Evaluation of HTTP/3 Protocol for Internet of Things and Fog Computing Scenarios

Marjan MILOŠEVIĆ, Vladimir MLADENOVIĆ, Uroš PEŠOVIĆ*
University of Kragujevac, Faculty of Technical Sciences Čačak, Svetog Save 65, Čačak, 32102, Serbia
marjan.milosevic@ftn.kg.ac.rs, vladimir.mladenovic@ftn.kg.ac.rs,
uros.pesovic@ftn.kg.ac.rs (**Corresponding author*)

Abstract: The paper investigates the performance properties of the new HTTP/3 protocol in IoT scenarios, with the focus on fog computing. A specific experimental environment is created, providing realistic IoT conditions. End-to-end delay and three different models for packet loss are introduced using the NetEm simulator in Linux. Three contemporary HTTP protocols are contrasted with two scenarios of IoT use. By comparing the experimental results obtained by the protocols HTTP/1, HTTP/2 and HTTP/3, it can be noticed that the protocol HTTP/3 outperforms the protocol HTTP/2 in 19 cases out of 24, yet it only outperforms HTTP/1 in 12 cases out of 24. Even if HTTP/3 is under development, it fulfils the requirements for performance and security for IoT and fog computing scenarios, involving unconstrained devices. With a lower connection overhead and an inherent security, HTTP/3 has secured a firm place for itself among other specialised IoT protocols.

Keywords: Internet of Things, Fog computing, HTTP/3, Web protocols, NetEm, Raspberry Pi.

1. Introduction

IoT (Internet of Things) refers to the networked interconnection of everyday objects, which are often equipped with ubiquitous intelligence (Xia et al., 2012). IoT represents a concept of physical computational objects (such as sensors, home devices etc.) made accessible to humans via Internet. IoT devices are employed in a broad range of applications where information about device operation, performance and environmental conditions in device vicinity need to be monitored and controlled remotely. IoT applications have spread to almost every part of human activity, including consumer, commercial, industrial and infrastructure applications. Smart home IoT devices have a rising presence in consumer households: learning thermostats, energy tracking switches, video doorbells, smart baby observers, and remotely controlled washing machines, all increasingly available and affordable. These devices, using their sensors and actuators transform the regular house into a digital home, providing their user with data and control via the Internet. Commercial applications include healthcare, transportation, and building automation. Industrial applications of IoT include manufacturing, agriculture, and the military. Infrastructure applications include energy metering, environmental monitoring etc.

Under the umbrella of the IoT paradigm, several other models have emerged. In certain scenarios, it is of great convenience to use the edge device not only to collect and measure different parameters (such as temperature or humidity) or act accordingly but to store and process certain amounts of data. This model is called "fog computing". As Fog computing is implemented at the edge of the network, it provides low latency, location awareness, and improves quality-of-services (QoS) for streaming and real-time applications (Stojmenovic & Wen, 2014, Chiang & Zhang, 2016).

IoT devices typically use wireless transmission, utilising a broad range of communication standards such as short-range IEEE 802.15.4 or IEEE 802.11 networks or long-range networks such as GSM, LTE, 5G and others. In the case of the short-range communication standards, IoT devices are mutually interconnected to form a local network, further connected to the Web server through network gateway, which also acts as a network controller or network sink node (Figure 1). In the case of long-range networks, IoT devices are capable to directly communicate with the Web server. The concept of fog computing uses fog nodes for data processing to significantly decrease the load on cloud infrastructure, while additionally reducing the data latency.

Despite very diverse communication technologies on lower layers of protocol stack, IoT devices tend to use common protocols on higher layers to enable compatibility with ever-evolving Internet technologies. Most IoT devices rely on TCP/IP protocol stack to be able to interact via Internet infrastructure. Although there are special protocols developed for inter-machine communication, such as MQTT - Message Queuing Telemetry Transport (Banks et al., 2019, Kumar & Dezfouli, 2019) or CoAP - Constrained Application Protocol (Shelby et al., 2014), general web protocols are in heavy usage in IoT. Improvement of these protocols can potentially bring benefits to the performance and security of IoT, which has been particularly challenged (Mahmoud et al., 2016).

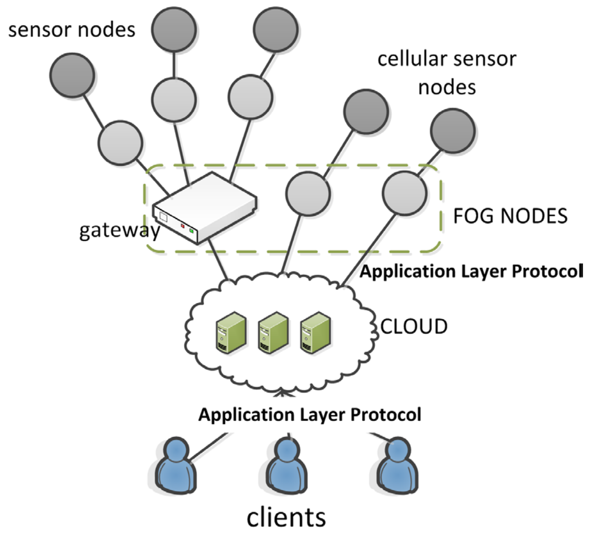

Figure 1. Typical IoT architecture

HTTP (HyperText Transfer Protocol) has come a long way from its initial version and use for basic modest web-pages. Today the needs for performance and security require complex protocols capable to serve multimedia and various web applications.

