Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

A Comprehensive Review of Internet of Things Platforms and Protocol: A Comparative Analysis

Mandeep kaur

Assistant Professor, Department of ECE, Punjabi University Patiala, India, ermandeep0@gmail.com

This study provides a thorough comparison of the protocols MQTT, XMPP, and CoAP within the framework of Internet of Things communication. Performance parameters including protocol efficiency, packet formation time, and packet transmission time are monitored and examined via an experimental assessment. A laptop from ASUS and an Intel Galileo Gen 2 board are used in the experiment, and protocols are installed on both to provide comparative data. As a result of its broad usage and optimization, the results show that MQTT performs better in terms of packet formation and transmission times. On the other hand, because of its XML stanza format, XMPP performs worse than CoAP, which is rather impressive. The significance of optimization and standardization is emphasized by these results, which provide insightful guidance for protocol selection in Internet of Things applications.

Keywords: CoAP, MQTT, XMPP, IoT communication, Protocol comparison, Performance evaluation, Experimental analysis.

1. Introduction

With proper communication systems, connectivity, and virtual assistance, life has become easier and smarter in the digital age. Innovations have effectively changed every field, including agriculture, healthcare, education, safety and security, and household appliances. Today's innovations are much more automated than those of the previous ten years. And it goes without saying that the need for connection has led to the internet becoming a necessary component of every aspect of human existence, including the frameworks that are included into the equipment we use for business and pleasure.

As a result, IoT is quietly but quickly encroaching on human existence in an effort to improve consumers' quality of life. This is the reason why integrating IoT-enabled gadgets into our daily lives is becoming more and more popular. IoT-enabled devices let you do several tasks at once by providing necessary monitoring for predetermined criteria, which removes the requirement for your physical presence and frees up your time to complete multiple tasks. A modernized house is sometimes referred to as an intelligent home. A Remote House

Automated System (RHAS) employing Internet of Things (IoT) is a structure that employs PCs or PDAs to regulate vital home limitations and features generally over web from anywhere in the globe. The goal is to preserve both human need and electrical power. Through online affiliation, the home-bot structure differs from other systems in that it enables the user to operate the system from any location in the globe. In this work, we provide a remote house automated system (RAHS) that makes use of Node MCU and cloud coordination and remote communication to provide the user with control over different lights, fans, and household appliances as well as cloud data management.

Usually, the framework will alter based on data from the sensors. This framework must be extendable and minimal effort in order to allow for the control of a variety of devices. Given the comfort it gives, particularly in confidential homes, it appears sense that 21st-century homes would turn out to be more independent and automated. A home-bot framework is a theory that gives clients command over factor type electric gadgets. Countless laid out home-bot systems depend on wired communication. This doesn't take care of an issue except if the system is executed during the genuine underlying overhaul and is planned appropriately ahead of time. In any scenario, the usage cost increases for already in place structures. Curiously, distant structures may provide computerization frameworks with astounding assistance. With the development of remote developments, such as Wi-Fi and cloud systems in the continual past, inaccessible structures are used on a daily basis everywhere.

1.1. MQTT

It's the greatest IOT protocol for being lightweight and bandwidth-efficient.The publish/subscribe model is supported by MQTT.It is designed to communicate reliably across erratic channels.Originally, using other protocols to communicate films and photos was quite difficult. However, MQTT is a bi-directional and efficient protocol that does not care what you send over the internet, so you can send pictures and videos using it.Sending data to and from devices to the cloud is possible using MQTT. It is essentially based on push communication and is used to link over 10 million devices over the same channel.It is suitable for devices with limited resources, which means you may still establish message transmission using MQTT protocols if your computer power and memory are limited. The broker is the core of protocols for MQTT. Dispatching messages between senders and recipients is done via brokers. Every customer sends a message to the broker, including the subject matter. Subjects are crucial elements of MQTT. To separate communications for each related consumer, the merchant uses points. Each client that requires communications subscribes to a certain theme, and the merchant sends all messages together with the coordinating point. Thus, it is not necessary for the clients to know one another. They only communicate the idea. Extremely flexible arrangements without restrictions between the information providers and the information consumers are made possible by this architecture. Topic with at least one level, with forward cuts isolating each point the longest topic is 65000; however themes are often rather short. It has a dynamic quality and is lightweight. Model my front room, bottom floor, and house. Furthermore, subjects may be used as a trump card.

The following are Topic's applications:

- The MQTT protocol is used by the Facebook-lite messenger.

