LEVERAGING HTTP/3 FOR EFFICIENT RESTFUL API COMMUNICATION

Jiří Balej[1], Andrej Juríčka[1], Jiří Passinger[1]

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

ABSTRACT

The rise of IT systems automation required the shift of the management of network devices and servers to machine-oriented interfaces, such as REST APIs. It enables large-scale batch requests but also creates challenges such as latency and limited concurrency handling. This paper investigates the potential benefits of adopting HTTP/3 for REST-based network and server management, examining its impact on communication efficiency. Specifically, the research compares different versions of HTTP in terms of communication performance, latency, message count, and execution speed, while exploring the advantages and challenges of such solutions. The findings provide valuable information on the integration of HTTP/3 into modern network management practices, with the aim of optimising automation and improving performance across IT infrastructures.

Keywords: HTTP/3, QUIC, REST API, RESTCONF, server, network, management

JEL Code: L86, C88, O31

1 INTRODUCTION

Traditionally, servers, services, and network devices have been managed through SSH or web interfaces. However, with the growing adoption of IT systems management automation, machine-oriented interfaces have gained significant importance. These interfaces enable rapid, large-scale batch requests to devices, improving efficiency in the management of IT infrastructure. However, this approach introduces challenges related to response times and the ability of managed devices to handle large volumes of concurrent requests effectively.

Although SSH (Secure Shell) remains one of the primary protocols for network automation, its usage is typically limited to executing standard command-line instructions through automation tools such as Ansible, Chef, or Puppet. Legacy protocols such as SNMP (Simple Network Management Protocol) and TR-069 (Technical Report 069), once popular for device configuration, have become largely outdated due to their limited scalability and security. On the contrary, modern automation interfaces increasingly utilise NETCONF (Network Configuration Protocol), RESTCONF (RESTful Configuration Protocol), and general REST APIs (Representational State Transfer Application Programming Interfaces). Both the RESTCONF and REST APIs rely on HTTPS as their underlying communication layer.

https://doi.org/10.11118/978-80-7701-047-4-0008

The evolution of the HTTP protocol raises the question whether adopting the modern HTTP/3 version could improve communication efficiency for REST-based management services. In the case of network devices, the HTTP version is typically fixed, with HTTP/1.1 remaining the most widely used. However, for servers and service interfaces, it is essential to evaluate which HTTP version is best suited to optimise request-response performance in management operations. The benefits of HTTP/2 and HTTP/3 over traditional HTTP/1.1 in terms of speed, parallelism, and latency can be crucial factor in enhancing automation efficiency in modern IT environments.

Therefore, this paper explores how the adoption of HTTP/3 can improve REST API communication and its impact on communication latency. Specifically, the research addresses the following questions:

- Research Question 1 (RQ1): Can HTTP/3 reduce latency and enhance performance for large-scale network management operations?
- Research Question 2 (RQ2): How does the performance of network and server management differ between HTTP/1.1, HTTP/2, and HTTP/3 in terms of message count and execution speed?
- Research Question 3 (RQ3): What challenges might arise when migrating existing network management tools to HTTP/3?

These questions aim to provide insights into the benefits and potential limitations of adopting HTTP/3 for network management tasks.

2 RELATED WORKS

The current principles of network device automation are primarily based on the NETCONF and RESTCONF protocols, utilising JSON and XML as the primary data formats (Abuelanain, 2021). While RESTCONF is specifically designed for network management, REST APIs are widely used beyond this scope, serving as interfaces for data exchange between various systems and services. Consequently, this chapter will first explore the foundational principles of REST APIs and methodologies for their testing. The second part will focus on recent research advancements related to the HTTP protocol.

2.1 REST API

The systematic review of the literature on current methodologies and challenges in RESTful API testing was carried out by Ehsan (2022) and Golmohammadi (2023). Wu (2022) provided an in-depth analysis of the number of operations required in specific scenarios and the interaction of various parameters in each operation.

Kim (2023) reviewed existing REST API testing tools and proposed an adaptive testing technique using reinforcement learning. Gowda (2024) compared response times across different REST APIs, applying a developer-focused method to assess both performance and security. Most recent studies on REST API testing treat APIs as black boxes, with one of the latest works by Poth (2024) focusing on contemporary performance evaluation.

