International Journal of Advanced Smart Convergence Vol.12 No.1 9-17 (2023)
http://dx.doi.org/10.7236/IJASC.2023.12.1.9

IJASC 23-1-2

Performance Comparison of HTTP, HTTPS, and MQTT for IoT Applications

Sukjun Hong[1], Jinkyu Kang[2], and Soonchul Kwon[3*]

[1] Master, Department of Smart System, Kwangwoon University, South Korea
[2] Chief Executive Officer, TSP-XR, South Korea
[3] Associate professor, Graduate School of Smart Convergence, Kwangwoon University, South Korea*

000jun2@kw.ac.kr, kang.jinkyu@tsp-xr.com, ksc0226@kw.ac.kr

Abstract

Recently, IoT technology has been widely used in many industries. Also research on integrating IoT technology with IoT sensors is actively underway. One of the important challenges in IoT is to support low-latency communication. With the development of communication networks and protocols, a variety of protocols are being used, and their performance is improving. In this paper, we compare the performance and analyze the characteristics of some of the major communication protocols in IoT application, namely MQTT, HTTP, and HTTPS. IoT sensors acquired data by connecting an Arduino equipped with ESP8266 and a temperature and humidity sensor (DHT11). The server measured the performance by building servers for each protocol using AWS EC2. We analyzed the packets transmitted between the Arduino and the server during the data transmission. We measured the amount of data and transfer time. The measurement results showed that MQTT had the lowest data transmission time and data amount among the three protocols.

Keywords*: Communication Protocol, HTTP, HTTPS, IoT, MQTT*

1. Introduction

The Internet of Things (IoT) has become increasingly prevalent across various industries, including healthcare, smart homes, transportation, and wearable devices such as fitness bands and watches, all of which are connected to the internet [1]. IoT devices have become particularly useful in environments where human access is difficult or automation is required [2]. However, as the use of IoT technology continues to expand, it faces significant challenges in providing long battery life and resolving communication latency issues [3,4].

To address these challenges, IoT systems need to support low power and low latency communication. Several wireless technologies support low power communication, including RFID (Radio-Frequency Identification), WLAN (Wireless Local Area Network), WPAN (Wireless Personal Area Network), and WMAN (Wireless Metropolitan Area Network) [5]. Additionally, efficient communication with low latency communication protocols is required to reduce server overload and latency. Many protocols have been

Manuscript Received: December. 6, 2022 / Revised: December. 10, 2022 / Accepted: December. 12, 2022
Corresponding Author: kcs0226@kw.ac.kr
Tel: +82-2-940-8637 , Fax: +82-50-4174-3258
Associate professor Graduate School of Smart Convergence, Kwangwoon University, South Korea

developed, improved, and performance-enhanced to meet these requirements [6].

In IoT applications, continuous data communication occurs between IoT sensors and edge or cloud servers. However, not all protocols can meet all requirements, such as energy efficiency, security, and stability [7]. Therefore, it is essential to choose an efficient communication protocol to reduce server overload and latency [8]. Representative IoT communication protocols include HTTP (HyperText Transfer Protocol), HTTPS (HyperText Transfer Protocol over Secure Socket Layer), and MQTT (Message Queue Telemetry Transport).

This paper aims to configure HTTP, HTTPS, and MQTT servers and compare and analyze their performance. The paper structure includes characterization of HTTP, HTTPS and MQTT protocols, followed by experiments where temperature and humidity data are read using DHT11 sensors and transmitted to each server. Real-time measurements of the amount of data and transfer time were taken using the Wireshark program. By analyzing the performance of each protocol, this study aims to provide information on selecting an appropriate IoT communication protocol for efficient and effective communication.

2. Background Theory

2.1 HTTP

HTTP is a hypertext transfer protocol, which serves as a fundamental protocol interface for moving extensive data quickly, easily, and reliably from a server to user devices such as browsers in various applications [9]. It is a text-based protocol that allows for the transmission of a large number of small packets and has no size limit on the header section or message [10]. The HTTP protocol operates over TCP/IP, enabling stable communication that ensures data is transmitted without being corrupted between devices. Data is transmitted based on IP addresses and URLs. The connection is dynamic and terminated every time access is made [11]. HTTP provides methods such as GET, POST, PUT, and DELETE for data communication. However, HTTP is not suitable for IoT systems with limited resources and consumes significant amounts of energy and time. Figure 1 illustrates the HTTP communication architecture.

