, 14, 15].

3.2 The QUIC Protocol

The QUIC Protocol stands for *Quick UDP Internet Connections* and was developed by Google starting in 2012[11][17]. The main objective behind QUIC is to overcome the limitations of TCP discussed in Section 3.1. To achieve this, despite functioning as a transport protocol QUIC is deployed in the application layer and is built on top of UDP meaning that when a receiving end encounters QUIC traffic it interprets it as UDP, this ensure that maximal compatibility is achieved[18, 11, 19]. Moreover, since QUIC is built on a unreliable protocol, it handles reliability features at the application layer[16, 1]. This section is going to present some features of QUIC and how it differs from TCP[13, 22, 23].

3.2.1 Connection establishment

As previously discussed in section 3.1.1, TCP results in a lot of overhead when establishing many different short-lived connections[24]. In QUIC, this problem is solved by reducing the number of RTTs for connection establishments to 1 RTT by having the client send a **CHLO** packet to the server it wants to connect to[25]. Consequently, the server responds with a **REJ** packet which stands for rejection, this packet contains information regarding certificates and tokens required to establish a connection[17, 1, 11]. When this information is obtained by the client, it enables the client to establish a new connection in 0 RTTs by using the cached credentials from the **REJ** packet to immediately start sending encrypted data to the server, meaning that there is no overhead for repeated connection establishments[7, 16, 11].

3.2.2 Congestion control

In Section 3.1.5, we described how TCP is unaware of the HTTP abstraction resulting in that successfully delivered bytes are being re-transmitted[12]. In QUIC, the congestion control algorithm is given more data to work with. An example of this is that each packet sent has a unique sequence number allowing QUIC to keep track of which packets that have been delivered, solving the TCP problem of re-transmitting successfully delivered bytes[9, 11].

3.2.3 Multiplexing Connections

When using TCP to send HTTP traffic, a client may open several connections resulting in overhead. QUIC reduces this overhead by combining many streams in one UDP pipeline meaning that, unlike TCP, QUIC only makes use of one UDP connection for transporting data between endpoints[11, 17, 12].

3.2.4 Improved migration

When switching devices or applications, TCP may need to re-establish the connection. This is not the case with QUIC since it makes use of a randomly generated 64-bit integer for identifying connections. This results in that QUIC can keep a connection open even if a parameter changes in the 5-tuple flow identifier[26, 9, 1].

3.2.5 Encryption

In order to encrypt TCP traffic, a secondary header is required adding overhead and RTTs. With QUIC, the encryption handshake is combined with the initial **REJ** message which ensures that the connection is always encrypted[11, 27]. This allows QUIC traffic to be encrypted without having to add secondary packet headers or RTTs, greatly reducing the overhead of encrypting traffic[17]. Additionally, QUIC even encrypts metadata preventing proxies or firewalls from interfering with the connection[16].

3.2.6 Head-of-Line Blocking

Since network stacks interpret QUIC traffic as UDP datagrams, there is no need to keep packets in a queue for reordering before delivering to the application[11, 17]. Moreover, if the application domain allows for arbitrary delivery order, QUIC pushes the packets directly to the application. This solves the problem of Head-of-Line blocking since the network stack pushes packets to the application layer as soon as it receives them[22, 9].

3.3 QUIC Packet

This section presents how the QUIC packet is formatted. However, there is a lot of variance regarding how a QUIC packet can be constructed including short and long packets, initial packets for **REJ** messages, and depending on whether a 0 or 1 RTT is needed. Because of this, this work is only going to cover a high-level overview of this topic[11, 1].

3.3.1 Header Format

QUIC packets come in two types, short and long. The majority of QUIC traffic is sent using the short version. However, for 1 RTT traffic, *i,e,* for negotiating keys and tokens, the longer packet version is used. The long packet contains fields including version, source and destination ID, and flags such as packet type and header form. The shorter version only contains destination connection ID, packet sequence number, and an encrypted payload with the data[1, 11].