2.2 HTTP protocol

The use of HTTP/3 as a communication protocol for infrastructure and management tasks was proposed and validated by Saif (2021) through the MQTT protocol. Michel (2023) explored the concept of SSH over HTTP/3, highlighting key advantages such as faster session establishment and reduced response times. These two studies demonstrate the potential of HTTP/3 to improve device management protocols.

Perna (2022) conducted one of the first comprehensive evaluations of HTTP/3 performance. Gahtan (2024) performed extensive testing of HTTP/3 responses on the open Internet, analysing approximately 7 million images to estimate response times. Ravuri (2023) provided a practical example by implementing an interactive service using HTTP/3. Gupta (2024) investigated content delivery prioritisation in HTTP/3, focusing on minimizing head-of-line blocking and evaluating Quality of Experience (QoE) across various websites.

The security issues in networks were evaluated by Kashtalian (2023), where the problem of their detection was investigated. Attacks on HTTP/3 were comprehensively reviewed by Chatzoglou (2023), where all current QUIC libraries and applications were tested.

3 REST API MANAGEMENT OVER HTTP/3

To effectively compare the efficiency, responsiveness, latency, and security of REST API communication across various HTTP versions, it is essential to dive deeper into the structure and principles of REST APIs.

3.1 REST API management

Representational State Transfer (REST) has become a standard interface not only for application data exchange but also for device management. Originally proposed by Fielding (2000), but the structure of REST APIs are now primarily standardized by OpenAPI (2024).

Communication with REST APIs is carried out using the standard HTTP protocol, predominantly through its secure variant, which utilizes SSL/TLS layer. REST APIs support all CRUD operations (Create, Read, Update, Delete) through the standard HTTP methods such as POST, GET, PUT, and DELETE. One of these methods is specified in the initial part of the request, followed by the resource identification via a unique URL and the used HTTP version. When creating or updating a resource, the message body contains the relevant data, typically in JSON or YAML format.

The server‘s response includes the HTTP version, a three-digit status code, and a status message. In the case of data retrieval, the requested content is also included in the message body. Both the request and response may contain additional headers, such as client/server versions, timestamps, accepted content types, and more. The fundamental structure of the mandatory components is illustrated in Figure 1.

