# PERFORMANCE COMPARISON OF HTTP/3 SERVER IMPLEMENTATIONS

Jiří Balej[1], Tomáš Sochor[1]

[1]Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic

## ABSTRACT

The paper opens the question of comparison published open-source webservers with support of HTTP/3. This is brand new protocol standardised in 2022 was developed with aim to speed up web communication and instead of TCP uses UDP with new QUIC protocol. There is summarized current state of HTTP protocols development and studies, where is compared performance of various HTTP versions. The aim of paper is to compare different open-source webserver implementations with HTTP/3 support in laboratory. Five different scenarios were presented to test ordinary real-life situations. The results of all three servers with different content and HTTP/1.1 or HTTP/3 protocols are presented. Main result would be better performance of Caddy and nginx server in bad connection conditions, but without speed limit, large delay or loss the OpenLiteSpeed was fastest.

**Keywords:** HTTP/3, QUIC, Caddy, Nginx, OpenLiteSpeed, webserver

**JEL Code:** L86, C88, O31

## 1 INTRODUCTION

Thirty-five years have passed since Tim Barnes-Lee at CERN proposed the first version of web communication. Since then, the HTTP (HyperText Transfer Protocol) has become the most used application protocol in the Internet. The first standardization by IETF (Internet Engineering Task Force) happened in 1996 in RFC 1945. Since this version many improvements like keep--alive connections, content negotiation and virtual hosts (in 1997 with HTTP/1.1, RFC 2068), binary coding, multiplexing and header compression (in 2015 with HTTP/2, RFC 7540) and using QUIC instead TCP protocol (in 2022 with HTTP/3, RFC 9114) were included to make HTTP complex application. The issue of security has been addressed by adding SSL/TLS layer under the basic HTTP communication (Lysenko, 2023), which provides authentication of server (and optionally a client), confidentiality and integrity of sent data.

HTTP communication also affect Cookies, which are capable of sharing session-related information between independent HTTP requests.

The main research question here is to check the difference among webservers implementations for HTTP/3 and to determine the dependence of performance on the network condition.

https://doi.org/10.11118/978-80-7509-990-7-0017

The key attribute for user's Quality of Service/Experience is the load time of the webpage, which will be tested as the "performance" parameter. Currently several webserver implementations support HTTP/3 and our experiment should give answer if those implementations perform similarly under the same conditions. As a reference, the well-known HTTP/1.1 protocol was tested to find out whether the performance of webservers with HTTP/1.1 and HTTP/3 protocols are similar.

After the introduction and stating the aim of article, the next part contain an actual state of HTTP protocol and other comparison studies of HTTP protocol versions are presented. Third section is devoted to experiment settings, topology and network conditions used in scenarios. Finally, the results are presented in graphs ontaining the measurement results of all prepared combinations. The findings are concluded and discussed in the last section.

## 2 ADVANCES IN HTTP PROTOCOL

One of the stable and even nowadays common version of HTTP protocol is HTTP/1.1 that has provided all necessary web functions for normal web operation and ruled the web for over 20 years. This changed around 2010, when Google started development of advanced HTTP protocol, which would speed up the web communication. The resulting SPDY should have been able to speed up loading time of webpage by 50% (Belshe, 2012). One of the key ideas was to use header compression and binary encoding. The multiplexing of packets for various content, its prioritization and server push have improved the speed greatly. All the functions were later included in the official HTTP/2 standard.

Right after that, work on development of HTTP/3 started, again driven by Google. All HTTP protocol were using TCP (Transmission Control Protocol) as transport-layer (L4) protocol. The TCP provide connection-oriented and reliable data transfer. TCP also uses sliding window parameter to dynamically utilize the maximum available bandwidth (congestion control). Biswal (2022) compares some of current TCP congestion control algorithms and states that BIC is default algorithm on Linux.