The goal of this paper is to evaluate the use of HTTP for different scenarios in IoT, focusing on the analysis of the data latency between fog nodes and the web server. The main aim is to test the new HTTP/3 protocol, whose standardisation is still under development and which provides a breakthrough in the HTTP evolution. For that purpose, a IoT infrastructure was created and different scenarios were configured.

The contributions of this paper are as follows:

- A network performance measurement methodology for IoT scenarios is built. It can be tuned according to the additional scenarios needed;
- The novell HTTP/3 protocol boosts the performance of IoT-related traffic, in comparison with the HTTP/2 protocol.

The rest of the paper is organised as follows. In Section 2, the problem statement is presented, along with the selected related works. Section 3 sets forth the methodology and describes the testbed. The results are described in Section 4. Section 5 concludes the paper and presents ideas for future work.

2. Background

Application-level network protocols in use for IoT fall into two categories: general-purpose protocols and specific protocols. While the HTTP is the main protocol in the first group, the second one is populated with several protocols, such as: MQTT, CoAT, XMPP (Extensible Messaging and Presence Protocol) and others. Extensive previews of the protocols utilised for IoT are given in (Čolaković & Hadžialić, 2018; Dizdarević et al., 2019, Al-Masri et al., 2020, Silva et al., 2021).

HTTP, the running force of World Wide Web (WWW), in its first version was a simple client-server pull protocol, using one TCP (Transmission Control Protocol) connection for transfer of one web-page object, from server to client. HTTP requires a reliable transfer to render the web content completely and without errors. Therefore, only TCP was considered as a transport layer protocol. Although WWW by definition is not a real-time service, the price of TCP connection establishment has proven to be significantly high. In the non-persistent HTTP (versions 0.9 and 1.0) every object (such as JavaScript code, GIF images, and text) was transferred using its separate connection. Since the Internet grew, the round time trip (RTT) became a significant element in the equation, increasing the whole TCP connection establishment time. Version 1.1 of HTTP brought about a plethora of optional features. However, many of them were never implemented.

The second implementation of HTTP was forged under the supervision of Google and Mozilla and inherited SPDY. HTTP/2 is supported by all major web browsers and by virtually all popular web-servers, such as Apache, Nginx and Internet Information Server. HTTP/2 support is usually not enabled by default but requires special configuration.

While HTTP/2 has brought about many improvements, the inherent burden of the TCP connection negotiation delay, as well as HOL blocking, remained as serious pitfalls (Oda & Yamaguchi, 2018). QUIC (Quick UDP Internet Connection) was initiated as a Google experimental protocol, which breaks up the continuity of TCP-based WWW and uses UDP (User Datagram Protocol). While in the beginning it was solely supported by Google services, QUIC swiftly expanded over Google's border, getting support on web-servers Caddy (Anon, 2021) and LiteSpeed (LiteSpeed Technologies Inc, 2021). The IETF renamed the term "HTTP over QUIC" as HTTP/3, therefore officially bringing about the next HTTP generation (IETF, 2018; Yong et al., 2017, Polese et al., 2019).

The first delay reduction is related to the three-way handshake, which is absent in UDP and therefore in HTTP/3 too. Another means of performance improvement is related to the TLS (Transport Layer Security) handshake, which involves an exchange of cryptographic data required for the connection encryption. It is important to state that TLS 1.3 handshake includes fewer steps than version 1.2 when using TCP or UDP. Various cases of connection establishment are given in Figure 2.

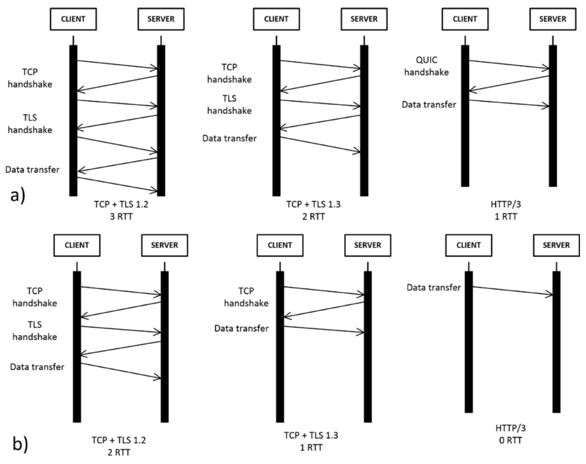

Figure 2. Different RTTs when (a) the client has never connected to the server, (b) the client has already connected to the server before

HTTP over QUIC was from the very beginning supported by Chrome and Opera. This option can be enabled/disabled through advanced settings. HTTP/3 is supported by various command-line tools, such as cURL and many libraries support both client and server.

The standardisation of HTTP/3 is still in progress. Currently (April, 2021) there is a 34th draft version published by the IETF (IETF, 2021).

Security is mandatory in HTTP/3. A special protocol called QUIC Crypto was designed by Google to be compatible with TLS 1.3.

2.2 Related Work

The issue of HTTP performance in IoT scenarios was investigated by Bziuk et al. (2018), with focus on version 1.1 with pipelining. This work concluded that HTTP 1.1 suffered from a significant overload due to handling multiple connections. It recommended further research to be directed towards newer protocols.

