International Journal of Networked and Distributed Computing
https://doi.org/10.1007/s44227-024-00021-4

RESEARCH ARTICLE

Analysis and Performance Comparison of IoT Message Transfer Protocols Applying in Real Photovoltaic System

Khoa Thi Minh Tran[1] · **Anh Xuan Pham**[1] · **Nam Phuong Nguyen**[1] · **Phuc Thi Dang**[1]

Received: 2 August 2023 / Accepted: 17 January 2024

Abstract
The adoption of reliable and real-time communication technology is an absolute necessity for the advancement of Internet of Things (IoT) applications. Messaging protocols such as MQTT, AMQP, and HTTP are frequently used for communication with resource-constrained IoT devices. However, choosing a suitable and effective messaging protocol presents a daunting challenge for organizations, as it depends on the specific characteristics and messaging requirements of the IoT system. Therefore, it is crucial to have a comprehensive understanding of three established messaging protocols, such as the Hypertext Transfer Protocol (HTTP), the Message Queuing Telemetry Transport (MQTT), and the Advanced Message Queuing Protocol (AMQP), to appropriately apply them in practical projects. In this paper, information technology solutions are provided for a chain of solar farms to improve harvest productivity, facilitate warning notifications, and enable remote control. Subsequently, a detailed comparative analysis is performed, considering various interconnected criteria, to gain valuable insight into the strengths and limitations of these protocols. The results show that MQTT and AMQP play a role in enhancing overall efficiency and speed within the framework of our suggested photovoltaic system.

Keywords IoT Message Transfer Protocols · IoT smart agriculture · Message Queuing Telemetry Transport · Advanced Message Queuing Protocol · Hypertext Transfer Protocol

1 Introduction

Technological revolutions have a significant impact on human development and constantly shape our lives in a more progressive direction. With the growing population and increasing demands, services and utilities evolve accordingly. In particular, the integration of electronic, information technology, and communication technologies into our daily lives has been particularly transformative. In the realm of the Internet of Things (IoT), the application of these technologies, specifically messages-transfer protocols between devices, extends beyond research purposes to include entertainment, manufacturing, business, and more. This ever-expanding scope of applications aims to meet the diverse needs of various fields [1–6]. IoT encompasses a range of messaging protocols designed to respond to different factors, such as application deployment, communication modes, suitability for specific applications, device characteristics, security features, and message transmission over the Internet. The IoT landscape, including devices, standards, technologies, and platforms, is constantly evolving and progressing. Currently, the Internet of Things has gained widespread familiarity and finds extensive applications in various domains of human life, particularly in technologically advanced and developed countries. The communication technology between IoT devices has evolved, driven by the practical implementation of various systems, resulting in distinct advantages and improved service quality [7–9]. Consequently, constructing an effective IoT model requires

Anh Pham Xuan, Nam Nguyen Phuong, and Phuc Dang Thi have contributed equally to this work.

✉ Khoa Thi Minh Tran
ttmkhoa@iuh.edu.vn

Anh Xuan Pham
phamxuananh.309@gmail.com

Nam Phuong Nguyen
nguyenphuongnam2698@gmail.com

Phuc Thi Dang
phucdt@iuh.edu.vn

[1] Faculty of Information Technology, Industrial University of Ho Chi Minh City, 12, Nguyen Van Bao Street, Ward 4, Go Vap District, Ho Chi Minh City 7000, Vietnam

Published online: 23 February 2024

a deep understanding of the characteristics and features of different types of information communication. Selecting the most appropriate protocols for specific cases or services becomes crucial to optimizing IoT performance and achieving the desired results. The chosen approach emphasizes the importance of understanding the intricacies and complexities that underlie each data transport event within the Internet of Things (IoT) ecosystem. By recognizing and analyzing the specific circumstances and activities that occur behind these data transfers, it becomes possible to ensure smooth and accurate operation of a universal Internet system that manages multiple devices seamlessly. To achieve this, a comprehensive examination of the communication protocols, data formats, and network infrastructure involved in IoT interactions is vital. Each data transport event may differ in terms of data volume, frequency, latency requirements, and security considerations. Properly addressing these factors can optimize the performance of the IoT system and improve the user experience.