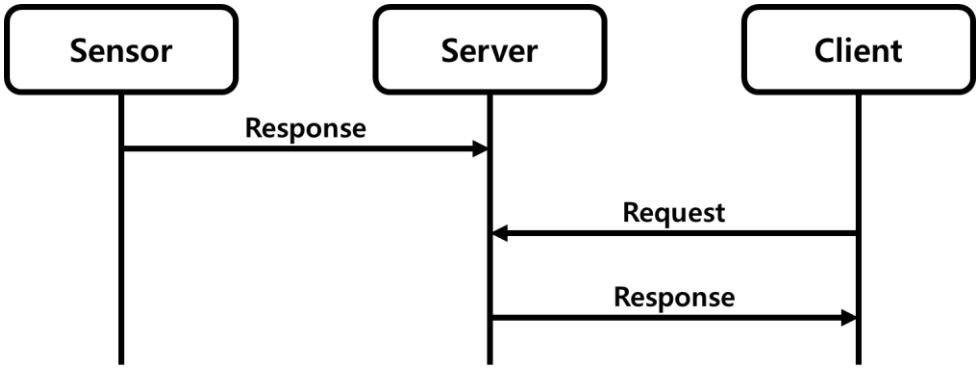


Figure 1. HTTP Communication Architecture

2.2 HTTPS

HTTPS is a protocol that runs HTTP over SSL (Secure Socket Layer) and TLS (Transport Layer Security) protocols, allowing for the establishment of an encrypted bidirectional communication channel to protect network communications. Data communication is slower than HTTP because encryption/decryption process is included. By presenting a certificate signed by a trusted certification authority to the client and server, identity authentication is guaranteed [12]. Table 1 shows a comparison of HTTP and HTTPS.

Table 1. Comparison of HTTP and HTTPS

	HTTP	HTTPS
Security	X	O
Encryption	X	O
Certificate	X	O
Port	80	443

2.3 MQTT

MQTT is a protocol used for communication in IoT application. It is a lightweight messaging protocol that efficiently uses network bandwidth with a fixed 2-byte header [13]. MQTT uses a publish-subscribe communication model [14]. It is divided into a broker server, publishers, and subscribers. The broker server operates by pushing data to MQTT subscribers. Publishers share their specific information by publishing it, while subscribers receive data from publishers by registering their interests. The broker ensures that the data from publishers is delivered to subscribers. Subscribers can connect to publishers through the broker, which regulates subscriptions [15]. MQTT is designed to simplify implementation on the client side by concentrating all data processing on the broker. However, brokers do not inherently provide security during data communication, so a separate security system must be established [14]. Figure 2 shows the MQTT communication architecture.