- AWS also makes partial use of the MQTT protocol.
- MQTT is the primary protocol used by Microsoft Azure IoT Hub for
- MQTT protocols are used by most IoT-enabled platforms and devices

The basis of MQTT is TCP.It does support heartbeat mechanism, security at the transport level, and persistent TCP connections. When a device connects to the cloud, MQTT employs a TCP connection. It also uses a heartbeat mechanism to continue operating even if the TCP connection is lost.

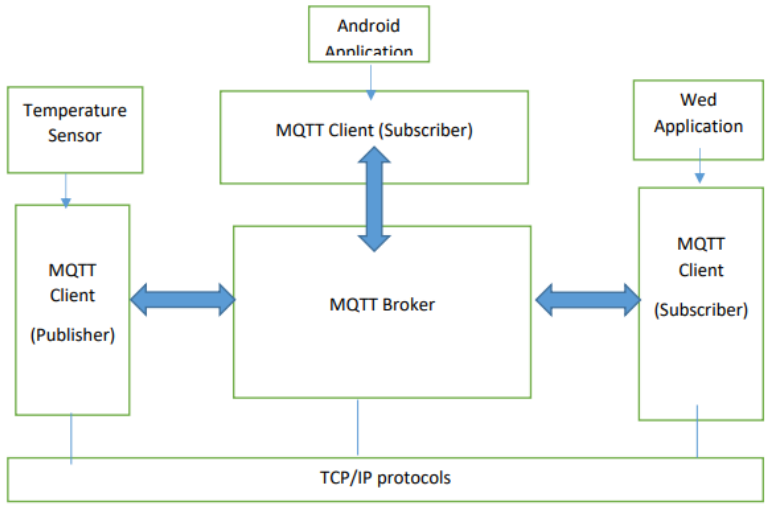

Figure 1: Broker-based MQTT model

1.2. CoAP

For restricted devices, the constrained application protocol is used.Its foundation is the request-response model. It is mostly used for residential usage and supports UDP protocols. Devices that are automated and tailored for limited resources have lower processing speeds, lower power consumption, and smaller memory capacities. When MQTT is not needed, the CoAP protocol may be utilized. CoAP has a 4 byte header and is also used for one-to-one communication. Compared to TCP and http, CoAP packets are smaller. Designed for Machine to Machine (M2M) uses, such as intelligent energy and automated buildings. The IETF Constrained Restful Environment (Core) working group created the Constrained Application Protocol (CoAP), a meeting layer convention, to provide a lightweight RESTful (HTTP) interface. The standard interface used by HTTP customers and workers is called Illustrative State Transfer, or REST. REST might be forced to use lightweight apps, like those in the Internet of Things, which could result in significant overhead.Because of the way the CoAP architecture is built, low-power sensors may still be enabled to use the Restful method to consume less power and meet certain constraints. Reliability is included into the CoAP architecture via the usage of the User Datagram Protocol rather than the Transition Control Protocol.

The layers of CoAP architecture are separated:

- Layer of information and communication;
- Layer of authorization and acknowledgment;

The solicitation/reaction sub-layer is responsible for communication, whereas the informing sub-layer is accountable for consistency and message duplication.

Four informing mechanisms exist for CoAP:

- Confirmable
- Nonconfirmable
- Piggyback
- Separate

1.3. XMPP

It has presence and communications protocols that are extendable. The primary issue with IOT protocols is interoperability; however XMPP federation offers a fantastic remedy. While XMPP specifies the message structure and obtains structural data, MQTT does not specify the message structure. It also simplifies the message and validates it. Essentially, it is used to comprehend data originating from various gadgets. Device identities, or Jabber ids, are produced by XMPP, while MQTT identities are created by the broker's implementation. Because XMPP allows federation, devices from many manufacturers linked to various platforms may communicate with one another using standard communication protocols. In contrast, MQTT implementation becomes quite challenging as the number of devices rises, whereas XMPP expands relatively easily. Actually, the protocols are open standards. XML, or the real-time interchange of structured data, is also its foundation. The protocol is open-standard.

Advantage

- Decentralization: Anyone may operate their own XMPP server; there is no central server.
- Open standards: These specifications may be implemented without the need for royalties or permits.
- Security: Encryption, authentication, and so on.
- Adaptability — Encourages compatibility

Disadvantage

- Lacks support for Quality of Services (QoS).
- Communications using text result in increased network overhead.
- Before being sent, binary data must first be converted to base64.

1.4. COMPARATIVE STUDY

While distinct modes are used for demand/reaction, confirmable and non-confirmable modes speak to the solid and problematic transmissions separately. Piggyback is used for direct communication between employees and customers, in which the employee responds directly to the message after receiving it—that is, inside the affirmation message. However, when the employee response is received in a message apart from the affirmation, a different manner is used, which may require some effort on the part of the employee. Similar to HTTP, CoAP employs requests for GET, PUT, PUSH, and DELETE messages to retrieve, create, update, and remove each message separately.