Within the IoT domain, numerous protocols have emerged with diverse characteristics and specifications. Among the popular IoT application layer protocols are message queueing telemetry transport (MQTT) [10–12], Constrained Application Protocol (CoAP) [13], Advanced Message Queuing Protocol (AMQP) [13, 14], eXtensible Messaging and Presence Protocol (XMPP) [15], simple text-oriented messaging protocol (STOMP) [16], REpresentational State Transfer (RESTful) HyperText Transfer Protocol (HTTP) [13], Simple Object Access Protocol (SOAP), WebSocket and JavaScript IoT. These protocols cater to different use cases and communication requirements, offering a wide range of choices for developers and organizations seeking to deploy IoT solutions across various applications. Many IoT applications have been proposed that use IoT communication protocols. The authors in [17] proposed an application of LoRa (Long-Range Access) to optimize IoT using MQTT to monitor fish feeding. In the research paper [18], an IoT solution is suggested, using the MQTT protocol to transmit waveforms collected by seismic nodes. The study also provides a performance comparison between this proposed solution and the current standard used in early earthquake warning systems. The two papers [19, 20] work on evaluating the performance of IoT communication protocols for the application layer, such as AMQP, CoAP, MQTT and HTTP in the context of test scenarios or in the real smart city system.

Clean energy and perpetual mining are close to replacing the world's future emission-generating energy sources. However, the output generated is not competitive and impressive enough for investment, so we need solutions to maximize the use of natural energy sources. This project proposes an energy supply system for a chain of farms that uses solar energy. The system can take advantage of the full absorption of sunlight during the day and use that energy to operate at night. This paper focuses on an investigation of three widely recognized message-transfer protocols within the realm of IoT development. These protocols include the Hypertext Transfer Protocol (HTTP), which is used in conjunction with the Representational State Transfer (REST) data structure, as well as the Message Queuing Telemetry Transport (MQTT) and Advanced Message Queuing Protocol (AMQP). In addition, we have put these protocols into practical application in a real project aimed at finding appropriate solutions to improve the absorption of a solar farm chain. Through this implementation, we offer insightful comments and summarize the experiences gained from working with these protocols, providing valuable insights for future IoT endeavors.

The remainder of the paper is organized as follows. Section 2 provides an overview of studies that are commonly associated with IoT communication protocols. Section 3 provides a general overview of the system architecture of the recommendation protocols used for the photovoltaic system. In more detail, this section describes the approach to implementation, analysis, and evaluation of the proposed system. Finally, Sect. 4 presents the conclusions and future work of the project.

2 Literature Review

This section briefly summarizes the most prominent characteristics of IoT messaging transfer protocols that are commonly used to communicate among IoT devices that have resource limitations; namely, HTTP, MQTT, and AMQP.

2.1 REpresentational State Transfer (RESTful) HyperText Transfer Protocol (HTTP)

HTTP is a hypertext transfer protocol. This protocol is used to transmit information from Web Server to Web Client in the Client-Server model for the Internet, World Wide Web; HTTP belongs to the application layer of the TCP / IP protocol suite. The main mechanism of operation of HTTP is Request-Response: The Web Client sends the Request to the Web Server, the Web Server processes, and sends the Response to the Web Client.

REpresentational State Transfer (REST) is a type of data structure transformation, which is an architectural type for designing connected applications. It uses the HTTP protocol to create communication between machines. So instead of using a URL for processing some user information, REST sends an HTTP request like GET, POST, DELETE, etc. to a URL to process the data. Application Programming Interface (API) is a set of rules and mechanisms by which

Fig. 1 QoS 0 of MQTT protocol

an application or component will interact with another application or component. The API can return the data you need for your application in common data types such as JSON or XML. REST API is a standard used in the design of APIs for web applications to manage resources. RESTful is one of the most commonly used API design types today to let different applications (web, mobile, etc.) communicate with each other. Focuses on system resources (text files, images, audio, video, or dynamic data, etc.), including resource states that are formatted and transmitted via HTTP.

2.2 Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) is a publish/subscribe messaging protocol used for devices in the Internet of Things with low bandwidth, high reliability and the ability to be used in unstable networks. Because this protocol uses low bandwidth in high-latency environments, it is an ideal protocol for machine-to-machine (M2M) applications [21].

The high-level architecture of MQTT consists of two main parts: Broker and clients. In particular, the broker is considered the center; it is the intersection point of all incoming connections from the client. The main task of the broker is to receive messages from the publisher, arrange the messages in the queue, and then forward them to a specific address. The broker's secondary task is that it can take on a few more features related to the communication process, such as message security, message storage, logs, etc. The client is divided into two groups: publisher and subscriber. Clients are software components that work on edge devices, so they are designed to be lightweight. The client only does at least one of two things: publish messages on a specific topic or subscribe to a certain topic to receive messages from this topic [12]. The MQTT protocol was born in 1999 and up to the present time, the MQTT version 3.5 is recognized as OASIS standard.