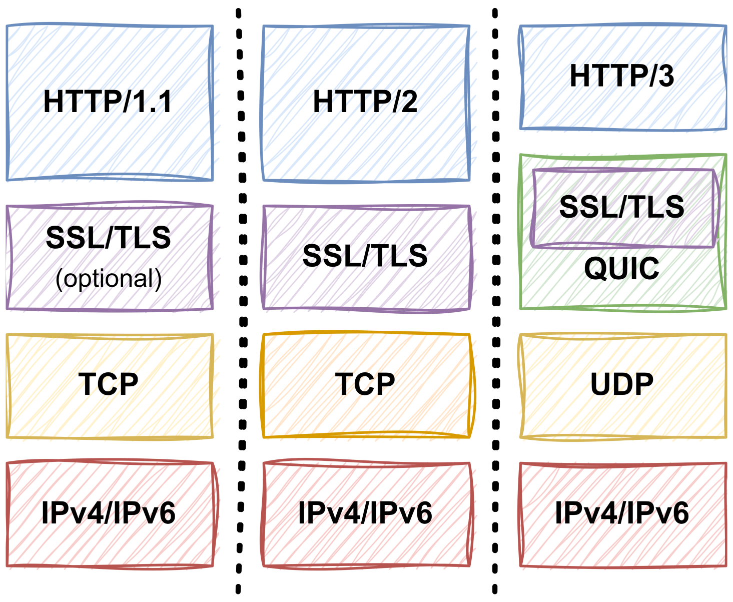


**Fig. 1:** Architecture of TCP/IP stack with HTTP
Source: Saadat, 2022

All the TCP congestion control algorithms rapidly slow down with loss of any packet and use numbered line of packets, which makes multiplexing of different streams complicated. That's why the new version of HTTP started to use UDP (User Datagram Protocol) as transport layer protocol, which offers connection-less best-effort send without any control of transmission. The architecture of TCP/IP stack with different versions of HTTP is on Figure 1, where is added the security layer (SSL/TLS) function.

## 2.1 QUIC

The QUIC (Quick UDP Internet Connections) protocol should outperform the drawbacks of TCP connections, with at the same time providing reliable data transfer necessary for HTTP. Moreover, QUIC contain the security layer (SSL/TLS) which is implied during initialization of communication and then protect whole communication.

Iyengar (2021) specifies in standard (RFC 9000) for QUIC that each communication starts with QUIC handshake, where are exchanged the supported cryptography algorithms and server is always authenticated by asymmetric cryptography (public key is in server certificate), optionally can be also the client authenticated. In case it is not the communication with server, QUIC can use so-called 0-RTT and because the server is known the client with the first packet send also encrypted data.

Other mechanisms used in QUIC specified by Lyengar (2021) are Connection Migration, Flow Control, Stream Multiplexing and Reliability. Packet numbering with positive acknowledgment and retransmission are common for reliability. QUIC moreover implies selective acknowledgment, fast retransmit and tail loss probes to achieve greater performance in case of data loss.

## 2.2 HTTP/3

The biggest difference of HTTP/2 is the usage of QUIC (and UDP) as transport layer, which could be a challenge for socket association, firewall permissions and packet inspection technologies. Bishop (2022) in specification of HTTP/3 (RFC 9114) also describes a new HTTP header "Alt-Svc" announcing its support and could ever redirect on different port number. Clients also open only one connection with the server, which is kept open until all requests are served. Loading of webpage could be also speeded up using server-push mode, where server sends required data without client's request.

Even the QUIC was standardized in 2022 the big companies like Cloudflare, Google, Microsoft, Meta and others uses it for several years. In January 2024 the average usage statistics (W3Techs, 2024) stated, that 28.4% of all websites use HTTP/3 and the number is increasing.

## 2.3 Studies and measurements over HTTP/3

Since the proposing idea of HTTP traffic over UDP till today, when HTTP/3 is standardized, there have been many survey comparing properties of HTTP/3 with its predecessors. One of the first papers was published by Carlucci (2015) and the compares TCP CUBIC with Google's public implementation of QUIC (v.21). Then Mefyesi (2016) published a study where the HTTP, SPDY and QUIC were compared for time needed to load a webpage. The question if QUIC would be better than others and for what cases is investigated by Cook (2017), all mentioned papers agree that older variants are better, at least in most common network conditions.

The state how QUIC is implemented was summarized by Rüth (2018), where 161 thousand of domain names support QUIC and about 3–9% of overall web traffic is QUIC. The better performance for video content and smaller files was HTTP/3 in the paper from Shreedhar (2022). Another performance measurement was done by Perna (2022), with conclusion of HTTP/3 advantages for large delays. Dubec (2023) compares not the browsers but client implementation.