3.3.2 Acknowledgements

QUIC uses two counts to keep track of acknowledgments (ACKs), *ACK Range Count* keeps track of which packet number data frames have been successfully

delivered while *First ACK Range* keeps track of which packets have been received as well as acknowledged[28, 11, 1]. This allows any QUIC stream to acknowledge any past or present data frame. If a frame is dropped, QUIC informs the sender how many packets were dropped through the gap in the ACK frame. Additionally, the NACK tells what packets are missing[1, 11].

3.4 HTTP/3

As discussed in Section 3.1.5, some of the problems with HTTP/2 is due to limitations of the TCP protocol. Therefore, IETF reported in late 2018 that the successor to HTTP/2 would be named HTTP/3 and use QUIC as its underlying transport protocol[21]. Prior to this decision, providers had already used HTTP/2 over QUIC for some time. In fact, this is a valid general description of HTTP/3 which basically runs the HTTP/2 over QUIC but with some optimizations. This section presents how HTTP/3 benefits from using QUIC as its underlying protocol and what potential problems HTTP/3 could pose[29, 14, 15].

3.4.1 HTTP/3 advantages

Thanks to QUIC, HTTP/3 does not suffer from the weak points of TCP. Since QUIC solves the issue of Head-of-Line blocking, HTTP/3 does not have to wait for successful transmission like its predecessor but instead continues the loading process[30]. HTTP/3 also greatly reduces the overhead of HTTPS encryption thanks to that the encryption handshake is built into the initial QUIC handshake, instead of dealing with this in the TLS layer. This reduces encryption overhead which lets clients load data faster[2]. Moreover, by using the connection ID of QUIC, clients can keep the same connection even when changing networks[31]. Just like its predecessor, HTTP/3 uses header compression, but one difference is that it uses QPack to resolve HTTP/2 HPACK compression, which is tied to a packet order[29, 32, 33]. HPACK allows for compression of the header and trailer sections, however, QUIC is not able to use this compression technique since it would once again lead to Head-of-Line blocking for field sections due to that it assumes a total order of frames across its multiple streams[33, 17]. QPACK solves this by reusing a majority of the HPACK techniques but is optimized to allow for correctness even in a case of out-of-order delivery[2, 31, 20, 21].

4 Discussion

This section discusses what potential problems HTTP/3 could pose along with what broader implications the protocol has for users and society as a whole.

4.1 Potential problems of HTTP/3

Even thus being widely used in areas other than QUIC, UDP is regarded skeptically as a network protocol[34]. Since HTTP/3 no longer implements security with TLS, data is no longer going to be as thoroughly verified due that UDP aims to deliver packets as fast as possible. Since QUIC is responsible for intermediate steps without a clear regulation on request-response, data and application security may be the main point of worry for entities such as ISPs. Providers may lose control of security when using HTTP/3 which may lead to that malware being spread in streams[34, 11].

4.2 Broader implications

Thanks to the QUIC protocol, HTTP/3 should bring several benefits for users and society. The implications seems to be the most profound for mobile users connecting through a high latency network where HTTP/3 offers to bring a more flexible, stable, and faster connection due to the changes discussed in previous sections[31]. Google reports that "QUIC outshines TCP under poor network conditions, shaving a full second off the Google Search page load time for the slowest 1% of connections."[35, 36, 36]. This gives users in the developing world with less reliable networks an easier access to the Internet which have big positive implications for several of the United Nations Sustainable Development Goals including areas covering quality education, industry, innovation, economic growth, and social sustainability in general[8]. Thanks to the more clever congestion control algorithm of QUIC, less bytes have to be re-transmitted saving both bandwidth and energy costs which moves society closer to achieving environmental sustainability[11, 8].