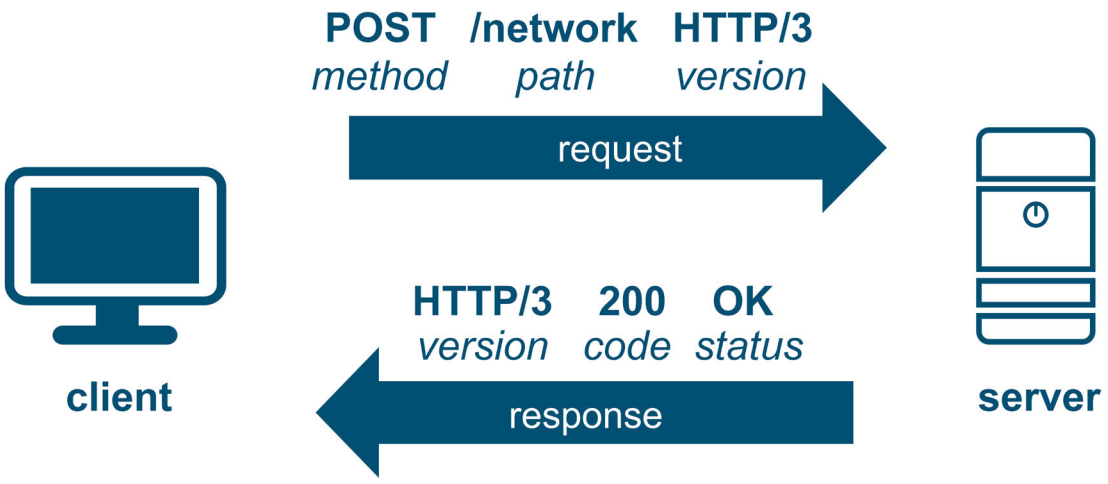

Fig. 1: REST API request and response structure

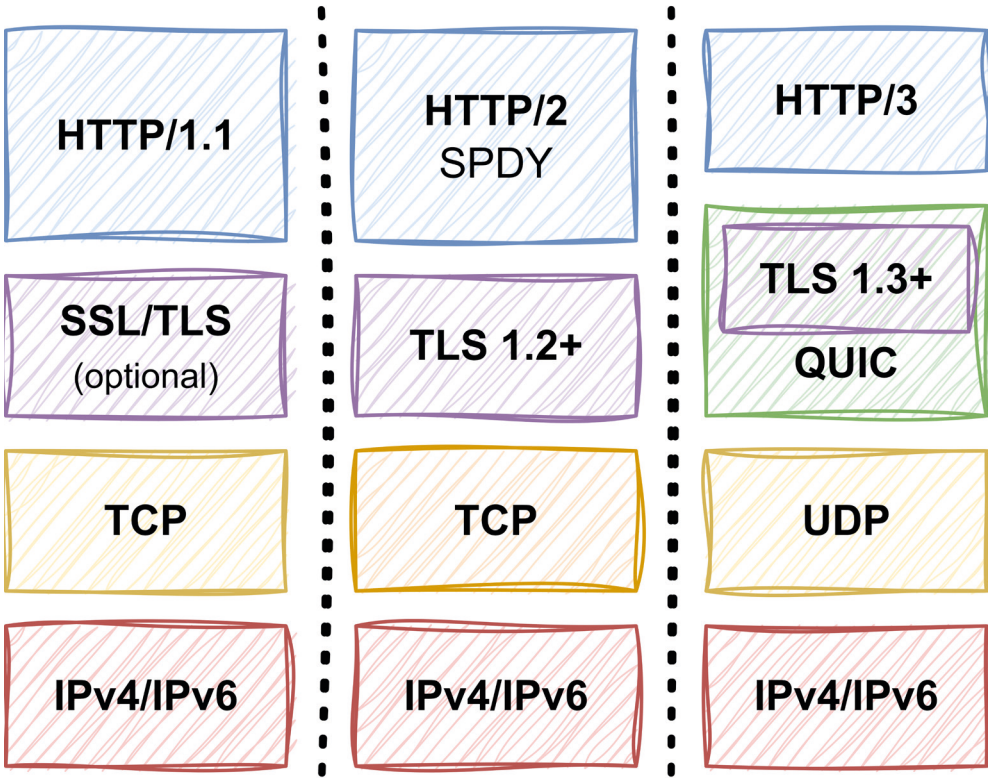

Fig. 2: Protocols stack by OSI layers for various versions of HTTP

3.2 HTTP development

Over the past decade, significant progress has been made in the evolution of HTTP protocols. The widely adopted HTTP/1.1, which dominated the web for over 20 years, has become less efficient in meeting the demands of modern services and networks.

The first major advancement was HTTP/2, developed based on the SPDY protocol draft. This version introduced several significant improvements, including mandatory TLS use, header compression, binary encoding, enhanced prioritization, and server push mechanisms. However, HTTP/2 still relies on TCP as its transport layer protocol and typically uses TLS 1.2 or TLS 1.3.

HTTP/3, the latest version, made a significant shift by replacing TCP with UDP as the transport protocol and using new protocol QUIC responsible for connection handling and TLS security layer. This change addresses the limitations of TCP's congestion control algorithms, which can slow down data exchange in cases of packet loss. HTTP/3 also mandates the use of TLS, specifically version 1.3. Although HTTP/3 was officially standardized only recently (Iyengar, 2021; Bishop, 2022), it has been in practical use for more than five years and currently it is adopted by approximately 30–40% of servers, according to W3Tech statistics. Figure 2 illustrates the protocol structures used for HTTP communication across different versions.

3.3 HTTP message exchange

The performance of communication is directly influenced by the number of exchanged message pairs (request-response), as each subsequent pair must wait for the previous one to complete. The delay caused by a single exchange is referred to as Round Trip Time (RTT), which includes the time taken to deliver a message, generate a response, and transfer it back. The request-response message pairs are illustrated in Figure 3 (Marx, 2021).