Most of studies used public web services like Google to make comparison of performance, functions and adoption. Only few papers used any of open-source webserver with HTTP/3 support and tested the site in laboratory under various networks conditions.

## 3 METHODOLOGY

The aim of the paper is to test performance of open-source webservers with HTTP/3 support. Currently (beginning of 2024) is fully or experimentally supported by ASP.NET Core Kestrel, Caddy, Microsoft IIS, Nginx, OpenLiteSpeed and LiteSpeed. In web browsers HTTP/3 is supported more commonly and all major browsers currently support it.

From the list above, three open-source webservers – Caddy (version 2.7.6), Nginx (1.25.3) and OpenLiteSpeed (1.7.18-2) were chosen. As a client the line-based web browser "curl" (version 8.2.1) built with HTTP3 support was used and Google Chrome was used too to compare some results for which the line-based client output was not clear. For all devices (servers, client and traffic control) Ubuntu Server 22.04.3 LTS installed as VM in VirtualBox (7.0.10) was used. As the topology for the experiment shows in Figure 2, all devices are run on independent hardware (Intel Core i5-3470, 16 GiB RAM, SSD hard drive, 1 Gbit/s network card), all virtual machines have assigned 4 CPU cores and 8 GiB RAM. The only switch is a dedicated hardware (MikroTik RB4011GS+RM) with integrated 5 port 1 Gbit/s switching module.

On each webserver, there were three types of content website and two files (1 MiB and 10 MiB). The website was copy of homepage for Mendel University in Brno, which contain 30 files (HTML, CSS, JS, JPG,..) and its overall size is about 5 MiB. Based on general characteristics of Internet traffic (Araújo, 2019) four scenarios were set as shown in Table 1 with the worsened network conditions (speed, delay and loss), which were intentionally affected by Linux Traffic Control (tc). To ensure correctness of tc settings, we also tested the communication without any downgrade (speed up to 300 Mbps, latency 2 ms, loss 0%). Each scenario was tested 10 times against all 3 servers with protocols HTTP/1.1 and HTTP/3.