5 Conclusion

For many decades, HTTP has been the main protocol for distributing information on the Internet[5]. However, due to ever-increasing demands, the current version of HTTP faces performance issues due to the use of TCP as its underlying transport protocol. With the introduction of HTTP/3, the protocol moves to the QUIC protocol as its underlying protocol for delivery[2]. Thanks to improvements in QUIC, HTTP/3 sees several improvements regarding, among other things, no Head-of-Line blocking, faster connection establishments, and improved migration flows and congestion control leading to overall better performance[9]. Consequences of moving to HTTP/3 include benefits for users accessing the Internet from high latency low bandwidth networks on particular mobile devices. This has the potential to increase both social sustainability as well as ensure an even more reliable Internet protocol stack[8, 7].

Bibliography

[1] Peng Wang et al. "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 '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 227–234. ISBN: 978-1-4503-5960-3. DOI: 10.1145/3242102.3242106. URL: https://doi.org/10.1145/3242102.3242106 (visited on 04/21/2022).

[2] Gianluca Perna et al. "A first look at HTTP/3 adoption and performance". en. In: *Computer Communications* 187 (Apr. 2022), pp. 115–124. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2022.02.005. URL: https://www.sciencedirect.com/science/article/pii/S0140366422000421 (visited on 04/21/2022).

[3] Joseph Johnson. *Internet users in the world 2021*. en. KEPIOS. Jan. 2022. URL: https://www.statista.com/statistics/617136/digital-population-worldwide/ (visited on 04/05/2022).

[4] *Usage Statistics of HTTP/2 for Websites, April 2022*. en. Apr. 2022. URL: https://w3techs.com/technologies/details/ce-http2 (visited on 04/21/2022).

[5] Santosh Kumar and Sonam Rai. *Survey on Transport Layer Protocols: TCP & UDP*. en. Graphic Era University, Dehradun (India). May 2012. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.7346&rep=rep1&type=pdf (visited on 04/21/2022).

[6] Ali H. Wheeb. "Performance Comparison of Transport Layer Protocols". en. In: *International Journal of Advanced Research in Computer Science and Software Engineering* Volume 5, Issue 12, December 2015 (Dec. 2015). ISSN: 22776451, 2277128X. URL: https://www.researchgate.net/publication/291775032_Performance_Comparison_of_Transport_Layer_Protocols.

[7] Martino Trevisan et al. "Measuring HTTP/3: Adoption and Performance". In: *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet)*. June 2021, pp. 1–8. DOI: 10.1109/MedComNet52149.2021.9501274.

[8] Antonio Guterres. *Goal 7 — Department of Economic and Social Affairs*. en. Jan. 2021. URL: https://sdgs.un.org/goals/goal7 (visited on 04/21/2022).

[9] Shintaro Kawai, Kouto Miyazawa, and Saneyasu Yamaguchi. "Performance Evaluation of HTTP/3 QUIC on a Network with High Latency and High Packet Loss Ratio". ja. In: *IEICE Proceedings Series* 63.N2-5 (Dec. 2020). Publisher: The Institute of Electronics, Information and Communication Engineers. ISSN: 2188-5079. URL: https://www.ieice.org/publications/proceedings/summary.php?iconf=ICETC&session_num=N2&number=N2-5&year=2020 (visited on 04/21/2022).

[10] Deborah Finfgeld-Connett and Diane Johnson. "Literature search strategies for conducting knowledge-building and theory-generating qualitative systematic reviews". In: *Journal of advanced nursing* 69 (May 2012). DOI: 10.1111/j.1365-2648.2012.06037.x.

[11] Adam Langley. *The QUIC Transport Protocol — Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. Aug. 2017. URL: https://dl.acm.org/doi/abs/10.1145/3098822.3098842 (visited on 04/21/2022).

[12] Wei-Qiang Xu and Tie-Jun Wu. "TCP Issues in Mobile Ad Hoc Networks: Challenges and Solutions". en. In: *Journal of Computer Science and Technology* 21.1 (Jan. 2006), pp. 72–81. ISSN: 1860-4749. DOI: 10.1007/s11390-006-0072-2. URL: https://doi.org/10.1007/s11390-006-0072-2 (visited on 04/21/2022).