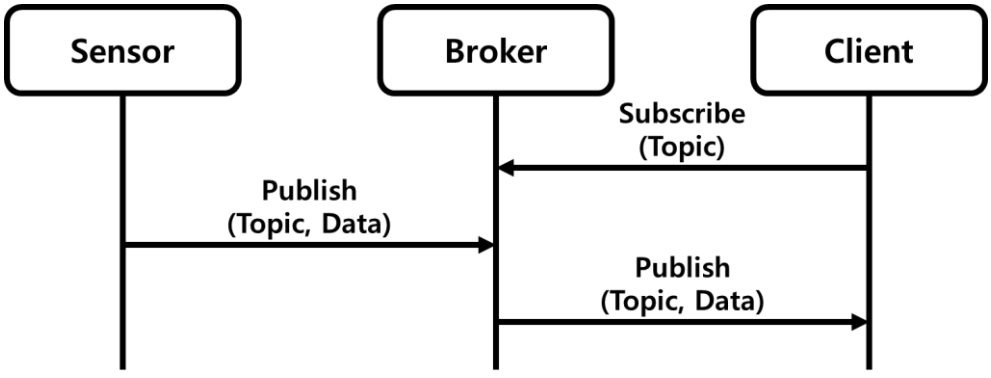


Figure 2. MQTT Communication Architecture

3. Experiments Environment

In this paper, DHT11 is used as the IoT sensor. DHT11 is a sensor that can measure temperature and humidity. It can acquire real-time environmental data. This sensor obtains sensor data by connecting to the Arduino board that includes the ESP8266 module. Arduino connects to the Internet using Wi-Fi, and the collected sensor data is transmitted through communication with the server. The Wi-Fi communication was established by connecting to the 2.4 GHz band. The data transmitted from the Arduino includes the current temperature and humidity values. Transmitted data consists of English letters, numbers, and special characters. In addition, there are a total of 39 characters including spaces, and the amount of data is 39 bytes. Figure 3 shows a circuit diagram where Arduino with ESP8266 is connected to DHT11.

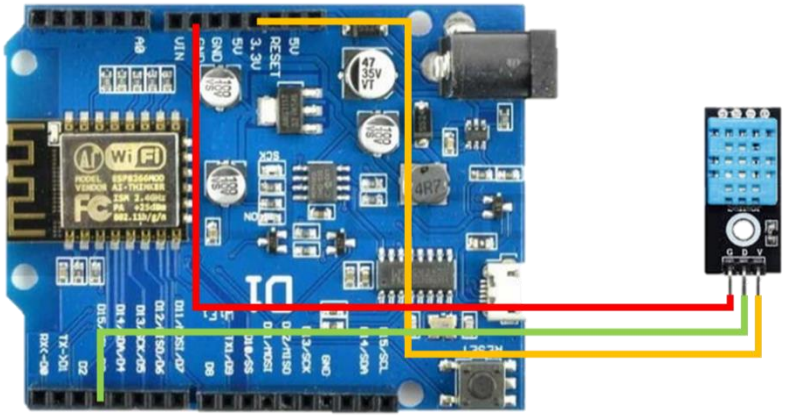


Figure 3. Circuit Diagram

In this paper, the server was built using EC2 of AWS. The OS of the server used Ubuntu 20.04 version. For the server, MQTT-Broker, MQTT-Subscriber, HTTP, and HTTPS servers were built respectively. In MQTT, broker server and subscriber are configured separately. The subscriber connects to the broker server, subscribes to a topic, and acquires data generated from the topic. HTTP and HTTPS built a web server using Apache2. After obtaining data using the Get method, it is stored in the DB. For MQTT, HTTP, and HTTPS, one client connects. Table 2 is the AWS server environment used in the experiment.

Table 2. Server Environment

AWS (EC2)	Type	T2 micro
	CPU	vCPUs, 2.5 GHz
	Ram	1G
	Network	10Gbps
	OS	Ubuntu 20.04
	Web Server	Apache 2.4.41
	DB	SQLite 3.31.1
	MQTT	Mosquitto 1.6.9

The analysis of the data amount and transfer time for a single transmission was conducted using the Wireshark program. Wireshark is a program that captures and analyzes network packets, enabling real-time monitoring of data transmission processes. In this paper, packet analysis was performed using Wireshark to measure the data amount and transfer time between the Arduino and the server during data transmission. Sensor data is automatically transmitted every 5 seconds, and the transfer time was measured from the start of data transmission to its completion. Figure 4 shows the data transmission flow for MQTT, HTTP, and HTTPS. Figure 5 is a screenshot of data communication using Wireshark. We analyzed the data transmitted in real time using MQTT from arduino to AWS.