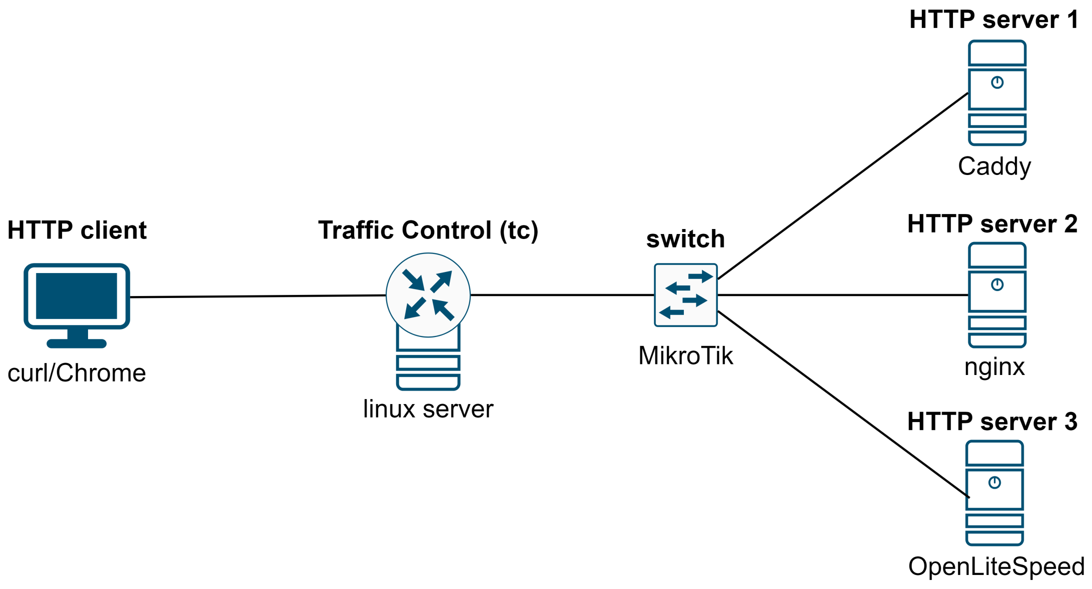


**Fig. 2:** Laboratory experiment topology

| | Speed | Delay | Loss |
|---|---|---|---|
| High speed LAN connection (cable) | 100 Mbps | 5 ms | 0.001% |
| Home WiFi | 60 Mbps | 25 ms | 0.01% |
| WiFi in public place (like Café) | 25 Mbps | 70 ms | 0.05% |
| Unstable connection in moving vehicle | 10 Mbps | 250 ms | 0.1% |

**Tab. 1** Parameter of scenarios for HTTP/1.1 and HTTP/3 measurement

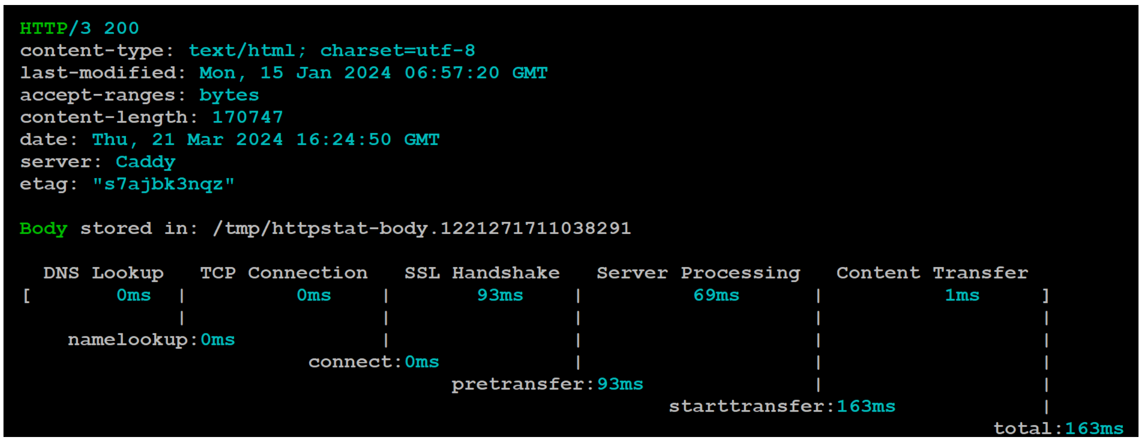
## 4 RESULTS

Collected measured values contained time needed to load whole file/webpage, all header values including status code of communication and the content. The example of result for one page load is shown in Figure 3, where are partial times for the process. In cast we got different status code, than 200 OK, the measurement was repeated. For each scenario there were 10 times of loading complete content and the median, average, minimum and maximum were used for comparisons.

For final comparison, we used the median values to avoid exceptional and random delays. The loading times in bar graphs for all scenarios are shown in Figure 4, mainly to compare performances between individual webservers. Each bar represents a time needed to load the whole content in milliseconds (ms), the lower number (lower bar) means better performance. The left side of graphs represents HTTP/1.1 protocol while the right one HTTP/3 protocol. Each of individual scenarios described in Table 1 are represented by a single graph as shown by the graph heading.

## 5 DISCUSSION AND CONCLUSIONS

The results prove that all three open-source webserver implementations are capable of handling HTTP/3 content. Of course, there are differences in implementations, for example webserver OpenLiteSpeed has best performance in scenario without any loss, delay or



```
HTTP/3 200
content-type: text/html; charset=utf-8
last-modified: Mon, 15 Jan 2024 06:57:20 GMT
accept-ranges: bytes
content-length: 170747
date: Thu, 21 Mar 2024 16:24:50 GMT
server: Caddy
etag: "s7ajbk3nqz"

Body stored in: /tmp/httpstat-body.1221271711038291

  DNS Lookup   TCP Connection   SSL Handshake   Server Processing   Content Transfer
[       0ms  |          0ms   |        93ms   |          69ms     |          1ms     ]
             |                |               |                   |                  |
    namelookup:0ms            |               |                   |                  |
                        connect:0ms           |                   |                  |
                                    pretransfer:93ms              |                  |
                                                      starttransfer:163ms            |
                                                                                 total:163ms
```