The issue of overhead was analysed by Yokotani & Sasaki (2016). They contrasted HTTP and MQTT and found out that the inherent properties of the TCP connection significantly degraded performances of HTTP.

Naik (2017) contrasted four protocols used in IoT: MQTT, CoAP, AMQP (Advanced Message Queuing Protocol) and HTTP/1.1 according to different criteria, such as message size, latency and reliability. This research also emphasised the issue of TCP connection overheads and its impact on the performance of the HTTP.

Elmangoush (2017) made a comparative review of 5 protocols used in IoT scenarios, and conducted measurement of performance of HTTP1 and HTTP2 under different channel conditions. HTTP2 proved to be superior in case of numerous concurrent requests, but added overhead with use of SSL (Secure Sockets Layer).

Currently, there is a modest number of papers dealing with the use of HTTP/3 for IoT. Mostly the focus is on the use of QUIC as a transport layer protocol.

Researchers have paired QUIC with native IoT protocols in order to achieve better security. Alqattaa & Loebenberger (2020) proposed a model for crypto-gateway using MQTT over QUIC with the goal of moving the overhead of encryption from constrained nodes to a gateway, which is connected to the cloud.

Liri et al. (2018) examined how robust IoT protocols are. It concluded that adapted QUIC protocol may greatly improve communication performance, which would enable QUIC to be a potential request-response IoT protocol alternative to CoAP.

Eggert (2020) explored the general feasibility of QUIC deployment for IoT infrastructure. The main reason for the use of QUIC was its inherent security. It concluded that the measured memory footprint, energy consumption, and performance were suitable for constrained devices, but with specific improvements.

3. Methodology

The basic idea for testing was to create an environment that would be closely mapped to realistic IoT scenarios and utilise web protocols. That implies the use of the wireless network, a typical IoT device acting as a fog node and a web server.

The established testbed consists of:

- Raspberry Pi 4 (Version B, 4 GB)
- Server (Ubuntu Server ver. 18.04)
- Wi-Fi router (802.11g)

The web-server used was Nginx, custom compiled using the quiche library to support HTTP/3. Nginx is an open-source web-server, capable of serving as a reverse proxy and load balancer.

The Raspberry Pi is used as a client. It is an inexpensive small computer, with support for a great number of peripherals and a common solution for different IoT based systems (Milošević et al., 2019). It has Raspbian OS installed and the cURL 7.75, a developer version, custom compiled with the quiche library. cURL is a versatile tool, supporting dozens of protocols and it is equipped with additional options for performance metrics. It is chosen also because of its strong support for HTTP/3 and its different implementations.

The environment where IoT devices work can cause packet loss and delay. Depending on the specific scenarios and service demands, these can be tolerated to a lesser or greater extent (Mocnej et al., 2017; Pekar et al., 2020).

To perform the tests in an environment which would be as realistic as possible, the delay and loss was simulated using the NetEm tool. (The Linux Foundation, 2021) NetEm is built using the existing Quality of Service (QoS) and Differentiated Services (diffserv) facilities in the Linux kernel. It is available out-of-the-box in Linux distributions and can be administered via the command line.

NetEm tool enables one to introduce various events in the network traffic, such as delay, packet reordering, and packet loss and so on.

Figure 3 shows a conceptual view of the testbed used. The cURL's timing in HTTP scenarios is reported using five variables (Cornwell, 2018): DNS Lookup, TCP handshake, SSL handshake, Wait and Data Transfer.

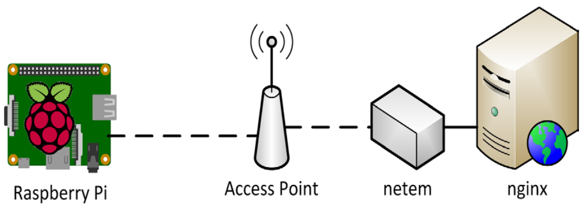

Figure 3. The testbed experiment

HTTP/1 without encryption has zero SSL handshake latency. HTTP/3, using UDP, has zero TCP handshake latency. HTTP/2 has both handshake and SSL latencies.

DNS Lookup can be a significant indicator and can greatly impact the whole timing. However, since the test was done in LAN environment and also DNS query response is cached, the DNS timing is not taken into account for the measurement.

Measured results are collected for each packet, where two metrics are used: response time and end-to-end delay. Response time represents the time required for web server to respond to user request and is calculated as the difference between the time when data transfer starts and the time when DNS name lookup is finished. End-to-end delay is calculated as the time difference between the total time and DNS name lookup time (as the DNS lookup was not relevant in this setup).

The scenarios involved three cases: lossless transfer, 10% packet loss and Gilbert Elliot model (Hasslinger & Hohlfeld, 2008) for three versions of HTTP (1, 2 and 3). The Gilbert-Eliot model was introduced with the following parameters: move-to-burstmode (p) of 1%, move-to-gapmode (r) of 10%, drop-in-burstmode (1-h) of 70% and drop-in-gap mode (1-k) of 0.1%.

In the first scenario, client and server communicate without packet loss, thus there are no additional delays in their communication, except due protocol overhead. In the second scenario packets are exchanged by client and server with 10% loss rate, thus delay will be introduced since transmit station will need a certain timeout before starting retransmission. In the last scenario, client and server communicate through the Gilbert-Elliot channel model that creates burst errors in transmitted packets. In this scenario packets will be received as incorrect so destination will immediately require packet retransmission from source thus delays are expected to be shorter than in the second scenario.

