TELECOMMUNICATIONS AND RADIO ENGINEERING

UDC 621.382.2(045)
DOI:10.18372/1990-5548.74.17311

[1]I. V. Zakutynskyi,
[2]I. Y. Rabodzey

MICROSERVICE COMMUNICATION FOR IOT-BASED SYSTEMS. ARCHITECTURE REVIEW AND PERFORMANCE TEST

[1]Radio Electronic Devices and Systems Department, Faculty of Air Navigation, Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine
[2]Department of Information Technology Security, Faculty of Cyber Security, Computer and Software Engineering, National Aviation University, Kyiv, Ukraine
E-mails: [1]ihor.zakutynskyi@nau.edu.ua ORCID 0000-0003-2905-3205,
[2]igor.rabodzei@gmail.com ORCID 0000-0002-2505-5249

***Abstract*—*One of the most important things in IoT system development is the right communication technologies and protocols. Communication of modern IoT systems can be divided into two main parts: device-to-cloud communication and communication between cloud microservices (application level). In this study, the authors designed a test-system environment for evaluating the performance of the existing transmitting protocols for the cloud microservices communication. The proposed environment allows emulate of IoT systems with low network latency which allows evaluating and comparing protocols performance. The authors provide tests for the most popular application-level protocols: HTTP, MQTT, AMQP, and GRPC. The performance evaluation was performed based on such metrics: throughput, concurrency, scalability, transmitting size, and init connection time. The obtained experimental results and testing environment can be used for the efficient design of microservice communication.*

Index Terms—Internet of things; communication protocols; performance evaluating; microservice communication; MQTT; HTTP; AMQP; GRPC.

I. INTRODUCTION

The number of connected IoT devices growing every day. According to the "State of IoT – Spring 2022 report", we'll have 27 billion connected IoT devices by 2025 year [1]. This is made possible by the development of a smaller and cheaper base of electronic components, as well as the development of energy-efficient data transmission technologies such as LTE, LORA, and their extensions for IoT-based systems.

The growing number of connected sensors creates real-time data processing challenges. Also, one of the biggest problems is large device fleet management.

Today there are many data transmission standards, so the problem of integration and compatibility of devices arises. The problem of integration and unification should be solved at the application level, by describing standardized interfaces. Also, today a load of IoT systems is dynamic and unpredictable, so the architecture should be available and scalable-ready. For this purpose was designed a set of special standards was by the World Wide Web Consortium (W3C) – Web of Things (WoT) [2].

WoT standards describe the interoperability of different Internet of things (IoT) platforms and application domains. At the applications level can be implemented device-agnostic abstractions for device management. Additional levels of abstraction allow us to divide the functionality into separate microservices. With this approach, we get many advantages, such as:

Scalability: Each microservice can be scaled independently of other services at high-load periods and reduced at idle periods.

Reliability: If one service fails, others continue to work.

Technologies agnostic: Each microservice can be implemented by different technologies and supported by separate teams

Testing: the functionality of the microservice can be tested without a full system test.

One of the most important things in system architecture is communication technologies and protocols. In this paper, we'll review existing techniques for microservice communication which can be applied to IoT systems development.

The object of study is the process of IoT system architecture development in order to find high-performance and cost-efficiency solutions.

The subject of study is the web microservice communication techniques and protocols which can be used to develop Cloud-based IoT systems.

The purpose of the work is to explore the performance of existing web protocols for different IoT systems scenarios.

II. Problem Statement

The Internet of Things (IoT) is rapidly becoming an important part of our lives, with more and more devices connecting to the Internet every day. One of the biggest challenges is communication between microservices. This is because microservices are often distributed across multiple devices and networks, making it difficult for them to reliably and securely communicate with each other.

The following main problems can be identified.

A. A lot of data

There are more and more connected devices every day, each of which produces huge amounts of data. However, the capacity of data storage systems is limited. Storing and managing large arrays becomes a major challenge. Hence, it has become imperative to create frameworks or mechanisms that can collect, store and process data.