In a system using MQTT protocol, many station nodes (called MQTT clients - referred to as clients) connect to an MQTT server (called broker). Each client will subscribe to one or several channels (topic), such as "/ client1 / channel1", "/ client1 / channel2". This sign-up process is called "subscribe", just as we subscribe to a YouTube channel. Each client will receive data when any other station sends data and the registered channel. When a client sends

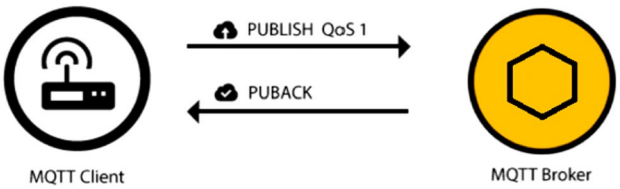

Fig. 2 QoS 1 of MQTT protocol

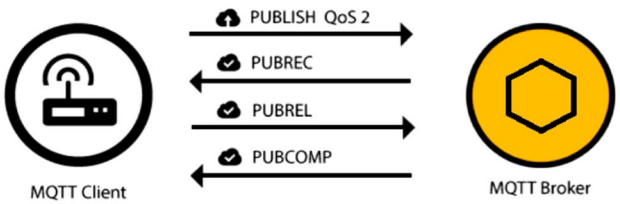

Fig. 3 QoS 2 of MQTT protocol

data to that channel, it is called a "publish" [22, 23]. There are 3 QoS options when "publish" and "subscribe":

1. Broker/client will send data exactly once, and sending process is confirmed by only TCP/IP (Fig. 1).
2. Broker/client will send data with at least one confirmation from the other end, meaning that there may be more than 1 confirmation received data (Fig. 2).
3. Broker/client makes sure that when sending data, the receiver only receives them once; this process has to go through 4 handshaking steps (Fig. 3).

Retain is a flag attached to a message of the MQTT protocol. Retain only receives values 0 or 1 (corresponding to 2 logical values false or true). If retained by 1, the broker will save the last message from a topic with the corresponding QoS level. When the client begins to subscribe to the topic where the message is saved, the client immediately receives the message.

MQTT Bridge is a feature of MQTT Broker that allows MQTT Broker to connect and exchange data with each other. To use this feature, we need a minimum of 2 Brokers, in which any Broker will be configured to Bridge. When

configuring the MQTT bridge, we need to pay attention to the following parameters:

- Address: broker address to connect
- Bridge protocol version: the version of the MQTT protocol that is shared by two brokers.
- Topic: this section defines three parameters: topic name is exchanged between two brokers, exchange direction (1-way or 2-way), and topic mapping between two brokers

MQTT is designed to be as light and flexible as possible. Therefore, it has only one layer of security at the application layer: authentication security (authenticating clients that have access to the broker). However, MQTT can be installed in combination with other security solutions, such as combining with VPN at the network layer or SSL / TLS at the transport layer. MQTT is designed to serve machine-to-machine communication, but, in fact, it proves to be more flexible than expected. It is completely applicable to other communication scenarios such as machine-to-cloud, cloud-to-machine, and app-to-app. As long as there is a suitable broker and the MQTT client is properly installed, devices built on different platforms can communicate with each other easily.

2.3 Advanced Message Queuing Protocol (AMQP)

AMQP specifies the concept of a broker, a network service that routes messages between parties communicating with various levels of reliability. Create interoperability between clients and brokers, with the purpose of allowing different applications and systems to work together, regardless of their internal design, standardize the delivery of messages on an industrial scale [10, 13, 24–26].

AMQP defines how the network works and how the message broker works:

- Routing and storing messages with message brokers and rule sets to determine how components are involved.
- Connection protocol to show how the client and message broker communicates as above.

Before AMQP, there were many types of message brokers and signal transmission applications from many vendors. However, their big problem is their lack of interoperability; there is no simple way for all to work together. The only way that different systems with different protocols work is to use an additional layer of signal conversion called a messaging bride. These systems must use adapters to receive signals like normal clients, from many and different signal systems.

In a simple system, a message broker represents intermediaries for all types of services. Functioning as a "Post Office", the Producer sends a "letter" to this post office. At the end of the session, the consumer, who registered at the post office, came to pick up this letter (Fig. 4).

Queue is a component defined in the Broker, operating under the First In First Out (FIFO) rule. The message placed in the first queue is first retrieved. Of course, messages are not sent directly to Queue, spawning Exchange to redirect messages to the specified rules. There are some types of Exchange, such as: Direct Exchange, Default Exchange, Topic Exchange, Fanout Exchange, and Header Exchange.

- Direct Exchange: transmits messages based on the given routing key, which exchanges data to queue with the correct binding key for that routing key.

Default Exchange: is a pre-declared Direct Exchange form without a name, usually an empty string. When using the default exchange, the message is delivered to the queue, with the main name being the routing key of the message.