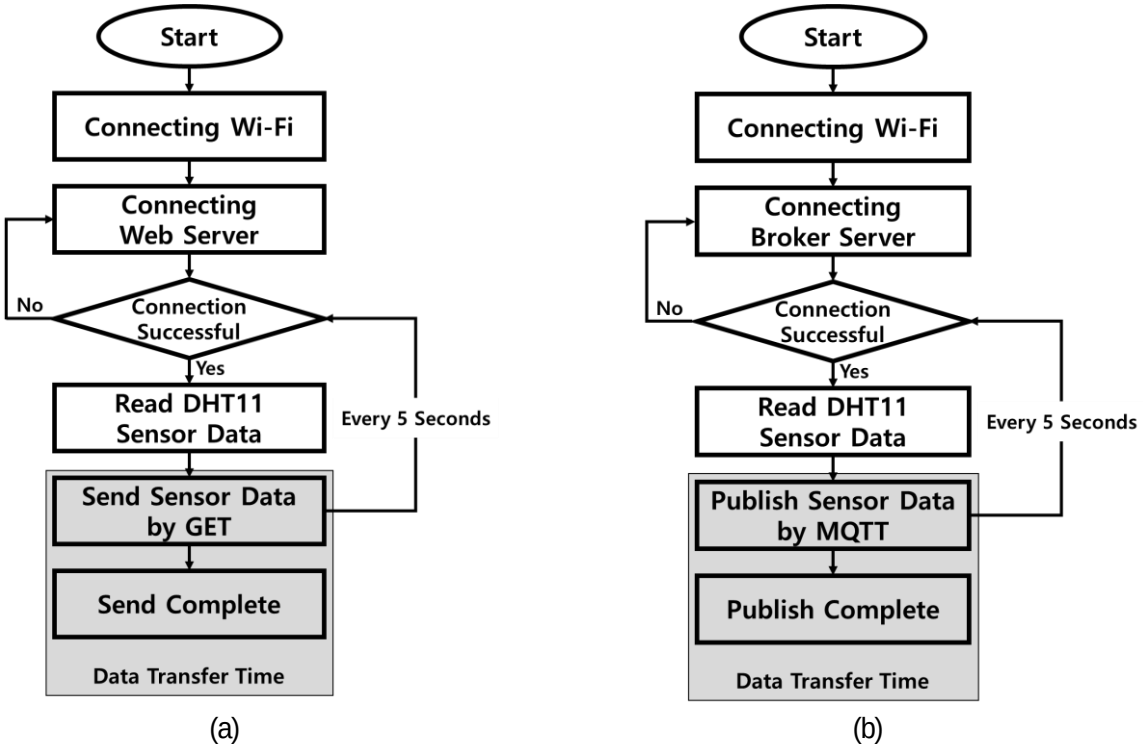


Figure 4. Data Transfer Flow Chart

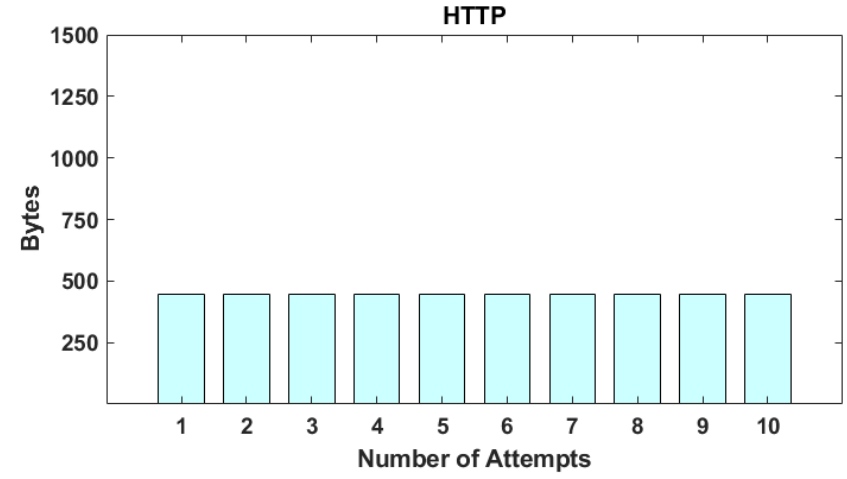
(a) HTTP, HTTPS (b) MQTT

No.	Time	Source	Destination	Protocol	Length	Info
108	160.161869	192.168.137.154	3.34.94.204	MQTT	81	Connect Command
110	160.179086	3.34.94.204	192.168.137.154	MQTT	58	Connect Ack
111	160.186256	192.168.137.154	3.34.94.204	MQTT	77	Publish Message [outTopic]
118	160.289867	192.168.137.154	3.34.94.204	MQTT	127	Subscribe Request (id=2) [inTopic], Publish Message [Sensor/Humi_Temp]
120	160.317630	3.34.94.204	192.168.137.154	MQTT	59	Subscribe Ack (id=2)
142	165.217495	192.168.137.154	3.34.94.204	MQTT	113	Publish Message [Sensor/Humi_Temp]
144	170.245280	192.168.137.154	3.34.94.204	MQTT	113	Publish Message [Sensor/Humi_Temp]
146	175.272612	192.168.137.154	3.34.94.204	MQTT	113	Publish Message [Sensor/Humi_Temp]
148	180.300161	192.168.137.154	3.34.94.204	MQTT	56	Ping Request
150	180.381772	3.34.94.204	192.168.137.154	MQTT	56	Ping Response
151	180.485144	192.168.137.154	3.34.94.204	MQTT	113	Publish Message [Sensor/Humi_Temp]
156	185.330461	192.168.137.154	3.34.94.204	MQTT	113	Publish Message [Sensor/Humi_Temp]
158	190.357874	192.168.137.154	3.34.94.204	MQTT	113	Publish Message [Sensor/Humi_Temp]
160	195.386739	192.168.137.154	3.34.94.204	MQTT	113	Publish Message [Sensor/Humi_Temp]
162	200.414052	192.168.137.154	3.34.94.204	MQTT	56	Ping Request