[13] Gaetano Carlucci, Luca de Cicco, and S Mascolo. *HTTP over UDP: an experimental investigation of QUIC — Proceedings of the 30th Annual ACM Symposium on Applied Computing*. en. Apr. 2015. URL: https://doi.org/10.1145/2695664.2695706 (visited on 04/21/2022).

[14] Alessandro Ghedini and Rustam Lalkak. *HTTP/3: the past, the present, and the future*. en. Sept. 2019. URL: https://blog.cloudflare.com/http3-the-past-present-and-future/ (visited on 04/21/2022).

[15] Achim Weiss. *HTTP/3: the next Hypertext Transfer Protocol explained simply*. en. Aug. 2020. URL: https://www.ionos.com/digitalguide/hosting/technical-matters/http3-explained/ (visited on 04/21/2022).

[16] Mukesh Soni and Brajendra Singh Rajput. "Security and Performance Evaluations of QUIC Protocol". en. In: *Data Science and Intelligent Applications*. Ed. by Ketan Kotecha et al. Lecture Notes on Data Engineering and Communications Technologies. Singapore: Springer, 2021, pp. 457–462. ISBN: 9789811544743. DOI: 10.1007/978-981-15-4474-3_51.

[17] Puneet Kumar. "QUIC Quick UDP Internet Connections". en. In: *Santa Clara University, Computer Networks: COEN-233Department of Computer Science* (Oct. 2020), p. 21. URL: https://www.researchgate.net/publication/344490561_QUIC_Quick_UDP_Internet_Connections.

[18] Shawn Maust. *Some QUIC Benefits of HTTP/3 — A Faster Web*. en. Apr. 2019. URL: https://www.afasterweb.com/2019/04/30/some-quic-benefits-of-http/3/ (visited on 04/21/2022).

[19] Alyssa Wilk, Ryan Hamilton, and Ian Swett. *A QUIC update on Google's experimental transport*. en. Apr. 2015. URL: https://blog.chromium.org/2015/04/a-quic-update-on-googles-experimental.html (visited on 04/21/2022).

[20] Marjan Milošević, Vladimir Mladenovic, and Uroš Pešović. "Evaluation of HTTP/3 Protocol for Internet of Things and Fog Computing Scenarios". In: (Jan. 2021). Accepted: 2022-02-02T17:50:14Z. ISSN: 12201766. DOI: 10.24846/v30i3y202107. URL: https://scidar.kg.ac.rs/handle/123456789/14024 (visited on 04/21/2022).

[21] Jasenka Dizdarević and Admela Jukan. "Experimental Benchmarking of HTTP/QUIC Protocol in IoT Cloud/Edge Continuum". In: *ICC 2021 - IEEE International Conference on Communications*. ISSN: 1938-1883. June 2021, pp. 1–6. DOI: 10.1109/ICC42927.2021.9500675.

[22] Amit Srivastava. "Performance Analysis of QUIC Protocol under Network Congestion". en. In: *Worcester Polytechnic Institute* (May 2017), p. 64. URL: https://web.cs.wpi.edu/~claypool/ms/quic/quic-thesis.pdf (visited on 04/21/2022).

[23] Yong Cui, Tianxiang Li, and Cong Liu. *Innovating Transport with QUIC: Design Approaches and Research Challenges — IEEE Journals & Magazine — IEEE Xplore*. en. Tech. rep. IEEE Internet Computing, Mar. 2017, pp. 72–76. URL: https://ieeexplore.ieee.org/abstract/document/7867726?casa_token=o4IA1hCtcSEAAAAA:_SSQk_WL0dq_qbDNudto7EB_8txIRs6E9E0UGCwZEB715P1zIBz_-JO4T3J_VrfuqVn7JyQ (visited on 04/21/2022).

[24] Quentin De Coninck and Olivier Bonaventure. "Multipath QUIC — Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies". In: *CoNEXT '17: Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies* (Nov. 2017), pp. 160–166. DOI: 3143361.3143370. URL: https://doi.org/10.1145/3143361.3143370 (visited on 04/21/2022).