**Fig. 3:** Detail print of loading time result, where individual delay components are displayed.

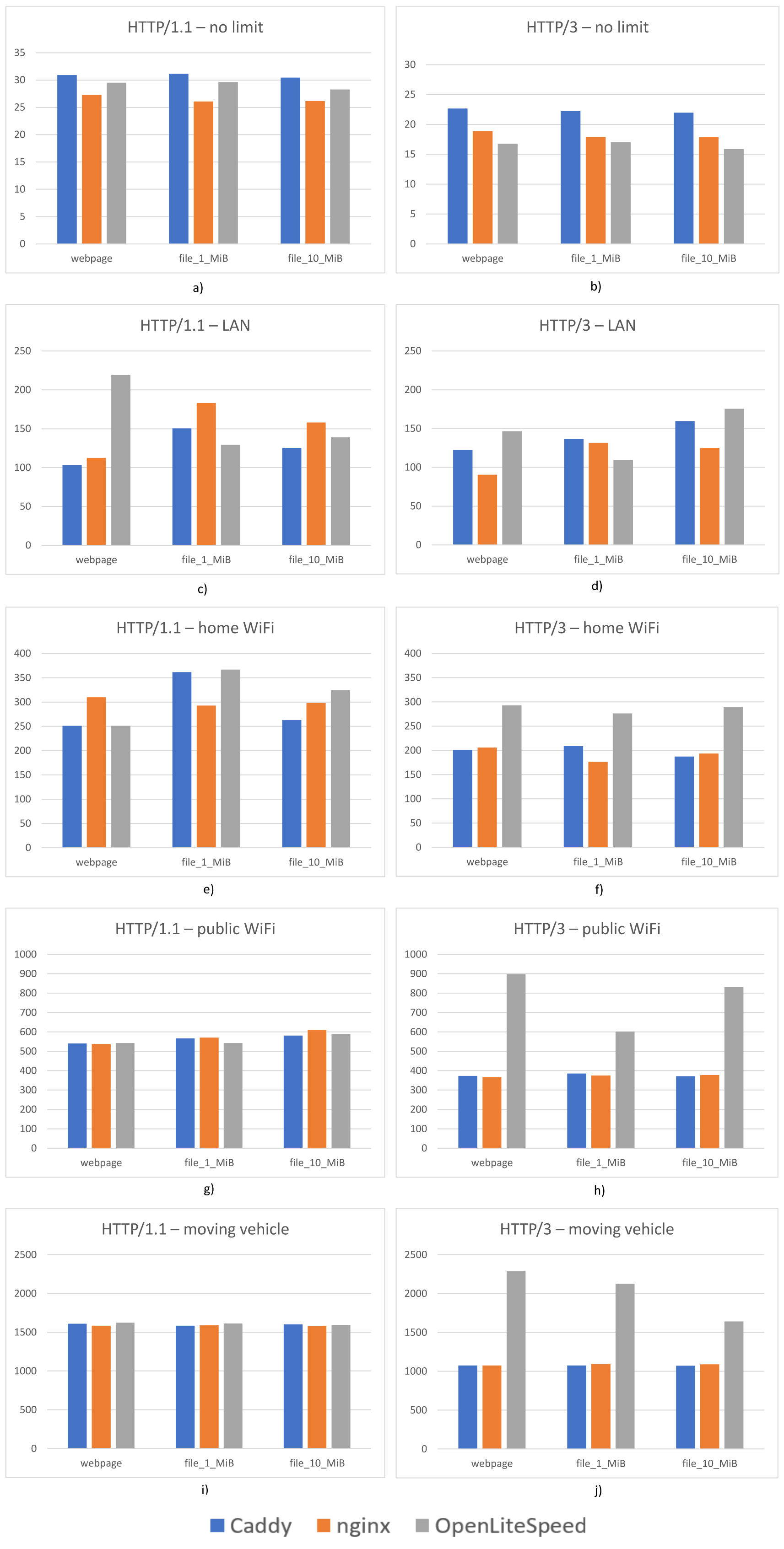


**Fig. 4:** Graph of median loading time for each scenario and type of content, webserver and HTTP protocol.

constrain of speed, on the other hand in most other scenarios is the one of worst. Webservers Caddy and nginx show similar performance in bad situations.