164	200.601277	3.34.94.204	192.168.137.154	MQTT	56	Ping Response
165	200.754813	192.168.137.154	3.34.94.204	MQTT	113	Publish Message [Sensor/Humi_Temp]

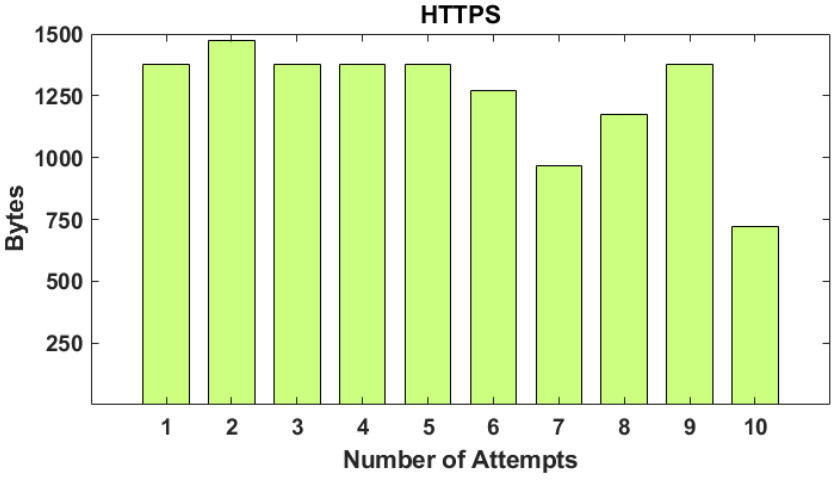
Figure 5. Data Analysis Using Wireshark

4. Experiments Result

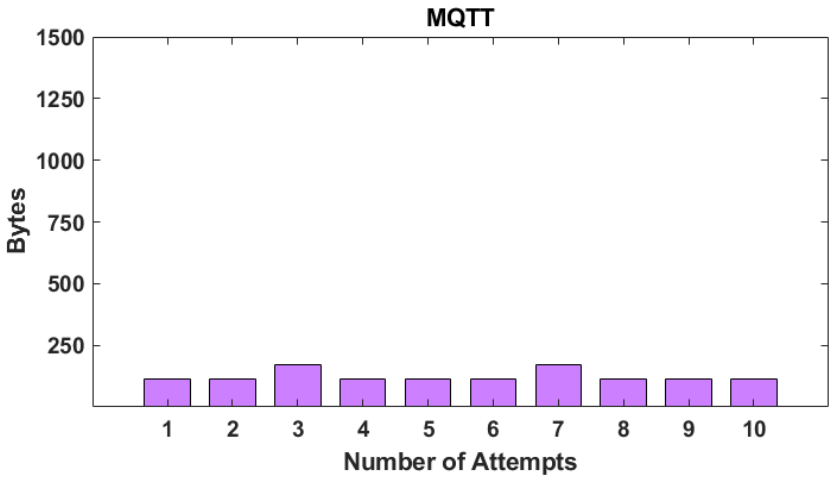
We monitored and analyzed 10 data transmissions in real-time using Wireshark, which utilized HTTP, HTTPS, and MQTT protocols. Figure 6 shows the result of measuring the amount of data transferred. (a) shows the results using the HTTP communication protocol, which showed a higher amount of data compared to MQTT. However, the data amount remained stable at 448 Bytes for all transmissions, enabling very stable communication. (b) represents the results using the HTTPS communication protocol. The average data amount was approximately 1407.4 Bytes, and the standard deviation was around 79, showing relatively unstable results. HTTPS encrypts the data before transmitting it, which results in a relatively larger amount of data. (c) represents the results using the MQTT communication protocol. Except for the 3rd and 7th transmissions, 113 Bytes of data were transmitted. Pings are sent regularly for the 3rd and 7th transmissions to check the connection with the broker server. The MQTT average data amount was 124.2 Bytes with a standard deviation of 23.611, indicating relatively stable results.