Table 1: Comparative Study of MQTT, CoAP and XMPP

Protocols	Header size	Quality of services	Security	Port Number	REST Architecture	Encoding format
MQTT	2bytes	It supports 3 layers of QoS	Supports TLS and SSL	1883	Not supported	Binary/Text
CoAP	4byte	CON and NON	DTLS	5683	It supports	Binary/Text
XMPP	No limit	Till no quality of services	Supports TLS and SSL	5222	Absent	XML based

2. Related Works

This section includes a few earlier studies that examined the effectiveness of various communication protocols in the literature.

The IoT concept's several data protocols, including XMPP, CoAP, AMQP, MQTT, DDS, and MQTT-SN, are handled by Anusha et al. By comparing each data protocol's functionality with that of the others, the authors hope to provide light on performance measures including latency, bandwidth use, message size, and packet loss rate. The performance of each technique is assessed based on the intended use. Furthermore, since XMPP uses XML stanza-based transmission for online instant messaging apps, it performs better.

By contrasting XMPP with CoAP, Bandyapadhyay et al. want to ascertain whether protocol is more appropriate for various application domains with limited devices. Evaluations of protocols are carried out on Intel X86 processors and Android operating systems. The "Mosquitto" project for MQTT and the "libcoap" library for CoAP are the software technologies being used for implementation. Additionally, network traffic is analyzed using Wireshark. The dependability, energy usage, and bandwidth utilization of several protocols are compared. Based on the findings, CoAP outperforms MQTT in terms of optimizing energy utilization.

To look at the capacities of CoAP, MQTT, DDS, and a customized UDP-based protocol in clinical checking applications, Chen et al. lead a broad review that tends to the transfer speed utilization, dormancy, and parcel misfortune measurements on constant information that is assembled from patients. Moreover, the creators explain how protocols work on restricted, shoddy remote organizations. The Raspberry Pi model 2, Arduino Uno rendition 3, and ASUS

Zenbook Windows PC are the equipment innovations. "Californium CoAP" is the product innovation for the CoAP server and client; "HiveMQ" is the product for the MQTT server; "Mosquitto" is the product for the Agent and MQTT clients (both endorser and distributer); and "OpenDDS" is the product for the DDS server and client. The instruments "TBF," "NetEM," and "Wireshark" are utilized to inspect protocol performances. As per performance measurements, in low quality remote organizations, TCP-based protocols (DDS and MQTT) are both more trustworthy than UDP-based protocols (Custom-UDP and CoAP). In a similar organization climate, TCP-based protocols display higher postponement contrasted with UDP-based protocols. Besides, DDS beats MQTT under horrible organization conditions.

The MQTT and CoAP protocols' performances are evaluated and differentiated by Thangavel et al. as far as parcel misfortune, dormancy in retransmitting messages, and information sent per message. The journalists give close consideration to how information is sent from sensors at the door hub to the merchant or back-end server for MQTT or CoAP. The equipment innovations used are a netbook for Wide Region Organization (WAN) emulator, a PC for server, and a BeagleBoard-xM for middleware execution. The product advancements incorporate "Wireshark" for estimating measurements, "Mosquitto" for MQTT Agent, "libcoap" library for CoAP, and "Wanem," the wide region network copying, for message transport. As per the outcomes, MQTT messages have less postponements for less parcel misfortune than CoAP messages. Interestingly, MQTT has more bundle misfortune because of longer postponements than CoAP. Moreover, when the message size and parcel misfortune rate are lower, CoAP encounters less traffic.

CoAP, MQTT, XMPP, and WebSocket are among the communication protocols dealt with by Kayal et al. The paper's essential objective is to evaluate protocol performances in restricted gadgets to empower proficient communication. As needs be, by changing the traffic stacks, a savvy stopping situation is instituted to look at the response seasons of the protocols. "Openfire" project for XMPP Server, "libcoap" library for CoAP, "Mosquitto" project for MQTT Agent, "Smack Client" for XMPP Client, "HiveMQ" for Websocket Specialist, and "Paho Python" library for WebSocket client are used as programming advancements. The outcomes show that CoAP beats elective line based protocols when server utilization is lower.

3. Preliminaries

The initials for the MQTT, XMPP, and CoAP protocols are provided in this section.