Testbed experiment is performed for two different data sizes representing two different types of fog nodes.

In the first case the upload of a very small file (30 B) is tested, which represents direct upload of data from fog node to web server. This kind of scenario is typical for singular IoT nodes, which are connected directly to the Cloud, where a certain amount of data processing is performed on the fog node (Figure 1), which is typically implemented as part of the IoT node itself.

In the second case the upload of a 3.5 KB file is tested, which can be taken as an example of cumulative data that a fog node collects from multiple sensors and then sends to the webserver. This kind of scenario is typical when multiple IoT nodes send data items to a fog node, which performs data processing and bundles data in a larger data unit before sending it to the web server.

4. Results and Discussion

The timings logged with cURL are serialised into a CSV file and then processed using Matlab, which contains around 3000 packets for every scenario. Since some of the measured values represent outliers, those values which amount to more than three scaled median absolute deviations (MAD) are removed from measured results.

The web server response time is displayed for all three scenarios (lossless, 10% packet loss, the Gilbert-Elliot channel model) in Table 1 for 30-byte files, and also in Table 2 for 3.5 KB files. Results are shown in form of mean deviation and standard deviation value of measured results for every scenario.

4.1 Response Time for Two File Sizes

Response time is given for a file of 30 B and one of 3.5 KB in Tables 1 and 2.

Obtained results for web response times are the shortest for HTTP/1 and HTTP/3 in all scenarios for both file sizes, while HTTP/2 protocol obtains a significantly longer web response time. HTTP/1 has a short response time since it does not use authentication.

Table 1. Web server response time for a 30 B file

Scenario	HTTP/1	
	Mean	St. Deviation
Lossless	52.86 ms	47.19 ms
10% packet loss	52.68 ms	49.57 ms
Gilbert-Elliot	20.22 ms	35.23 ms
	HTTP/2	
Lossless	88.50 ms	31.19 ms
10% packet loss	107.07 ms	54.94 ms
Gilbert-Elliot	41.74 ms	45.88 ms
	HTTP/3	
Lossless	27.26 ms	31.47 ms
10% packet loss	54.50 ms	46.33 ms
Gilbert-Elliot	55.21 ms	41.77 ms

Table 2. Web server response time for a 3.5 KB file

Scenario	HTTP/1	
	Mean	St. Deviation
Lossless	44.43 ms	42.88 ms
10% packet loss	50.84 ms	50.19 ms
Gilbert-Elliot	62.06 ms	49.21 ms
	HTTP/2	
Lossless	99.25 ms	35.60 ms
10% packet loss	137.94 ms	79.68 ms
Gilbert-Elliot	85.52 ms	98.55 ms
	HTTP/3	
Lossless	46.45 ms	35.09 ms
10% packet loss	54.93 ms	47.43 ms
Gilbert-Elliot	56.57 ms	45.63 ms

Due to the new TLS 1.3 handshaking mechanism HTTP/3 is a close match for non-secure HTTP/1. On the other hand, HTTP/2 has a significantly higher web response time due to TLS 1.2 handshaking mechanism which involves three-step handshake mechanisms. Also, in scenarios which include complete packet loss or damaged packets with burst errors, HTTP/3 significantly outperforms HTTP/2, thanks to the new QUIC protocol.

4.2 End-to-end Upload Delay for a 30 B File

Results for end-to-end delay for the transfer of a small file are presented in Table 3, while these values are also presented in the form of histograms of end-to-end delay for all protocols displayed on the same graph.

Table 3. End-to-end delay for the upload of small packets (30 B) in different scenarios

Scenario	HTTP/1	
	Mean	St. Deviation
Lossless	65.4 ms	50.5 ms
10% packet loss	82.3 ms	78.6 ms
Gilbert-Elliot	64.3 ms	51.5 ms
	HTTP/2	
Lossless	96.6 ms	31. 4 ms
10% packet loss	130.5 ms	70.2 ms
Gilbert-Elliot	65.8 ms	54.5 ms
	HTTP/3	
Lossless	67.5 ms	51.1 ms
10% packet loss	77.7 ms	52.5 ms
Gilbert-Elliot	73.5 ms	45.9 ms

In the case of the lossless upload transmission from a node to the server (Figure 4), results indicate that HTTP/1 protocol has the shortest end-to-end delay followed closely matched by HTTP/3 protocol. HTTP/1 has a shorter end-to end delay since it does not use any form of security, like HTTP/2 and HTTP/3 protocols do. According to the results, HTTP/3 outperforms HTTP/2 which on average has a 43% longer end-to-end delay, due to the newer TLS 1.3 handshake mechanism.

In the case of the 10% packet loss transmission from node to server (Figure 5), results indicate that the new HTTP/3 protocol has the shortest end-to-end delay, followed closely by HTTP/1 protocol. The results show that HTTP/3 outperforms HTTP/2 which on average has a 68% longer end-to-end delay, due to the QUIC protocol which is based on UDP unlike the other two protocols which are based on TCP protocol.