(a)



(b)



(c)

Figure 6. Result of Measuring the Amount of Data Transferred

(a) HTTP (b) HTTPS (c) MQTT

Figure 7 shows the results of 10 measurements of data transfer time from the Arduino to MQTT, HTTP, and HTTPS. MQTT showed a data transfer time of approximately 0.001 seconds, except for the 3rd and 7th measurements, as MQTT protocol sends pings to the broker server to confirm the connection. The average data transfer time for MQTT was approximately 0.0197 seconds with a standard deviation of approximately 0.042.

HTTP showed a slower data transfer time of approximately 0.169 seconds compared to MQTT, but had a stable result with a standard deviation of approximately 0.012. However, HTTPS showed the most unstable result with an average data transfer time of approximately 2.211 seconds and a standard deviation of approximately 2.515, which is the longest among the three protocols. This is because the HTTPS protocol adds encryption and decryption processes during the data transfer, which takes more time.

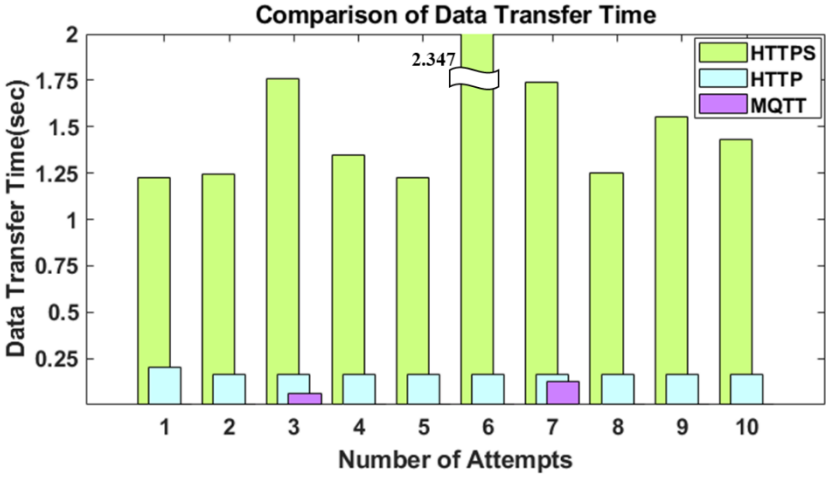


Figure 7. Comparison of Data Transfer Time

5. Conclusion

The IoT application is expected to be widely utilized in various industries such as healthcare, smart home, security, etc. To fully utilize IoT, the development of low-latency systems is necessary. In this paper, we measured and compared communication data amount and transfer time of MQTT, HTTP, and HTTPS protocols used for IoT sensors and communication. The experimental results show that MQTT has the lowest transfer time and data amount. HTTP has more data amount compared to MQTT but shows the most stable results with consistent transfer. On the other hand, HTTPS shows high transfer time and data amount.

MQTT and HTTP have almost no basic security system. It is essential to establish a separate security system in environments where security is critical for these protocols. However, HTTPS has a basic security system unlike the protocols. MQTT is suitable for environments requiring low-latency and stable communication, while HTTP is suitable for environments that require stability over speed. Although HTTPS is not suitable for real-time communication, it is used to protect user's personal information or organization's security. Through this study, analysis of IoT communication protocols and the development of low-latency IoT application can be performed to optimize performance. The selection of protocols suitable for the IoT system environment can contribute to the efficient construction of the system.