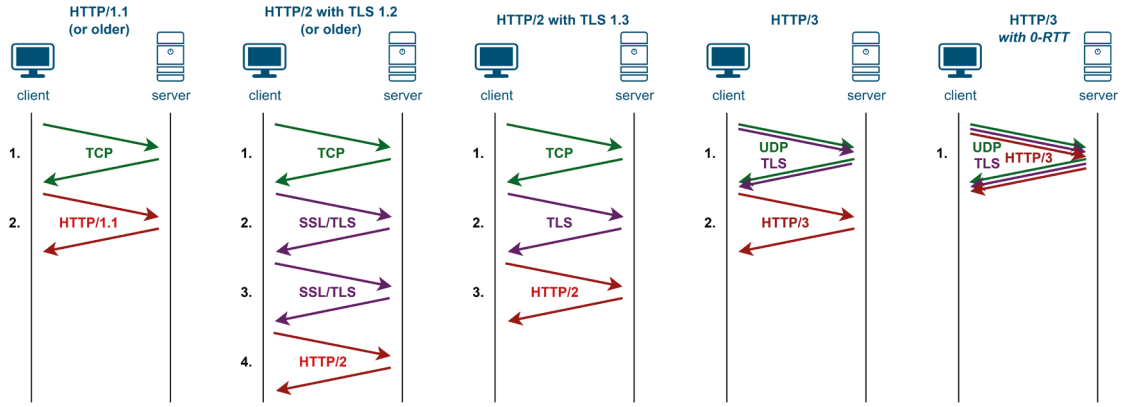

Fig. 3: HTTP versions session establishment communication
Source: Marx, 2021)

In the first versions of HTTP (as shown in the first example in Figure 3), establishing a TCP session required one RTT, while an additional RTT was necessary for the actual data transfer. This exchange did not include a security layer (TLS/SSL), which is now essential for modern communication.

The introduction of HTTPS added two additional message exchanges to the process, increasing the total RTTs to four (second example in Figure 3). However, with the advent of TLS version 1.3, only one message exchange is needed to establish a secure channel, reducing the total number of RTTs to three (third example in Figure 3).

HTTP/3 eliminates the use of the TCP protocol, which required the first RTT for connection establishment. Instead, channel establishment and TLS negotiation are completed in a single exchange (fourth example in Figure 3). Additionally, HTTP/3 supports a method called 0-RTT, which enables session resumption based on a previously established secure connection. However, 0-RTT introduces vulnerabilities, such as replay attacks, and is considered less secure.

1.1 HTTP communication experiment

The theoretical concepts described above were tested in a real-world environment. To validate our hypotheses, we conducted experiments using different HTTP versions and evaluated their security aspects.

We used the curl client, with supports of all current HTTP versions. As the target was used cloudflare-quic.com, which also supports all current HTTP versions. To minimize the size of request and response we used method HEAD, which grabs only website header. All communication finished with status code 200 OK, but with plain HTTP (non-secure), the server responds with a 301 (Moved Permanently) status code. This response does not affect the validity of our results.

HTTP version negotiation and communication details were verified through the verbose output of the curl command. Additionally, all exchanged messages were captured using Wireshark and analyzed with various tools.

Figure 4 shows the resulting flow graph. We highlighted the key differences in total communication time and message direction. Rather than counting individual packets, we focused on communication directions, as the primary delays sources from message transmission between communication partners.

To maintain relevance to data retrieval performance, we limited the analysis to the point of receiving the HTTP response status code, excluding the additional 3–4 message exchanges typically involved in closing a TCP session—since those steps do not impact the retrieval of data itself.

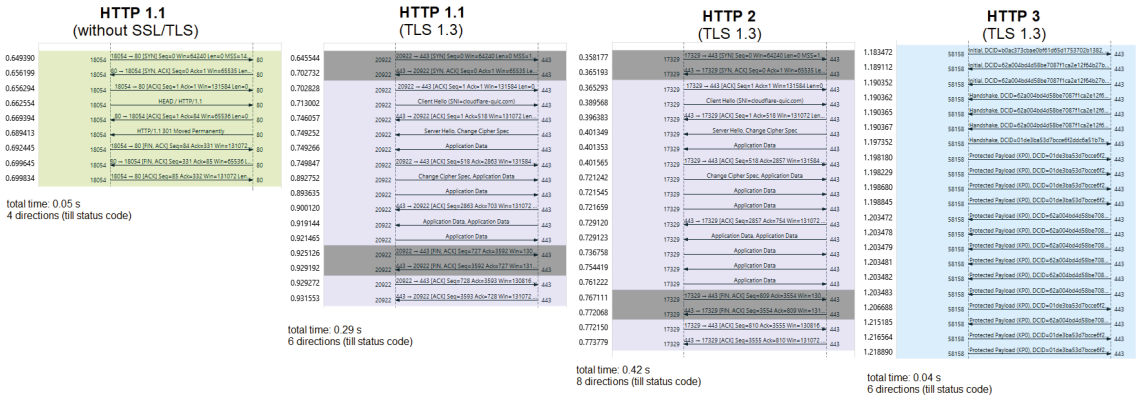

Fig. 4: Retrieving website header (method HEAD) using various HTTP versions

Data in Figure 4 clearly shows that HTTP/1.1 without TLS and HTTP/3 with TLS are comparable in terms of total communication time. While the non-secure HTTP required fewer message exchanges overall, HTTP/3—despite having the highest number of individual messages—achieved communication in just six directional exchanges. In our assessment, HTTP/3 outperforms the other secure HTTP variants in this experiment, offering better efficiency in terms of both speed and message flow structure.