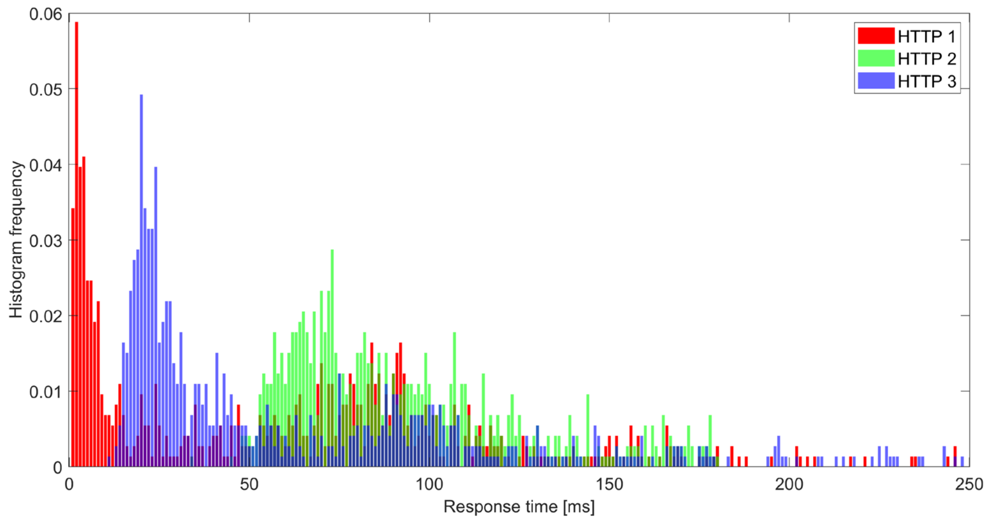

Figure 4. Lossless upload of a very small file (30 B)

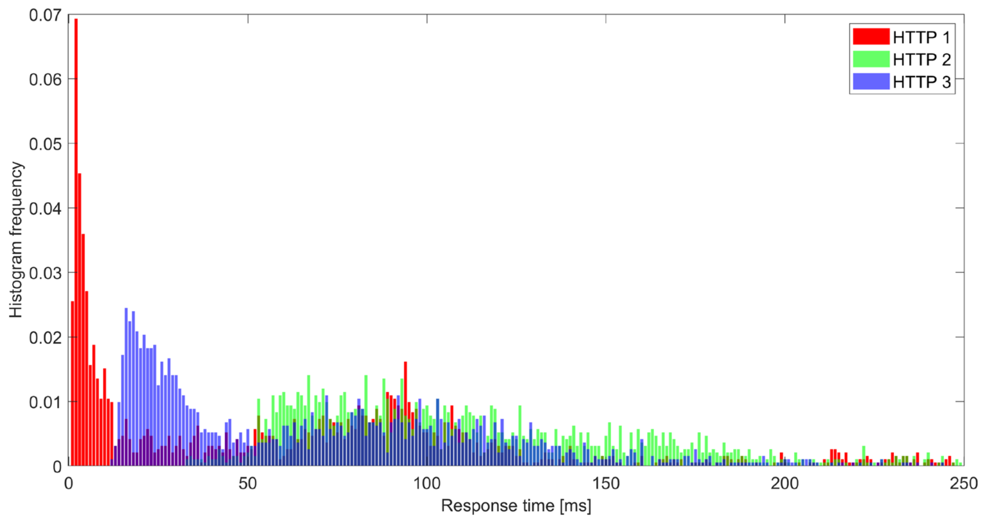

Figure 5. Upload of a very small file (30 B) with 10% packet loss

In the case of packet transmission modelled by the Gilbert-Elliot channel model (Figure 6), results indicate that HTTP/1 protocol has the shortest end-to-end delay, followed closely by HTTP/2 protocol. The results indicate that HTTP/3 has on average a 11% longer end-to-end delay than HTTP/2. The slight increase of end-to-end delay in HTTP/3 protocol is due to connectionless UDP transfer protocol, which does not start retransmission immediately when damaged packet loss is detected as connection-oriented TCP protocol does for HTTP/1 and HTTP/2 protocols.

4.3 Upload of a 3.5 KB File

The measurements results, presented in Table 4, show the distribution of end-to-end delay for a packet size of 3.5 KB for all three protocols in different simulation scenarios.

Table 4. End-to-end delay for the upload of a 3.5 KB file in different scenarios

Scenario	HTTP/1	
	Mean	St. Deviation
Lossless	53.2 ms	44.27 ms
10% packet loss	79.7 ms	78.3 ms
Gilbert-Elliot	70.0 ms	51.3 ms
	HTTP/2	
Lossless	110.9 ms	36. 2 ms
10% packet loss	184.9 ms	110.4 ms
Gilbert-Elliot	239.5 ms	233.3 ms
	HTTP/3	
Lossless	78.3 ms	44.9 ms
10% packet loss	94.8 ms	57.8 ms
Gilbert-Elliot	84.6 ms	60.8 ms

In the case of the lossless upload transmission from node to server (Figure 7), results indicate