A. CoAP (Constrained Application Protocol)

A transfer protocol designed specifically for limited nodes and networks, CoAP is built on the UDP layer. Because HTTP had a major design influence on CoAP, the REST architectural style is used. The Internet Engineering Task Force, an impartial organization, then standardizes it (IETF). Furthermore, peer-to-peer communication takes place between the client and server. However, unicast and multicast queries might be answered by the server or the client. To request resources from the server, the CoAP offers four distinct message types: GET, PUT, POST, and DELETE. Fig. 2 depicts the communications architecture of CoAP.

Advantages of CoAP:

- Uses UDP layer to transmit tiny packets for quick communication.
- There is asynchronous communication available.
- Intermediary servers are not necessary for peer-to-peer communication between clients. Furthermore supported is many-to-many communication.
- By approving encryption and security, Datagram Transport Layer Security (DTLS) offers integrity, security, and privacy.
- A good choice for devices with limitations.

Disadvantages of CoAP:

- Messages are not trustworthy. To verify that the communication arrived, ACK (acknowledgment) packets are transmitted. It does not, however, make it evident whether these signals are fully or accurately deciphered.
- Standardizing is still ongoing. It is chosen as the protocol that is the least standardized among all of the others.

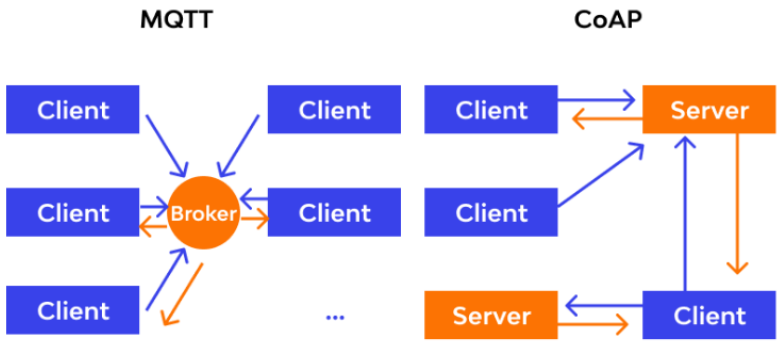

Figure 2: Constrained nodes-server CoAP architecture

B. MQTT (Message Queuing Telemetry Transport)

A lightweight M2M communication protocol for limited devices and unstable networks is called MQTT. It uses TCP/IP and features a publisher/subscriber client. Moreover, TCP offers bidirectional connections between nodes and message dependability. Messages pertaining to certain subjects may be published by nodes on Broker, wherein other nodes can subscribe to receive messages. The Broker participates in communication and is responsible for managing this message's flow. In actuality, Broker serves as a server via which message traffic is routed and clients may post and subscribe to topics. Additionally, customers may approve a broker by logging in with their login and password. To assess the message's transmission quality, MQTT offers three QoS tiers. SSL/TLS encrypts messages to provide security. The publisher and subscriber structure-based messaging architecture of MQTT is seen in Figure 3.

Advantages of MQTT:

- Message dependability is ensured by supporting QoS levels, and bandwidth is efficiently used via packet agnosticism. Text or binary may be present in the data.

- The publish/subscribe technique may work in one-to-one, many-to-many, or one-to-none scenarios. Furthermore, bi-directional communication is provided via this technique.
- Uses straightforward communication techniques;
- Communication occurs between nodes. Messages are always published and subscribed.

Disadvantages of MQTT:

- It makes use of TCP/IP, which, in contrast to UDP, demands higher communication capabilities.
- The broker's communication skills are restricted.
- Every node is linked to the broker. Therefore, when the broker fails, the communication breaks down.

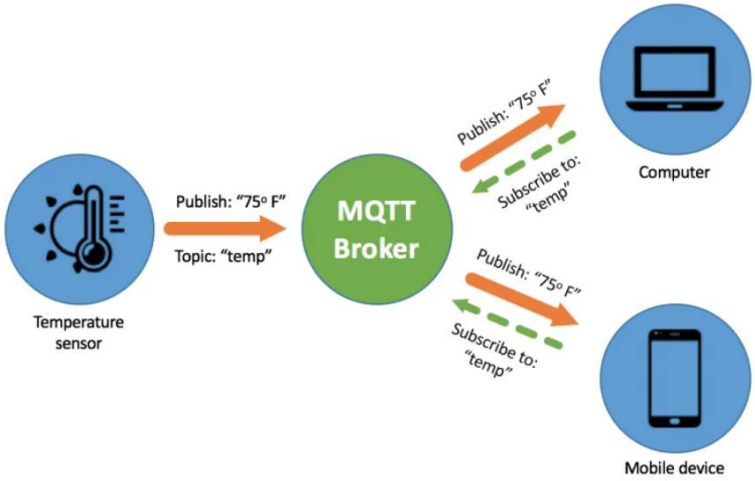

Figure 3: SUB: sign up, Humidity: HUM, Temperature: TEMP MQTT architecture connecting the server and limited nodes

➢ CoAP vs. MQTT

Since there are a lot of similarities, it's okay if you think these two are the same. For example, they are both used in Internet of Things devices because they need fewer network packets, which results in better power-optimized performance, lower storage requirements, and longer battery life.