4 RESULTS

4.1 Research Question 1

- Can HTTP/3 reduce latency and enhance performance for large-scale network management operations?

HTTP/3 can reduce latency by eliminating the need for the TCP protocol and enabling 0-RTT (Zero Round-Trip Time) connections. It also offers improved concurrency handling compared to HTTP/2 and older versions. Furthermore, the use of the UDP protocol allows faster recovery from packet loss than TCP, contributing to enhanced overall performance.

4.2 Research Question 2

- How does the performance of network and server management differ between HTTP/1.1, HTTP/2, and HTTP/3 in terms of message count and execution speed?

Modern HTTP versions, such as HTTP/2 and HTTP/3, enable multiplexing, which enhances performance by addressing the issue of head-of-line blocking. HTTP/3, in particular, reduces the number of exchange messages compared to older versions, further improving both performance and execution speed.

Our experimental results confirm that HTTP/3 outperforms both secure HTTP/1.1 and HTTP/2 in terms of total required time. Interestingly, HTTP/2 was observed to be slower than HTTP/1.1 in our test scenarios, which may be attributed to its more complex connection management and dependency handling.

4.3 Research Question 3

- What challenges might arise when migrating existing network management tools to HTTP/3 or QUIC?

One of the main challenges with network devices and systems is the lack of support and willingness from vendors to adopt newer technologies. However, HTTP/3 is now well-established, widely used and there are available many public libraries to facilitate integration. Despite its growing adoption, issues may still arise during implementation. Additionally, the use of UDP instead of TCP in HTTP/3 remains relatively untested in some environments, which may pose challenges for stability and performance in certain network conditions.

5 CONCLUSIONS

RESTful APIs are widely applied in various domains, including IoT, cloud applications, services and device management, where efficient, scalable, and loosely coupled communication is necessary. These APIs have also become a foundation for popular web frameworks and technologies, enabling seamless integration of third-party services, and enhancing interoperability in modern software ecosystems.

The findings of the paper highlight the potential of HTTP/3 for REST APIs, offering benefits such as reduced latency and enhanced performance. By leveraging UDP protocol and introducing features such as 0-RTT connections, HTTP/3 addresses limitations of previous HTTP versions, particularly in handling packet loss and reducing exchange overhead. Comparisons of HTTP version reveal that modern ones like HTTP/2 and HTTP/3 benefit from advancements such as multiplexing and binary format, which improve execution speed and efficiency and at the same time secure the communication.

While HTTP/3 presents promising performance improvements for network and server management, transitioning existing tools to these protocols involves challenges such as compatibility issues, devices updates, and new security considerations (Kuhlewind, 2022). Additional testing and evaluation is necessary before implementing the protocol for device management.

A logical extension of this research involves testing the reliability of REST APIs over HTTP/3 under a variety of conditions. Future work will focus on evaluating performance across different network scenarios, including varying levels of delay, bandwidth, and packet loss, to assess how HTTP/3 handles adverse environments. The impact of concurrent client requests on API responsiveness and throughput will also be measured, providing insights into scalability. Additionally, we plan to investigate how the size of HTTP requests—particularly in Create and Update operations—affects performance. Another important aspect will be verifying the current level of support for REST APIs over HTTP/3 and HTTP/2 in existing network devices and server platforms. Finally, special attention will be given to cybersecurity, specifically through testing the security implications of the 0-RTT (zero round-trip time) feature when used in RESTful API communications.

REFERENCES

ABUELANAIN, K., DOYLE, J., KARNELIUK, A., JAIN, V. 2021. *Network Programmability and Automation Fundamentals.* Cisco Press, New Jersey, USA, 800 pages. ISBN: 978-1-58714-514-8

BISHOP, M. 2022. RFC 9114. HTTP/3. Jun. 2022. RFC *Editor*. https://www.rfc-editor.org/info/rfc9114

EHSAN, A., ABUHALIQA, M. A. M. E., CATAL, C., MISHRA, D. 2022. RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions. *Applied Sciences.* 12(9), 4369. https://doi.org/10.3390/app12094369

FIELDING, R. T. 2000. *Architectural Styles and the Design of Network-based Software Architectures.* Ph.D. Dissertation. University of California, Irvine.