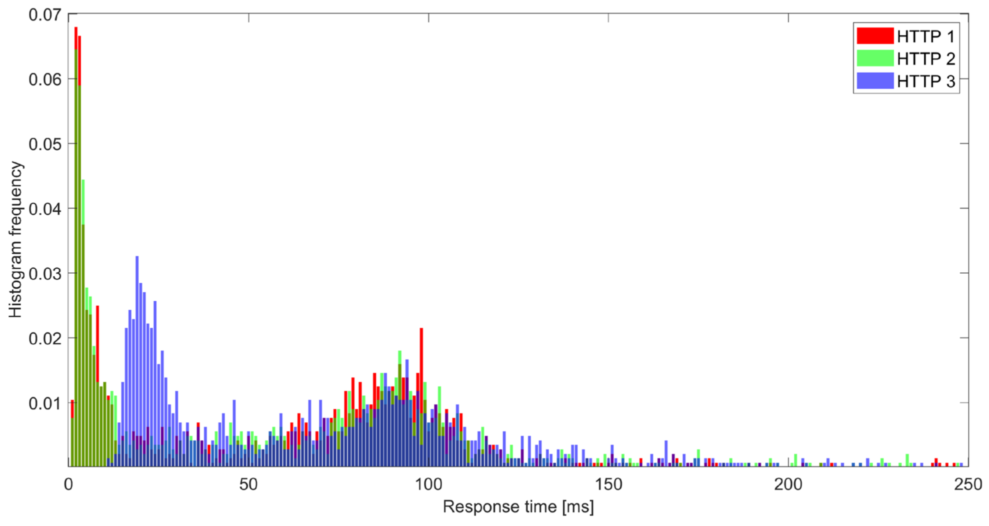

Figure 6. Upload of a very small file (30 B) modelled with the Gilbert-Elliot channel model

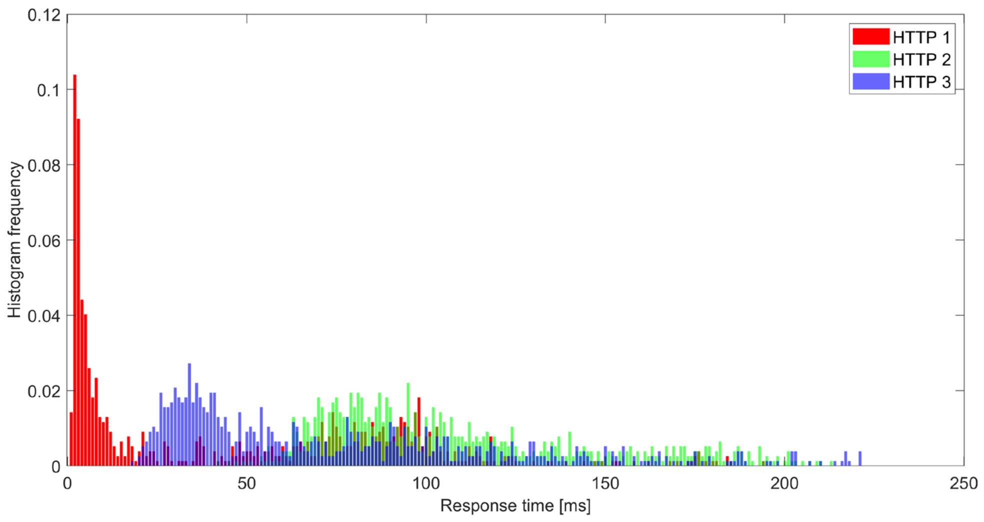

Figure 7. Lossless upload of a 3,5 KB file

that HTTP/1 protocol has the shortest end-to-end delay, while HTTP/3 protocol has a 47% longer end-to-end delay. According to the results, HTTP/3 outperforms HTTP/2 which on average has a 41% longer end-to-end delay. This delay originates from TLS security overhead in HTTP/3 and HTTP/3 protocols.

In the case of the 10% packet loss transmission from node to server (Figure 8), results indicate that HTTP/1 protocol has the shortest end-to-end delay, followed closely by HTTP/3 protocol with a 19% longer end-to-end delay time. According to the results, HTTP/3 outperforms HTTP/2 which on average has a two times longer end-to-end delay. The main reason for it is the problem with HOL (Head-of-Line) blocking in the case of lost packet block reception queue.

In the case of the packet transmission modelled by the Gilbert-Elliot channel model (Figure 9), results indicate that HTTP/1 protocol has the shortest end-to-end delay, followed closely by HTTP/3 protocol with a 20% longer end-to-end delay. According to the results, HTTP/3 outperforms HTTP/2 which on average has a three times longer end-to-end delay. The reason for that is the problem in the case of HOL blocking when lost packet block reception queue.

5. Conclusions

By comparing the results obtained by the protocols HTTP/2 and HTTP/3 as they are depicted in the Tables 3 and 4, it can be stated that the HTTP/3 protocol significantly outperforms the HTTP/2 protocol version in majority of scenarios (9