CoAP and MQTT differ from one another in a number of ways:

Table 2: CoAP vs. MQTT

MQTT	CoAP
The primary players in this arrangement are publishers and subscribers.	Utilizes inquiries and answers
Message dispatching is handled by a central broker who chooses the best route from publisher to client.	Dispatching messages occurs on a one-to-one (unique) basis. It works in the same way as HTTP.
Operations focused on events	feasible for transferring states

For the client to be successful, a persistent and continuous TCP connection must be established with the broker.	Sync UDP packets are used by the involved parties for communication and message passing.
No message labeling is required, but several messages must be used for various objectives.	It facilitates message discovery and provides accurate definitions.

C. XMPP (Extensible Messaging and Presence Protocol)

XMPP is a protocol that uses XML technology to facilitate file transfers and conversation between nodes in a dispersed network. TCP is the foundation for several forms of communication, including as audio and video transfers, group chat, presence, instant messaging, and collaboration. XML stanzas provide real-time communication as well. Through XMPP, the server may provide access to certain clients and facilitate communication between them via XML stanzas. Additionally, XMPP designates a client presence indication, such as busy, offline, or online. As a result, the client informs the server whether it is appropriate for communications. Figure 4 illustrates the communications architecture of XMPP.

Advantages of XMPP:

- It is flexible and extensible.
- Constant communication is provided by many servers.
- Facilitates communication between servers and clients as well as between clients and servers.
- Additional communications choices are provided by the presence indication.
- Its TCP protocol, which is based on XML-Stanzas, offers communication with more dependability.

Disadvantages of XMPP:

- The usage of XML Stanzas in communication results in delays;
- The server's communication capacity is restricted;
- Client requests for access to the server need a lengthy permission process

4. Experimental Results And Performance Evaluations

The experimental arrangement and the results of the CoAP, MQTT, and XMPP performance evaluations are introduced in this part. An Intel Galileo Gen 2 board and an ASUS PC running Windows 64-digit working framework with an Intel Center i7-6700HQ computer chip running at 2.60 GHz and 4 GB of Smash are utilized for the examination. The PC is utilized to carry out the server side of protocols and the information gathering parts of clients like XMPP and MQTT, while the board is utilized to execute the client side. These devices address each other over a neighborhood. To analyze the protocols under indistinguishable conditions, the clients of these protocols are coordinated on a similar board.

Table 3: Performance Metrics Comparison of CoAP, MQTT, and XMPP

Metric	CoAP (µs)	MQTT (µs)	XMPP (µs)
Packet Creation Time	419	119	12855
Packet Transmission Time	821	589	41383

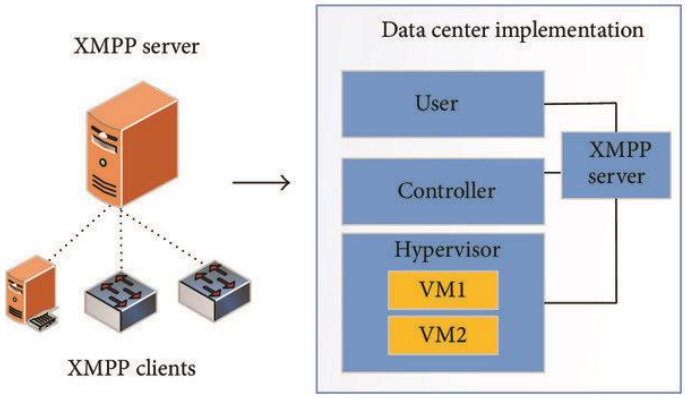

Figure 4:XMPP client-server communication

Table 4: Performance Comparison of CoAP, MQTT, and XMPP in IoT Communication

Metric	CoAP	MQTT	XMPP
Packet Creation Time (µs)	419	119	12855
Packet Transmission Time (µs)	821	589	41383
Average Message Delivery Time (µs)	-	589	41383
Standard Deviation of Delivery Time (µs)	-	12	115
Message Loss Rate (%)	Low	Low	Low
Bandwidth Utilization	Low	Low	Low
Scalability	Moderate	High	Moderate
Protocol Overhead (bytes)	Low	Low	High
Resource Usage (CPU/RAM)	Low	Low	High
Network Load	Low	Low	High
Complexity	Low	Low	High
Reliability	Moderate	High	Moderate
Security	Moderate	High	Moderate
Flexibility	Low	High	Moderate
Optimization	Low	High	Low
Standardization	Moderate	High	Low