GATHAN, B., SHAHLA, R. J., COHEN, R., BRONSTEIN, A. M. 2024. Estimating the Number of HTTP/3 Responses in QUIC Using Deep Learning. *arXiv*:2410.06140. https://doi.org/10.48550/arXiv.2410.06140

GOLMOHAMMADI, A., ZHANG, M., ARCURI, A. 2023. Testing RESTful APIs: A Survey. *ACM Transactions on Software. Engineering and Methodology.* 33(1), 27. https://doi.org/10.1145/3617175

GOWDA, P., GOWDA, A. N. 2024. Best Practices in REST API Design for Enhanced Scalability and Security. *Journal of Artificial Intelligence, Machine Learning and Data Science*. 2(1), 827–830. https://doi.org/10.1145/3617175 doi.org/10.51219/JAIMLD/priyanka-gowda/202

GUPTA, A., BARTOS, R. 2024. Improving Web Content Delivery with HTTP/3 and Non-Incremental EPS. In: *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*. Kailua-Kona, HI, USA, pp. 1–9. https://doi.org/10.1109/ICCCN61486.2024.10637641

IYENGAR, J., THOMSON, M. 2021. RFC 9000, QUIC: A UDP-Based Multiplexed and Secure Transport, May 2021. *RFC Editor*. https://www.rfc-editor.org/info/rfc9000

KASHTALIAN, A., LYSENKO, S., SAVENKO, B., SOCHOR, T., KYSIL, T. 2023. Principle and Method of Deception Systems Synthesizing for Malware and Computer Attack Detection. *Radioelectronic and Computer Systems*. 4, 112–151. https://doi.org/10.32620/REKS.2023.4.10

KIM, M., SINHA, S., ORSO, A. 2024. Adaptive REST API Testing with Reinforcement Learning. *ACM International Conference on Automated Software Engineering.* IEEE Press, pp. 446–458. https://doi.org/10.1109/ASE56229.2023.00218

KUHLEWIND, M., TRAMMELL, B. 2022. RFC 9308, Applicability of the QUIC Transport Protocol, Sep. 2022. *RFC Editor*. https://www.rfc-editor.org/info/rfc9308

MARX, R. 2021. HTTP/3 From A To Z: Core Concepts. *Smashing magazine: For Web Designers And Developers*. Aug. 9, 2021. https://www.smashingmagazine.com/2021/08/http3-core-concepts-part1/

MICHEL, F., BONAVENTURE, O. 2023. Towards SSH3: how HTTP/3 improves secure shells. *arXiv*:2312.08396. https://doi.org/10.48550/arXiv.2312.08396

OPENAPI. 2024. OpenAPI pecification v3.1.1. *OpenAPI Initiative*. https://spec.openapis.org/oas/latest.html

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

POTH, A., RRJOLLI, O., ARCURI, A. 2024. Technology adoption performance evaluation applied to testing industrial REST APIs. A*utomated Software Engineering.* 32, 5. https://doi.org/10.1007/s10515-024-00477-2

RAVURI, H. K., VEGA, M. T., VAN HOOFT, J. D., WAUTERS, T., DE TURCK, F. 2023. Adaptive Partially Reliable Delivery of Immersive Media Over QUIC-HTTP/3. *IEEE Access.* 11, 38094–38111. https://doi.org/10.1109/ACCESS.2023.3268008.

SAIF, D., MATRAWY, A. 2021. A Pure HTTP/3 Alternative to MQTT-over-QUIC in Resource-Constrained IoT. In: *2021 IEEE Conference on Standards for Communications and Networking (CSCN)*. Thessaloniki, Greece, pp. 36–39. https://doi.org/10.1109/CSCN53733.2021.9686113

WU, H., XU, L., NIU, W., NIE, C. 2022. Combinatorial Testing of RESTful APIs. In: *ACM 44th International Conference on Software Engineering (ICSE)*. Pittsburgh, USA, pp. 426–437. https://doi.org/10.1145/3510003.3510151

Acknowledgement

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.

Contact information

Jiří Balej: e-mail: jiri.balej@mendelu.cz
Andrej Juríčka: e-mail: andrej.juricka@mendelu.cz
Jiří Passinger: e-mail: jiri.passinger@mendelu.cz