B. Large traffic

Systems used for real-time monitoring generate a lot of traffic. For example, smart traffic lights, connected cars, and smart home systems. The system must withstand hundreds of millions of requests. Connecting across a range of networks, devices, and contexts increase the likelihood of data errors and communication losses, which can compromise the integrity and reliability of systems.

C. Auto-scaling

IoT-based systems often include a large number of devices that interact with each other to provide a service. The load on the system is traditionally uneven. for example, at night it is minimal, during the day it increases strongly. To support such an ecosystem we need a large number of servers. But at night they will be idle. Auto-scaling can help ensure the efficient use of resources and the system's ability to handle the current load.

D. Real-time processing

For example smart home systems, connected cars, and industrial automation systems. Real-time processing data problems can include issues such as latency, scalability, and security. Latency is the amount of time it takes for data to be processed and for the results to be returned. Scalability is the ability to handle increased data loads without compromising performance. Security is the ability to protect data from unauthorized access. Additionally, data integrity and privacy must also be taken into consideration when dealing with real-time processing data.

E. Concurrency

The concurrency problem is the synchronization of multiple devices. As the number of connected devices increases, so does the complexity of managing them. This can lead to data inconsistency, as different devices may be operating on different versions of data. Additionally, if multiple devices are trying to access the same resource, it can lead to conflicts and data corruption. This can be mitigated by using a centralized system for managing the devices, or by using distributed systems with appropriate synchronization protocols.

We should research the most popular protocols. This will help prevent errors and data loss and increase overall reliability.

III. Theoretical Basis

Each system component in microservice architecture can be distributed across multiple web servers or domains. Depending on the data type and structure should be implemented service communication methods which should be based on existing protocols and match the business logic of the system. We can split existing communication protocols into Synchronous and Asynchronous. The most popular protocols and their architectures are listed in Table I.

Table I. Application Level Protocols

	Model / Architecture	**Data type**	**Delivery guaranteed**
HTTP/ HTTPS	Sync Client-Server	T	False
HTTP2	Async Client-Server	B	False
AMQP	Async Pub/Sub	B	True
MQTT	Async Pub/Sub	B	True
gRPC	Sync/Async Client-Server	B	False

Note: Data type – textual (T) or binary (B)

A. Synchronous protocols

Protocols with synchronous logic send requests and wait for responses. In this case, the next request can be sent only after finishing the previous one. Protocols with synchronous logic are easy in

implementing and maintaining. Today, the main protocol of modern web systems is HTTP/HTTPS (Fig. 1), which is implemented by synchronous logic.

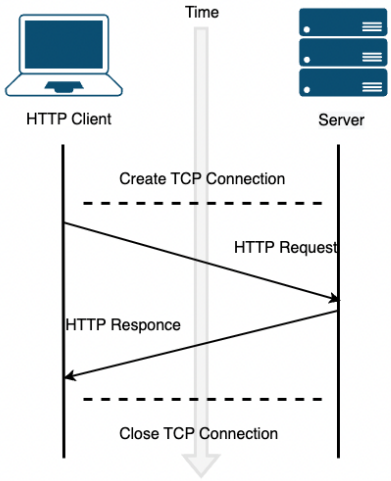

Fig. 1. HTTP protocol schema

B. *Asynchronous protocols*

In protocols with an asynchronous logic client (sender) sends a request without a response waiting.

With a non-blocking model, we can achieve higher sending performance, but the implementation and maintenance of the system are significantly complicated.

Such asynchronous protocols as AMQP (Fig. 2), and MQTT are often used in modern web and IoT systems.

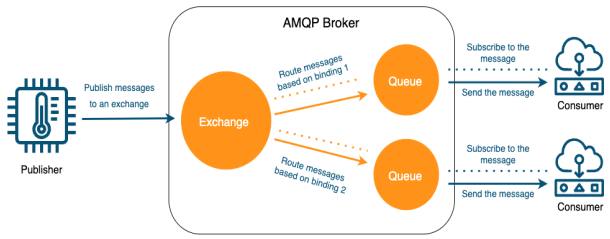

Fig. 2. AMQP protocol schema

IV. The Literature Review

The Internet of Things (IoT) has become one of the most popular technologies of the 21st century. The first stage of building such a system is the choice of architecture.