Using the same board, clients successively submit messages to the servers. "CoAP-simple-library," "Pubsubclient," and "XMPPArduino" are the names of the CoAP client, the MQTT publisher client, and the XMPP protocol message-sending application used on the board, respectively. Additionally, the "Californium," "Paho," and "Smack" projects are used to develop the MQTT subscriber client, the XMPP message listener client, and the CoAP server. Finally, the MQTT Broker and XMPP server are implemented using the "Mosquitto" and "Openfire" projects, respectively. Figures 5 and 6 depict the hardware configuration. The protocols that are discussed use these libraries to continually transmit real-time environmental data, such as temperature, humidity, and light, in order to guarantee fair comparison.

We then calculate each protocol's packet formation time with regard to these protocols. The rate of message production is indicated by the packet creation time. For instance, the PUT message type is exclusive to CoAP, enabling the real-time transmission of received data to the primary server (data is gathered in the client board for TCP-based protocols).

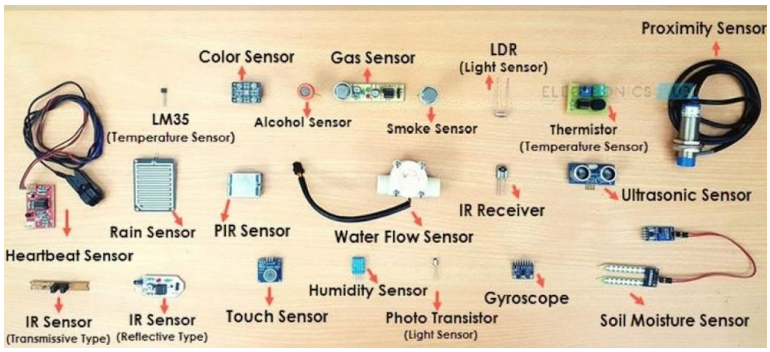

Figure 5: Test bed sensors (temperature, humidity, light)

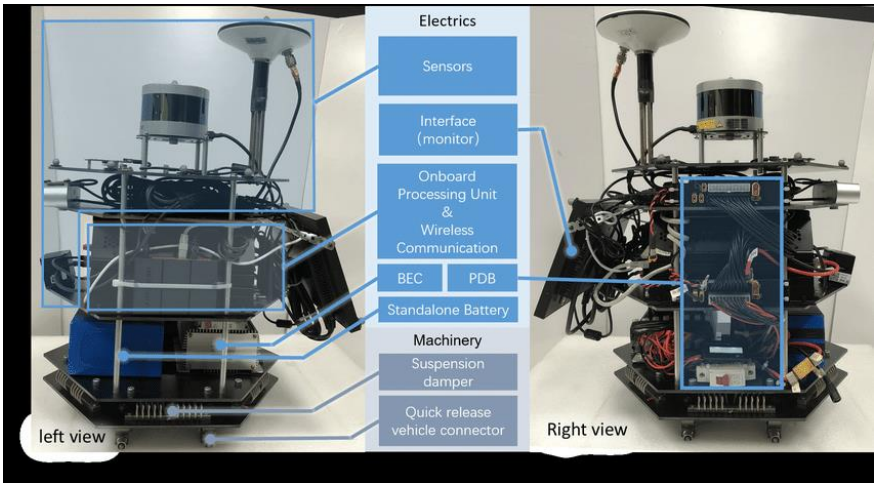

Figure 6: Test bed processing unit

For every 100 messages, 25 are made in order to provide reliable findings. The typical creation times for MQTT, XMPP, and CoAP are 119, 419, and 12855 microseconds, respectively. These observations show that MQTT generates packets more quickly than other protocols. The packet production times for MQTT and CoAP are almost the same, however XMPP takes much longer than the other protocols. Other protocols have more straightforward packet formats than XMPP since it creates packets using XML Stanza. Nevertheless, library optimization determines how the CoAP and MQTT vary from one another. The average creation times for each treatment are shown in Fig. 7.