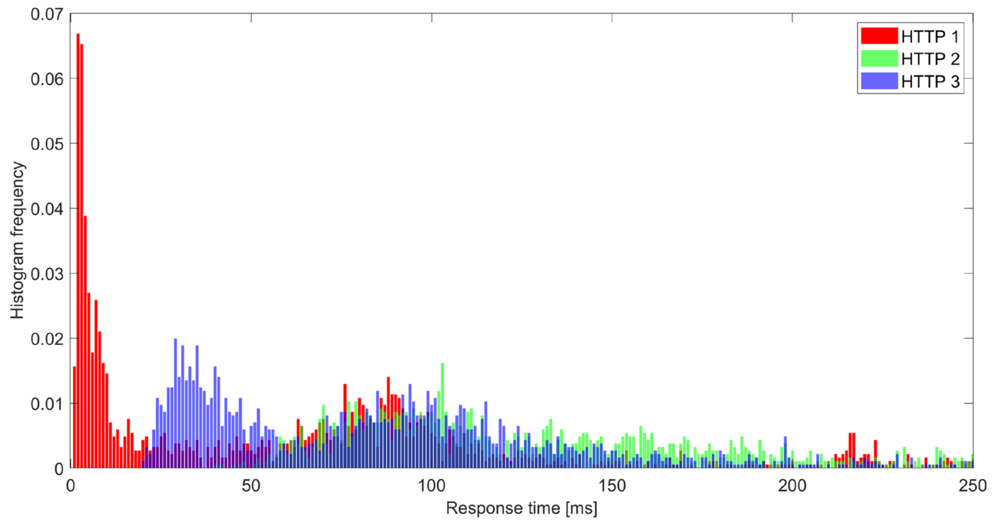

Figure 8. Upload of a 3.5 KB file with 10% packet loss

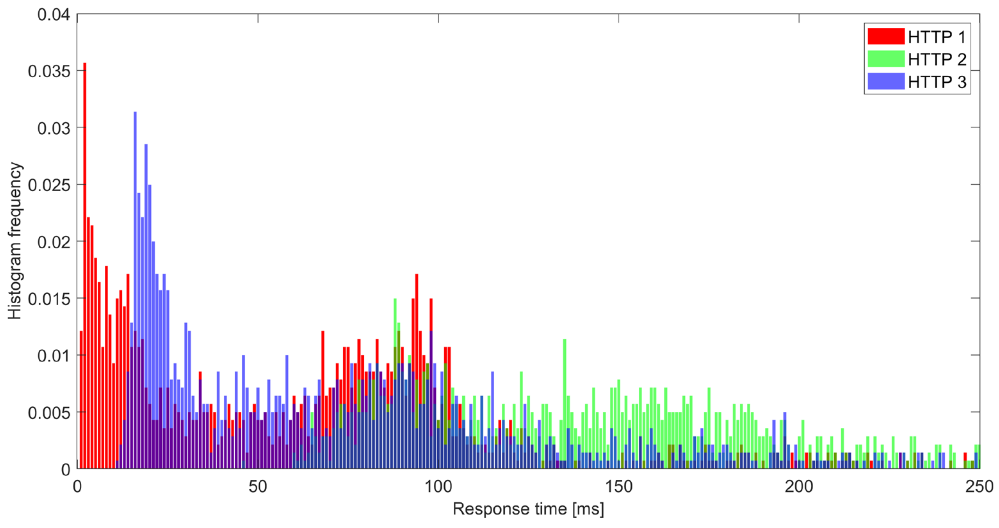

Figure 9. Upload of a 3.5 KB file modelled with Gilbert-Elliot channel model

out of 12), in terms of end-to-end delay, which significantly affects the responsiveness of implemented IoT applications. Compared to the HTTP/1 protocol, which offers no security unlike the new HTTP/3 protocol, the latter almost matches the end-to-end delay performance of the former.

The results of the experiment show that HTTP/3 protocol has successfully solved problems faced by other versions of the HTTP, and still improved end-to-end delay, which is one of the main performance evaluation parameters for IoT applications.

On the other hand, as encryption is mandatory for HTTP/3, the security is radically improved, in comparison with HTTP 1. Introducing client authentication (also built in TLS) can further foster security in specific scenarios with clients of a greater level of mobility. It is of essential importance to bring the standardization process to its end and have stable versions of both client and server implementations of HTTP/3.

As future work, the issue of power consumption related to the new protocol use should be investigated. Also, additional scenarios for IoT and performance testing including IoT specific protocols should be implemented. To that, additional scenarios will be investigated, to find out the potential performance gains with regard to intensive data traffic, such as multimedia streaming.

Acknowledgements

This study is supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, and these results were obtained as part of the Grant No. 451-03-9/2021-14/200132 for the University of Kragujevac - Faculty of Technical Sciences Čačak.

REFERENCES

Al-Masri, E., Kalyanam, K. R., Batts, J., Kim, J., Singh, S., Vo, T. & Yan, C. (2020). Investigating Messaging Protocols for the Internet of Things (IoT), *IEEE Access*, *8*, 94880-94911. DOI: 10.1109/ACCESS.2020.2993363

Alqattaa, A. & Loebenberger, D. (2020). An IoT Gateway for Resource-Constrained IoT Devices. In *CLOUD COMPUTING 2020: The Eleventh International Conference on Cloud Computing, GRIDs, and Virtualization*, Nice (pp. 50-54).

Anon (2021). *Caddy Server*. Available at: <https://caddyserver.com/>.

Banks, A., Briggs, E., Borgendale, K. & Gupta R. (2019). *MQTT 5.0 specification*. Available at: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>, last accessed: 10th Oct. 2020.

Bziuk, W., Phung, C. V., Dizdarević, J. & Jukan, A. (2018). On HTTP Performance in IoT Applications: An Analysis of Latency and Throughput. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),* Opatija, Croatia (pp. 0350-0355). DOI: 10.23919/MIPRO.2018.8400067.

Chiang, M. & Zhang, T. (2016). Fog and IoT: An Overview of Research Opportunities, *IEEE Internet of Things Journal*, *3*(6), 854-864. DOI: 10.1109/JIOT.2016.2584538

Čolaković, A. & Hadžialić, M. (2018). Internet of Things (IoT): A review of enabling technologies, challenges, and open research questions, *Computer Networks, 144*, 17-39. DOI: 10.1016/j.comnet.2018.07.017

Cornwell, P. (2018). A question of timing, *The Cloudflare Blog*. Available at: <https://blog.cloudflare.com/a-question-of-timing/>, last accessed: 30th Mar. 2021.