[25] Prashant Khrat and Muralidhar Kulkarni. "Modified QUIC protocol with congestion control for improved network performance - Kharat - 2021 - IET Communications - Wiley Online Library". en. In: 15.9 (Mar. 2021). DOI: https://doi.org/10.1049/cmu2.12154. URL: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/cmu2.12154 (visited on 04/21/2022).

[26] Marc Fischlin and Felix Günther. "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol — Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security". en. In: *ACM SIGSAC Conference on Computer and Communications Security* (Nov. 2014). DOI: https://doi.org/10.1145/2660267.2660308. URL: https://dl.acm.org/doi/abs/10.1145/2660267.2660308?casa_token=X5Yjc3y9ZX0AAAAA:Zt_umH-Hgx2jjOXi_HoP2uiOzVn8F8CS9EMcyx4ppzN_gOD1zm-J4aKGMtXW82gkiBmj_9nH60Q (visited on 04/21/2022).

[27] Arash Molavi Kakhki et al. "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 '17. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 290–303.

ISBN: 978-1-4503-5118-8. DOI: 10.1145/3131365.3131368. URL: https://doi.org/10.1145/3131365.3131368 (visited on 04/21/2022).

[28] Adam Langley et al. "The QUIC Transport Protocol: Design and Internet-Scale Deployment". In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 183–196. ISBN: 978-1-4503-4653-5. DOI: 10.1145/3098822.3098842. URL: https://doi.org/10.1145/3098822.3098842 (visited on 04/21/2022).

[29] Máté Tömösközi, Martin Reisslein, and Frank H. P. Fitzek. "Packet Header Compression: A Principle-Based Survey of Standards and Recent Research Studies". In: *IEEE Communications Surveys Tutorials* 24.1 (2022). Conference Name: IEEE Communications Surveys Tutorials, pp. 698–740. ISSN: 1553-877X. DOI: 10.1109/COMST.2022.3144473.

[30] Robin Marx et al. "Resource Multiplexing and Prioritization in HTTP/2 over TCP Versus HTTP/3 over QUIC". en. In: *Web Information Systems and Technologies*. Ed. by Alessandro Bozzon, Francisco José Domínguez Mayo, and Joaquim Filipe. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2020, pp. 96–126. ISBN: 978-3-030-61750-9. DOI: 10.1007/978-3-030-61750-9_5.

[31] Darius Saif, Chung-Horng Lung, and Ashraf Matrawy. "An Early Benchmark of Quality of Experience Between HTTP/2 and HTTP/3 using Lighthouse". In: *ICC 2021 - IEEE International Conference on Communications*. ISSN: 1938-1883. June 2021, pp. 1–6. DOI: 10.1109/ICC42927.2021.9500258.

[32] R. Peon and H. Ruellan. *RFC 7541: HPACK: Header Compression for HTTP/2*. en. May 2015. URL: https://www.hjp.at/doc/rfc/rfc7541.html (visited on 04/21/2022).

[33] R. Peon and H. Ruellan. *HPACK: Header Compression for HTTP/2*. en. Tech. rep. RFC7541. RFC Editor, May 2015, RFC7541. DOI: 10.17487/RFC7541. URL: https://www.rfc-editor.org/info/rfc7541 (visited on 04/21/2022).

[34] David Choffnes. *Problems that UDP and only UDP has*. en. Aug. 2018. URL: https://www.cryptologie.net/article/449/problems-that-udp-and-only-udp-has/ (visited on 04/21/2022).

[35] Ian Swett and Michael Behr. *Introducing QUIC support for HTTPS load balancing*. en. June 2018. URL: https://cloud.google.com/blog/products/gcp/introducing-quic-support-https-load-balancing/ (visited on 04/21/2022).

[36] Mattias Geniar. *Google's QUIC protocol: moving the web from TCP to UDP*. en. Section: blog. July 2016. URL: https://ma.ttias.be/googles-quic-protocol-moving-web-tcp-udp/ (visited on 04/21/2022).