Software architecture based on microservices design and ideal for easy scaling and efficient processing of data in real-time [3].

Choosing a protocol for microservice communication between software and sensors is an equally important step in the development of systems based on IoT. Over the past decade, there has been a lot of research into microservices architectures, protocols, and types of communication.

For example, in this paper [4], the authors review and test the performance of three microservice architecture approaches for real-time data processing.

One very important problem is sending huge amounts of data to databases, which can dramatically affect the performance of the entire system-wide performance. The authors of [5] propose a solution to this problem using the AMQP protocol and conduct a comparative study with a web service, considering the communication between the client and the server.

If we would like to use core from Google, it currently supports communication between devices and the cloud with the use of MQTT and HTTP protocols.

This article [6] provides a detailed analysis of MQTT and HTTP using response rate metrics and packet volume metrics when sending the same payloads.

There is also research [7] on the gRPC buffer protocol using the example of comparing the efficiency of communication tasks between gRPC and REST.

The paper [8] presented a benchmarking of HTTP, MQTT, AMQP protocols to real-time public data on cities.

In general, most of the reviewed papers consider protocols and test them without detailed analysis and graphs, comparing no more than two protocols. In this article, we will review the most popular microservice communication protocols for IoT-based systems and perform performance testing.

V. Materials and Methods

This research is experimental and analytical approaches. It also contains a comparison of the most popular data protocols used in IoT systems. Performance testing was performed for each protocol to determine in which case which protocol is better suited.

Performance tests are performed on device emulators, which allows you to effectively compare the capabilities of technologies.

Since the goal is to evaluate the quality of the protocols, they will be in the same Amazon network to reduce the impact of the network on the experiment.

For testing, JavaScript scripts were written using the Node.JS server environment.

VI. Experiments

In this section, we introduce the test scenario for our experiments. For performance testing, we have developed custom software that simulates

transmission performance measurements using HTTP, AMQP, MQTT, and GRPC clients. In our experiment, the tests performed on AWS EC2 instances are detailed in Table II.

TABLE II. TEST SERVER HARDWARE DETAILS

Instance type	**t2.large**
CPU	3.3 GHz Intel Xeon** Processor
vCPU	2
Mem (GiB)	8
Storage	EBS-Only
Network Performance	Moderate

The HTTP (Fig. 3) and GRPC protocols (Fig. 4) have a Client / Server architecture. The client initiates a request and waits for a response from the server. For testing, we will need to create two EC2 instances, one for the server and one for the client.

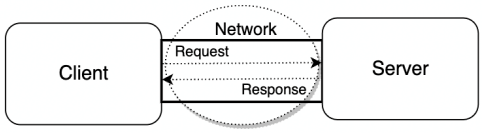

Fig. 3. HTTP Client-Server architecture

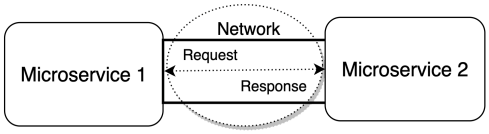

Fig. 4. GRPC Client-Server architecture

AMQP and MQTT use a Publisher-Subscriber architecture (Fig. 5). Messaging consists of three parts - the publisher sends the message to the broker, the broker creates a queue and topic, and the subscriber subscribes to the topic and receives messages from the broker. In our performance testing for these protocols, we highlight two options for testing Publish and Delivery. Using this architecture, we will need to create three EC2 instances: a publisher, a subscriber, and a message broker.

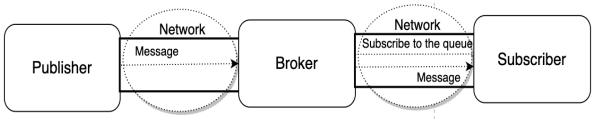

Fig. 5. AMQP Publisher-Subscriber architecture

To reduce the impact of network latency as much as possible, during testing, all instances work in the same network. The most popular issues are request processing speed, response time, and poor scalability.

We have selected the main metrics for evaluating protocol performance (Test 1 – Test 5). Testing results are listed in Figs 6 – 10.