Dizdarević, J., Carpio, F., Jukan, A. & Masip-Bruin, X. (2019). A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration, *ACM Computing Surveys 51*, 1-30. DOI: 10.1145/3292674

Eggert, L. (2020). Towards Securing the Internet of Things with QUIC. In *Network and Distributed Systems Security (NDSS) Symposium*, 23-26 February, San Diego, CA, USA. Available at: <https://easychair.org/publications/preprint_download/68D2>, last accessed: 30th Mar. 2020.

Elmangoush, A. (2017). Evaluating the Features of HTTP/2 for the Internet of Things. In *1st Conference of Industrial Technology (CIT2017)*, Misurata, Libya (pp. 202-207).

Hasslinger, G. & Hohlfeld, O. (2008). The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet. In *14th GI/ITG Conference - Measurement, Modelling and Evaluation of Computer and Communication Systems*, Dortmund, Germany (pp. 1-15).

IETF (2018). Identifying our deliverables, *IETF Mail Archive.* Available at: <https://mailarchive.ietf.org/arch/msg/quic/RLRs4nB1lwFCZ_7k0iuz0ZBa35s/>, last accessed: 1st Apr. 2021.

IETF (2021). *Hypertext Transfer Protocol Version 3 (HTTP/3)*. Available at: <https://tools.ietf.org/html/draft-ietf-quic-http-34>, last accessed: 1st Apr. 2021.

Kumar, P. & Dezfouli, B. (2019). Implementation and Analysis of QUIC for MQTT, *Computer Networks*, *150*, 28-45. DOI: 10.1016/j.comnet.2018.12.012

Liri, E., Singh, P. K., Rabiah, A. B., Kar, K., Makhijani, K. & Ramakrishnan, K. K. (2018). Robustness of IoT Application Protocols to Network Impairments. In *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, (pp. 97-103). DOI: 10.1109/LANMAN.2018.8475048

LiteSpeed Technologies Inc. (2021). *QUIC Support in LiteSpeed*. Available at: <https://www.litespeedtech.com/http3-solutions>, last accessed: 1st Apr. 2021.

Mahmoud, R., Yousuf, T., Aloul, F. & Zualkernan, I. (2016). Internet of things (IoT) security: Current status, challenges and prospective measures. In *10th International Conference for Internet Technology and Secured Transactions, ICITST 2015* (pp. 336 -341).

Milošević, M., Četić, N., Kovačević, J. & Anđelić, T. (2019). Lighting Control Using Raspberry Pi and Oblo Living Home Automation System, *Serbian Journal of Electrical Engineering*, *16*(1), 45-54. DOI: 10.2298/SJEE1901045M

Mocnej, J., Pekar, A., Seah, W. K. G. & Zolotova, I. (2018). *Network Traffic Characteristics of the IoT Application Use Cases*. School of Engineering and Computer Science, Victoria University of Wellington.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, (pp. 1-7). DOI: 10.1109/syseng.2017.8088251

Oda, N. & Yamaguchi, S. (2018). HTTP/2 performance evaluation with latency and packet losses. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, USA (pp. 1-2). DOI: 10.1109/CCNC.2018.8319285.

Pekar, A., Mocnej, J., Seah, W. K. G. & Zolotova, I. (2020). Application Domain-Based Overview of IoT Network Traffic Characteristics, *ACM Computing Surveys, 53*(4), 1-33.

Polese M., Chiariotti, F., Bonetto, E., Rigotto, F., Zanella, A. & Zorzi, M. (2019). A Survey on Recent Advances in Transport Layer Protocols, *IEEE Communications Surveys & Tutorials*, *21*(4), 3584-3608. DOI: 10.1109/COMST.2019.2932905

Shelby, Z., Hartke, K. & Bormann, C. (2014). *The Constrained Application Protocol (CoAP)*. Available at: <https://tools.ietf.org/html/rfc7252>, last accessed: 12th Dec. 2020.

Silva, D., Carvalho, L. I., Soares, J. & Sofia, R. C. (2021). A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA, *Applied Sciences*, *11*, 4879. DOI: 10.3390/app11114879

Stojmenovic, I. & Wen, S. (2014). The Fog computing paradigm: Scenarios and security issues. In *Federated Conference on Computer Science and Information Systems*, Warsaw, Poland (pp 1 - 8). DOI: 10.15439/2014F503

The Linux Fondation. (2021). *NetEM*. Available at: <http://www.linuxfoundation.org/collaborate/workgroups/networking/netem>, last accessed: 30th Mar. 2021.

Xia, F., Yang, L. T., Wang, L. & Vinel, A. (2012). Internet of Things, *International Journal of Communication Systems*, *25*, 1101-1102. DOI: 10.1002/dac.2417

Yokotani, T. & Sasaki, Y. (2016). Comparison with HTTP and MQTT on required network resources for IoT. In *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, (pp. 1-6). DOI: 10.1109/ICCEREC.2016.7814989

Yong C., Tianxiang, L. & Cong, L. (2017). Innovating Transport with QUIC: Design Approaches and Research Challenges, *IEEE Internet Computing 21(2)*, 72-76. DOI: 10.1109/MIC.2017.44