When one tries to compare HTTP/1.1. and HTTP/3 performance, there is no clear winner but it seems that our result confirms previously published results that HTTP/3 is better in good network conditions (as graphs on the left show) while the HTTP/3 dominance is not clear in worse transmission conditions.

**Acknowledgements**
This paper was supported by the project CZ.02.1.01/0.0/0.0/16_017/0002334 Research Infrastructure for Young Scientists, this is co-financed from Operational Programme Research, Development and Education.

## REFERENCES

ARAÚJO, J. 2019. Real-world latency and packet loss. *Codeavel blog* [online]. Available at: https://blog.codavel.com/performance-report-defining-use-cases [Accessed: 2023-12-17].

BELSHE, M., PEON, R., THOMSON, M. and MELNIKOV, A. 2012. SPDY Protocol draft-ietf-httpbis-http2-00. *Internet-Draft* [online]. Available at: https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-http2-00

BISWAL, S. P., GUPTA, S., KUMAR, A. and PATEL, S. 2022. *Comparative Analysis of Compound TCP with Various End-to-End Congestion Control Mechanisms* [online]. Available at: http://dspace.nitrkl.ac.in/dspace/bitstream/2080/3879/1/PatelS_ICAC3N-22.pdf

CARLUCCI, G., DE CICCO, L. and MASCOLO, S. 2015. HTTP over UDP: an experimental investigation of QUIC. *Internet-Draft In Proceedings of the 30th Annual ACM Symposium on Applied Computing* (SAC '15). Association for Computing Machinery, New York, NY, USA, 609–614. https://doi.org/10.1145/2695664.2695706

COOK, S., MATHIEU, B., TRUONG, P. and HAMCHAUOI, I. 2017. QUIC: Better for what and for whom? *IEEE International Conference on Communications* (ICC). Paris, France, pp. 1-6. https://doi.org/10.1109/ICC.2017.7997281

DUBEC, J., BALAŽIA, J. and ČIČÁK, P. 2023. Performance evaluation of the HTTP/3 client implementations. *46th International Conference on Telecommunications and Signal Processing* (TSP), Prague, Czech Republic, pp. 260-263- doi: 10.1109/TSP59544.2023.10197834

KUTTER, M. and JAEGER, B. 2022. Comparison of Different QUIC Implementations, *Network Architectures and Services*- doi: 10.2313/NET-2022-07-1_10

LYSENKO, S. and SAVENKO, B. 2023. Distributed Discrete Malware Detection Systems Based on Partial Centralization and Self-Organization. *International Journal of Computing*, 22, 117-139.

MEGYESI, P., KRÄMER., Z. and MOLNÁR, S. 2016. How quick is QUIC?. *IEEE International Conference on Communications* (ICC), Kuala Lumpur, Malaysia, pp. 1-6, doi: 10.1109/ICC.2016.7510788

PERNA, G., TREVISAN, M., GIORDANO, D. and DRAGO, I. 2022. A first look at HTTP/3 adoption and performance, *Computer Communications*, Volume 187, pp. 115-124, ISSN 0140-3664. https://doi.org/10.1016/j.comcom.2022.02.005

RÜTH, J., POESE, I., DIETZEL, C. and HOHLFELD, O. 2018. A First Look at QUIC in the Wild. *Passive and Active Measurement* (PAM 2018). Springer, doi: 10.48550/arXiv.1801.05168

SHREEDHAR, R., PANDA, R., PODANEV, S. and BAJPAI, V. 2022. Evaluating QUIC Performance Over Web, Cloud Storage, and Video Workloads. In: *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1366-1381, June 2022. https://doi.org/10.1109/TNSM.2021.3134562.

SAADAT, R. 2022. Spirent Communications. In: *Building Next-Generation Web with HTTP/3*. https://www.spirent.com/blogs/building-next-generation-web-with-http-3 [Accessed: 2024-01-10].

**Contact information**
Jiří Balej: e-mail: jiri.balej@mendelu.cz
Tomáš Sochor: e-mail: tomas.sochor@mendelu.cz