VII. RESULTS

Test 1. Init connection time. Is the time taken to complete the initial TCP connection and SSL negotiation (where applicable).

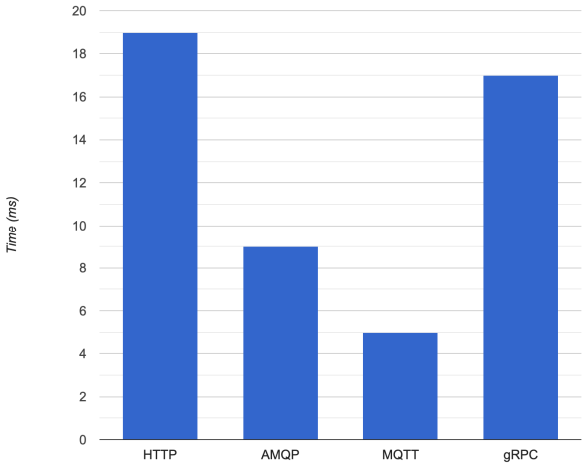

Fig. 6. Init connection time

The results (Fig. 6) of this test are very similar for all protocols. But it should be noted that for the Client-server architecture (HTTP), a new connection will be established for each request, which will negatively affect to general performance. Instead, for other protocols, this action will be performed only once.

Test 2. Request per second (throughput) is a measure of how many requests a server can handle in a second.

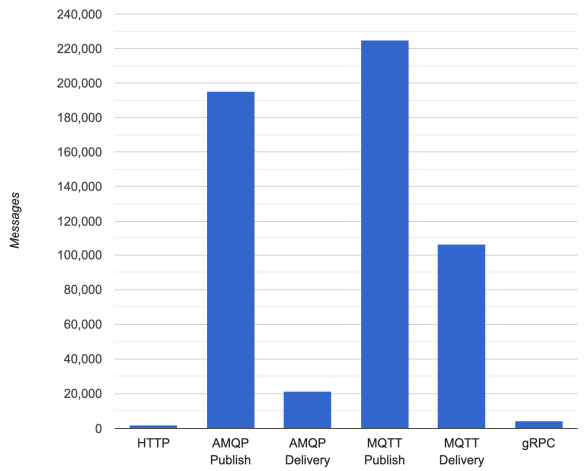

Fig. 7. Messages / Requests per second

The results of this test (Fig. 7) indicate that MQTT/AMQP has a significant advantage over HTTP/gRPC. Although MQTT/AMQP metrics are scaled from two Publish/Delivery tests, these protocols send more messages per unit of time.

The main reason is that the Client-Server paradigm establishes a new connection for each

message transmission, while the Publisher-Subscriber paradigm can transmit any number of messages in a single connection.

Test 3. Time to send 1M messages. An important metric for evaluating a protocol is how much it can handle over a million requests.

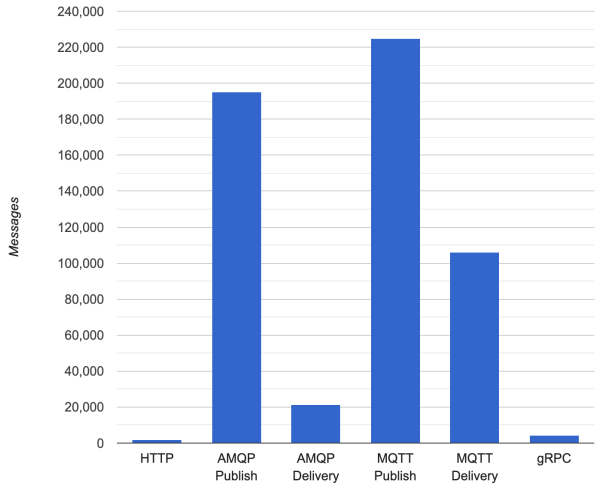

Fig. 8. Time to send 1M messages/requests

This test result correlates with Test 2 (Fig. 7) and shows a significant advantage of MQTT/AMQP over HTTP/gRPC.

Test 4. Throughput: Megabytes per second. Throughput is the data transfer rate and is commonly measured in bytes per second (b/s).