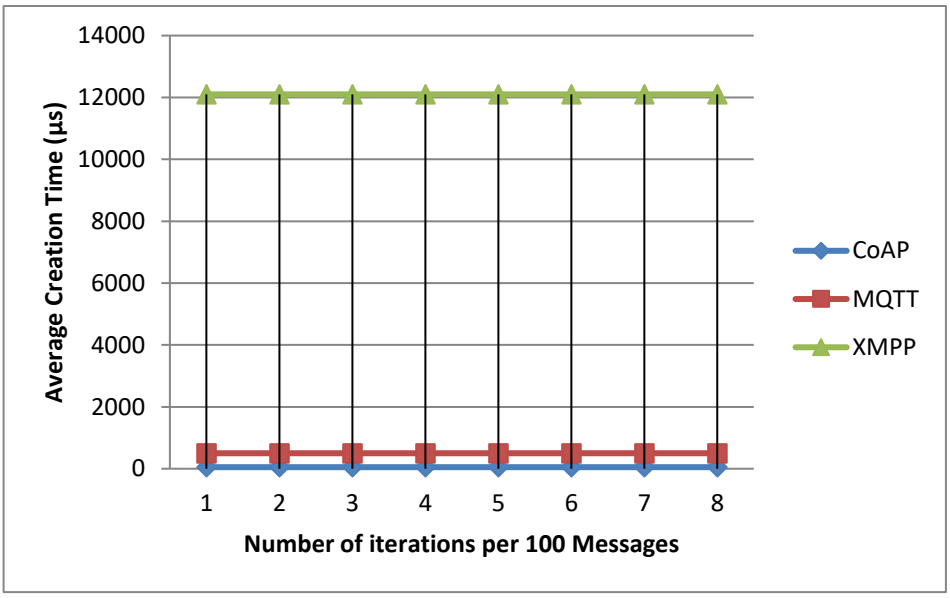

Figure 7:Packet creation time for protocols (µs)

Second, estimations are made of the parcel transmission timings from the board to the essential server. To be more exact, the time span between a client situated on the load up and the server situated on the principal server is utilized to measure the transmission time for CoAP. The timings between the clients on the board and the fundamental server are estimated for MQTT and XMPP. Besides, the ACK messages for XMPP and MQTT are missing from these information. They are computed by taking into account the protocols' delivery times under the identical circumstances. For this assessment, a total of 100 messages are utilized.

The average arrival time for MQTT messages is around 589 microseconds, while CoAP messages arrive in 821 microseconds and XMPP messages arrive in 41383 microseconds. XMPP interprets transmitted messages in accordance with the XML format, which is why it is slower than other protocols. Considering that MQTT uses TCP for informing and CoAP utilizes UDP, it is anticipated that CoAP performs better compared to MQTT. Then again, MQTT needn't bother with a message to sit tight for an ACK. Distributer sends messages to Facilitate, who then gives the endorser the information. Moreover, all protocols utilize small message bundle sizes. Besides, contrasted with CoAP, MQTT is a more normalized and streamlined protocol. These variables cause COAP messages to arrive at the primary server two times as leisurely as MQTT ones do. The average transmission times for each technique are shown individually in Fig. 8.

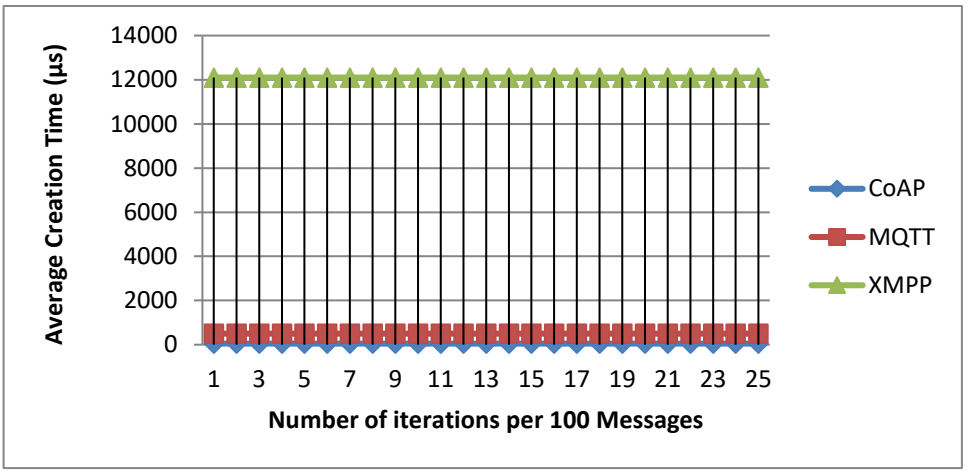

Figure 8: Packet transmission time for protocols (µs)