Acknowledgement

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.RS-2022-00156041, Development of object/drawing technology and web application for augmented reality interior service). And the present research has been conducted by the excellent researcher support project of Kwangwoon University in 2022

References

[1] B. Mishra and A. Kertesz, “The Use of MQTT in M2M and IoT Systems: A Survey,” IEEE Access, .Vol. 8, pp. 201071–201086, 2020.
DOI: https://doi.org/10.1109/ACCESS.2020.3035849

[2] I. Heđi, I. Špeh and A. Šarabok “IoT network protocols comparison for the purpose of IoT constrained networks”, 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, May-2017.
DOI: https://doi.org/10.23919/MIPRO.2017.7973477

[3] A. Javed, H. Larijani, and A. Wixted, “Improving Energy Consumption of a Commercial Building with IoT and Machine Learning,” IT Professional, Vol. 20, No. 5. pp. 30–38, Sep-2018.
DOI: https://doi.org/10.1109/MITP.2018.053891335

[4] N. Naik, “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP,” 2017 IEEE International Systems Engineering Symposium (ISSE). IEEE, Oct-2017.
DOI: https://doi.org/10.1109/SysEng.2017.8088251

[5] D. Glaroudis, A. Iossifides, and P. Chatzimisios, “Survey, comparison and research challenges of IoT application protocols for smart farming,” Computer Networks, Vol. 168. Elsevier BV, p. 107037, Feb-2020.
DOI: https://doi.org/10.1016/j.comnet.2019.107037

[6] N. Q. Uy and V. H. Nam, “A comparison of AMQP and MQTT protocols for Internet of Things,” 2019 6th NAFOSTED Conference on Information and Computer Science (NICS). IEEE, Dec-2019.
DOI: https://doi.org/10.1109/NICS48868.2019.9023812

[7] R. A. Atmoko, R. Riantini, and M. K. Hasin, “IoT real time data acquisition using MQTT protocol,” Journal of Physics: Conference Series, Vol. 853. IOP Publishing, p. 012003, May-2017.
DOI: https://doi.org/10.1088/1742-6596/853/1/012003

[8] J. Sidna, B. Amine, N. Abdallah, and H. El Alami, "Analysis and evaluation of communication Protocols for IoT Applications," Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications. ACM, 23-Sep-2020.
DOI: https://doi.org/10.1145/3419604.3419754

[9] C. B. Gemirter, C. Senturca, and S. Baydere, "A Comparative Evaluation of AMQP, MQTT and HTTP Protocols Using Real-Time Public Smart City Data," 2021 6th International Conference on Computer Science and Engineering (UBMK). IEEE, 15-Sep-2021.
DOI: https://doi.org/10.1109/UBMK52708.2021.9559032

[10] T. Yokotani and Y. Sasaki, "Comparison with HTTP and MQTT on required network resources for IoT," 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC). IEEE, Sep-2016.
DOI: https://doi.org/10.1109/ICCEREC.2016.7814989

[11] S. Calzavara, R. Focardi, M. Nemec, A. Rabitti, and M. Squarcina, "Postcards from the Post-HTTP World: Amplification of HTTPS Vulnerabilities in the Web Ecosystem," 2019 IEEE Symposium on Security and Privacy (SP). IEEE, May-2019.
DOI: https://doi.org/10.1109/SP.2019.00053

[12] R. K. Kodali and S. Soratkal, "MQTT based home automation system using ESP8266," 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). IEEE, Dec-2016.
DOI: https://doi.org/10.1109/R10-HTC.2016.7906845

[13] D. Dinculeană and X. Cheng, "Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices," Applied Sciences, Vol. 9, No. 5. MDPI AG, p. 848, 27-Feb-2019.
DOI: https://doi.org/10.3390/app9050848

[14] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks," 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08). IEEE, Jan-2008.
DOI: https://doi.org/10.1109/COMSWA.2008.4554519