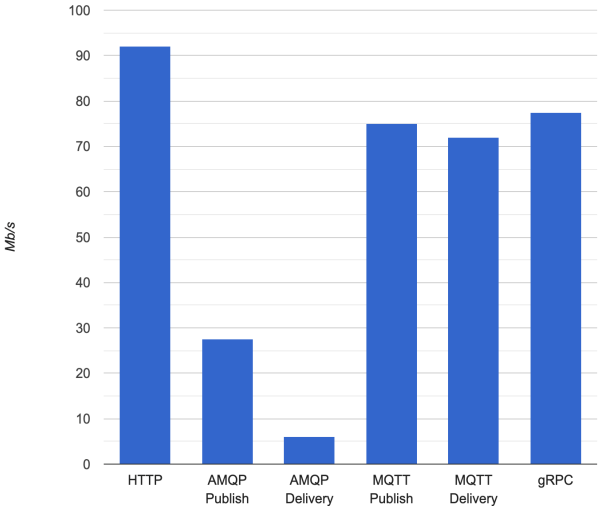

Fig. 9. Throughput Mb/s

Test 5. Data transferring performance. This test indicates how many protocols need to transmit such data size. The test used files from 0.1 to 500 Megabytes. Also, for some protocols, there is a limit to the maximum message size. Therefore, messages that exceeded this limit were transmitted in several batches.

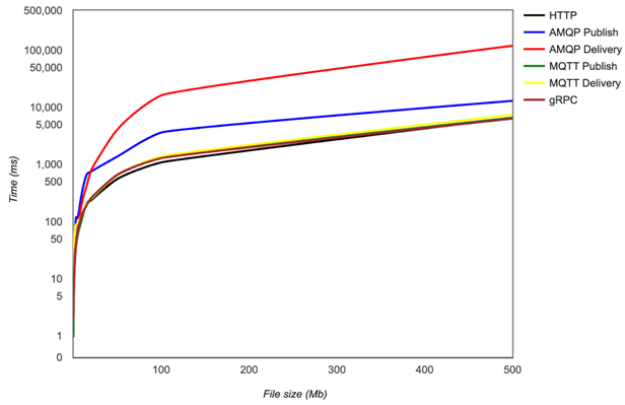

Fig. 10. Data transferring performance

In this test, HTTP performed the best result, therefore, it is best suited for transferring large files.

VIII. CONCLUSIONS

In this study, we developed an efficient test environment for analyzing performance application-level protocols for IoT-based systems. This environment allows evaluating of the protocol performance at the system design stage. Based on the above environment performed tests for the main protocols for microservice communication: HTTP, HTTP/2, MQTT, AMQP, and gRPC.

Based on the conducted results, we can evaluate the difference between asynchronous and synchronous protocols for different IoT systems scenarios, and their pros and cons.

The obtained results can be used to design stable and cost-efficient IoT system architecture.

REFERENCES

[1] Mohammad Nasan. (2022, May). State of IoT – May 2022. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices.

[2] Wikipedia contributors. (2023, February 2). World Wide Web Consortium. Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/World_Wide_Web_Consortium.

[3] U. Zdun, E. Navarro, and F. Leymann, "Ensuring and Assessing Architecture Conformance to Microservice Decomposition Patterns," In Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds) Service-Oriented Computing. ICSOC 2017. *Lecture Notes in Computer Science()*, vol. 10601, Springer, Cham. https://doi.org/10.1007/978-3-319-69035-3_29.

[4] Guadalupe Ortiz, Juan Boubeta-Puig, Javier Criado, David Corral-Plaza, Alfonso Garcia-de-Prado, Inmaculada Medina-Bulo, and Luis Iribarne, "A microservice architecture for real-time IoT data processing: A reusable Web of things approach for smart ports," *Computer Standards & Interfaces*, vol.

81, 2022, 103604, ISSN 0920-5489, https://doi.org/10.1016/j.csi.2021.103604.