We demonstrate that protocols are practically necessary for IoT devices to get real-time environmental data. We analyze the performance metrics of MQTT, XMPP, and CoAP in accordance with the criteria, and by contrasting these metrics, we want to highlight the variations between these protocols in an environment of real-time communication. In the context of real-time communication, protocols are evaluated based on metrics such as packet generation time and packet delivery speed to ascertain latency differences. Consequently, even though CoAP is an UDP-based protocol, MQTT has faster packet generation and transmission times than other protocols. Additionally, packet delivery via MQTT is twice as quick as packet delivery via CoAP. MQTT is superior to competing protocols for a number of reasons, including the large bandwidth of the network, the smaller size of the packets being sent, and the less standardized COAP. Examined in such a network, XMPP has a slowing structure similar to an XML stanza, adding significant delay in comparison to other protocols.

5. Conclusion

In the context of IoT communication, our experimental assessment of the CoAP, MQTT, and XMPP protocols shows unique performance characteristics for each protocol. When it comes to packet generation and transmission timings, MQTT leads the field, demonstrating its effectiveness in real-time communication situations. On the other hand, CoAP performs well but is not as optimized and standardized as MQTT. Because of its XML stanza format, XMPP is much slower, with slower packet generation and transmission times. These results highlight the significance of selecting protocols depending on particular application needs, which has practical consequences for developers and practitioners. MQTT is the recommended alternative in situations requiring dependable and quick communication because to its efficiency and extensive acceptance. Nonetheless, CoAP's lightweight architecture and compatibility with limited devices make it useful even today. All things considered, our comparison analysis highlights how important protocol standardization and optimization are to enabling smooth IoT communication, opening the door for further study and testing in various IoT scenarios.

References

1. 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.
2. Bayılmış, C., Ebleme, M. A., Çavuşoğlu, Ü.,Kücük, K., &Sevin, A. (2022). A survey on communication protocols and performance evaluations for Internet of Things. Digital Communications and Networks, 8(6), 1094-1104.
3. Bhuiyan, M. N., Rahman, M. M., Billah, M. M., & Saha, D. (2021). Internet of things (IoT): A review of its enabling technologies in healthcare applications, standards protocols, security, and market opportunities. IEEE Internet of Things Journal, 8(13), 10474-10498.
4. D. Thangavel, X. Ma, A. Valera, H. X. Tan and C. K. Y. Tan, "Performance Evaluation of MQTT and CoAP via a Common Middleware," 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 21–24 Apr. 2014, pp. 1-6.
5. Fortino, G., Savaglio, C., Spezzano, G., & Zhou, M. (2020). Internet of things as system of systems: A review of methodologies, frameworks, platforms, and tools. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51(1), 223-236.
6. Gebremichael, T., Ledwaba, L. P., Eldefrawy, M. H., Hancke, G. P., Pereira, N., Gidlund, M., &Akerberg, J. (2020). Security and privacy in the industrial internet of things: Current standards and future challenges. IEEE Access, 8, 152351-152366.
7. Gupta, B. B., &Quamara, M. (2020). An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols. Concurrency and Computation: Practice and Experience, 32(21), e4946.
8. Kassab, W. A., &Darabkh, K. A. (2020). A–Z survey of Internet of Things: Architectures, protocols, applications, recent advances, future directions and recommendations. Journal of Network and Computer Applications, 163, 102663.
9. Kolisnyk, M. (2021). Vulnerability analysis and method of selection of communication protocols for information transfer in Internet of Things systems. Radioelectronic and computer systems, (1), 133-149.
10. Lombardi, M., Pascale, F., &Santaniello, D. (2021). Internet of things: A general overview between architectures, protocols and applications. Information, 12(2), 87.
11. M. Anusha, E. S. Babu, L. S. M. Reddy, A. V. Krishna and B. Bhagyasree, "Performance Analysis of Data Protocols of Internet of Things: Qualitative Review," International Journal of Pure and Applied Mathematics, vol. 115, no. 6, pp. 37-47, 2017.
12. P. Kayal and H. Perros, "A Comparison of IoT application layer protocols through a smart parking implementation," IEEE 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Mar. 2017, pp. 331- 336
13. S. Bandyopadhyay and A. Bhattacharyya, "Lightweight Internet Protocols for Web Enablement of Sensors using Constrained Gateway Devices," IEEE 2013 International Conference on Computing, Networking and Communications (ICNC), pp. 334-340, Jan. 2013.
14. Y. Chen and T. Kunz, "Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network," IEEE 2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT), Apr. 2016, pp. 1-7.
15. Zeadally, S., Das, A. K., &Sklavos, N. (2021). Cryptographic technologies and protocol standards for Internet of Things. Internet of Things, 14, 100075.