[5] Joel Fernandes,& Ivo Lopes, & Joel Rodrigues, & Sana Ullah, "Performance evaluation of RESTful web services and AMQP protocol," *International Conference on Ubiquitous and Future Networks*, *ICUFN*, 2013. https://doi.org/10.1109/ICUFN.2013.6614932

[6] Charlie Wang, *HTTP vs MQTT: A tale of two IoT protocols*, 2018, November. [Online]. Available: https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols

[7] Marek Bolanowski, & Kamil Żak, & Andrzej Paszkiewicz, & Maria Ganzha, & Marcin Paprzycki, & Piotr Sowiński, & Ignacio Lacalle Úbeda, & Carlos Palau, *Eficiency of REST and gRPC realizing communication tasks in microservice-based ecosystems*, 2022. 10.48550/arXiv.2208.00682. https://doi.org/10.3233/FAIA220242

[8] Cavide Gemirter, & Sebnem Baydere, *A Comparative Evaluation of AMQP, MQTT and HTTP Protocols Using Real-Time Public Smart City Data*, 2021. https://doi.org/10.1109/UBMK52708.2021.9559032.

Received November 18, 2022

Ihor Zakutynskyi. ORCID 0000-0003-2905-3205. PhD student.
Radio Electronic Devices and Systems Department, Faculty of Air-navigation, Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine.
Education: National Aviation University, Kyiv, Ukraine, (current time).
Research area: neural networks, software development, automation systems.
Publications: 6.
E-mail: ihor.zakutynskyi@nau.edu.ua

Ihor Rabodzei. ORCID 0000-0002-2505-5249. Master's.
Department of Information Technology Security, Faculty of Cyber Security, Computer and Software Engineering, National Aviation University, Kyiv, Ukraine.
Education: National Aviation University, Kyiv, Ukraine, (current time)..
Research direction: neural networks, software development, automation systems.
Publications: 1.
E-mail: igor.rabodzei@gmail.com

І. В. Закутинський, І. Є. Рабодзей. Мікросервісна комунікація для IoT систем. Огляд архітектур та порівняння продуктивності

Важливим етапом у розробці сучасних IoT систем є вибір комунікаційних технології та протоколів. Комунікацію IoT системи умовно можна розділити на дві частини: зв'язок між пристроями та хмарними сервісами та зв'язок між хмарними мікросервісами (програмний рівень). У цій роботі розроблено середовище тестування для оцінювання продуктивності протоколів програмного рівня. Пропоноване середовище дозволяє емулювати IoT систему з низькою затримкою мережі, що дозволяє ефективно оцінити та порівняти продуктивність та архітектуру протоколів, а також доцільність їх використання у тих чи інших ситуаціях. Проведено тести для найпопулярніших протоколів програмного рівня: HTTP, MQTT, AMQP і GRPC. Оцінювання продуктивності проводилося на основі таких показників як: пропускна здатність, паралельність, масштабованість, та час початкового з'єднання. Отримані експериментальні результати та середовище тестування можна використовувати при проектуванні хмарної архітектури сучасних IoT систем.

Ключові слова: інтернет речей; протоколи комунікації; оцінка продуктивності; мікросервісна комунікація; MQTT; HTTP; AMQP; GRPC.

Закутинський Ігор Володимирович. ORCID 0000-0003-2905-3205. Аспірант.
Кафедра електроніки, робототехніки і технологій моніторингу та Інтернету речей, Факультет аеронавігації, електроніки та телекомунікацій, Національний авіаційний університет, Київ, Україна.
Освіта: Національний авіаційний університет, Київ, Україна, (2019).
Напрям наукової діяльності: нейронні мережі, розробка програмного забезпечення, системи автоматизації.
Кількість публікацій: 6.
E-mail: ihor.zakutynskyi@nau.edu.ua

Рабодзей Ігор Євгенович. ORCID 0000-0002-2505-5249. Магістр.
Кафедра безпеки інформаційних технологій, Факультет кібербезпеки, комп'ютерної та програмної інженерії, Національний авіаційний університет, Київ, Україна.
Освіта: Національний авіаційний університет, Київ, Україна (2019).
Напрямок досліджень: нейронні мережі, розробка програмного забезпечення, системи автоматизації.
Кількість публікацій: 1.
E-mail: igor.rabodzei@gmail.com