- Topic Exchange: redirects the data to the queue based on the "wildcard" that matches the routing key and the routing pattern string specified when binding.
- Fanout Exchange: works by copying and directing the received message to all the queues to which it is associated and can use some routing key or routing pattern similar to the way direct and topic exchange works. Fanout exchanges can be useful when the same message needs to be sent to one or more queues with consumers who can handle the same message in different ways.
- Header Exchange: navigate messages based on arguments that include headers and their values. This type of exchange is quite similar to the Topic exchange format,

Fig. 4 Simple operation of Broker AMQP

but routes messages by the values of the headers instead of the routing key. A message will match if the header value is equal to the value specified when binding.

3 Implementation, Analysis and Evaluation

3.1 Implementation of a Photovoltaic System

In the foreseeable future, clean energy and perpetual exploitation are poised to replace conventional emission sources worldwide. Despite the promise of these renewable resources, the generated output may not be competitive or impressive enough for substantial investments. As a result, it is imperative to design solutions that effectively optimize the utilization of these resources. The project's primary objective is to develop a comprehensive system that maximizes sunlight absorption throughout the day, thereby significantly increasing harvest yields for solar farms. Rather than relying solely on the current locations, the focus is on creating a fully utilized and integrated system. When data are obtained from a single unit, we can extrapolate and navigate the entire network of units, facilitating a more efficient and productive use of solar energy.

The system functions by collecting data from light sensors and transmitting it to another device for processing. This allows the solar panel to automatically align with the direction of sunlight. Additionally, the system integrates a rain sensor to provide rain alerts, sending notifications to users via email. Through an application installed on any device, users can conveniently monitor the light intensity's status and even control the navigation, enabling them to turn it on or off as needed.

Design description:

– On the panel surface, the four sensors must be at four corners, spreading the device across the maximum area to collect the lightest data from the directions, as shown in Fig. 5. The energy collected by the solar panel above is charged into a power supply, which will also power the devices on the surface that send data. The excess energy will also be the result of this project.

Figure 6 illustrates the system architecture used to evaluate the performance of the message protocols discussed in this document. The entire system will have three main components: The data collection and processing area on the solar farm; Cloud systems and services; User device.

The energy farm equipment consists of two primary components: The first part focuses on gathering optical data, which is subsequently transmitted to Cloud services. An Arduino Uno device receives data and communicates with NodeMCU, relaying them to cloud services via the MQTT protocol. The second part involves receiving information from the two aforementioned brokers, employing the MQTT protocol. It processes and controls two navigation motors based on this information. Additionally, it captures signals from a rain sensor and utilizes them to generate API requests to Adafruit's system.

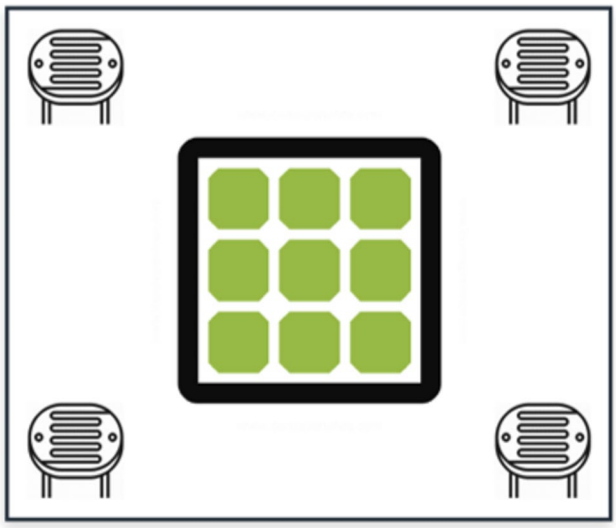

Fig. 5 Sensor design on the surface of solar panels

The cloud services will be used by three providers: CloudAMQP, which offers both MQTT and AMQP protocols, together with the API services of Adafruit IO and Zapier. Using the MQTT Broker, only one unit is required to collect light data and send it to a specific topic. Other units can easily connect and receive the data for processing. This simplifies project upgrades, maintenance, and scalability. Additionally, by integrating with a remote control application, we can monitor the current light intensity status of all four sensors simultaneously. When using the HTTP API protocol, data transmission occurs sequentially through each system based on the installation scenario. After sending rain data, the Adafruit IO service calculates the requirements and triggers a working signal to the email department through Zapier. Status information is stored in the message queue within the AMQP broker. This allows connected clients to receive data on their screens and even remotely activate or shut down the system using a mobile device.

User devices, which can be as simple as a phone or computer with a control application installed, are connected to the AMQP broker. This connection enables the user devices to access the system's operating status and even send control signals as needed.

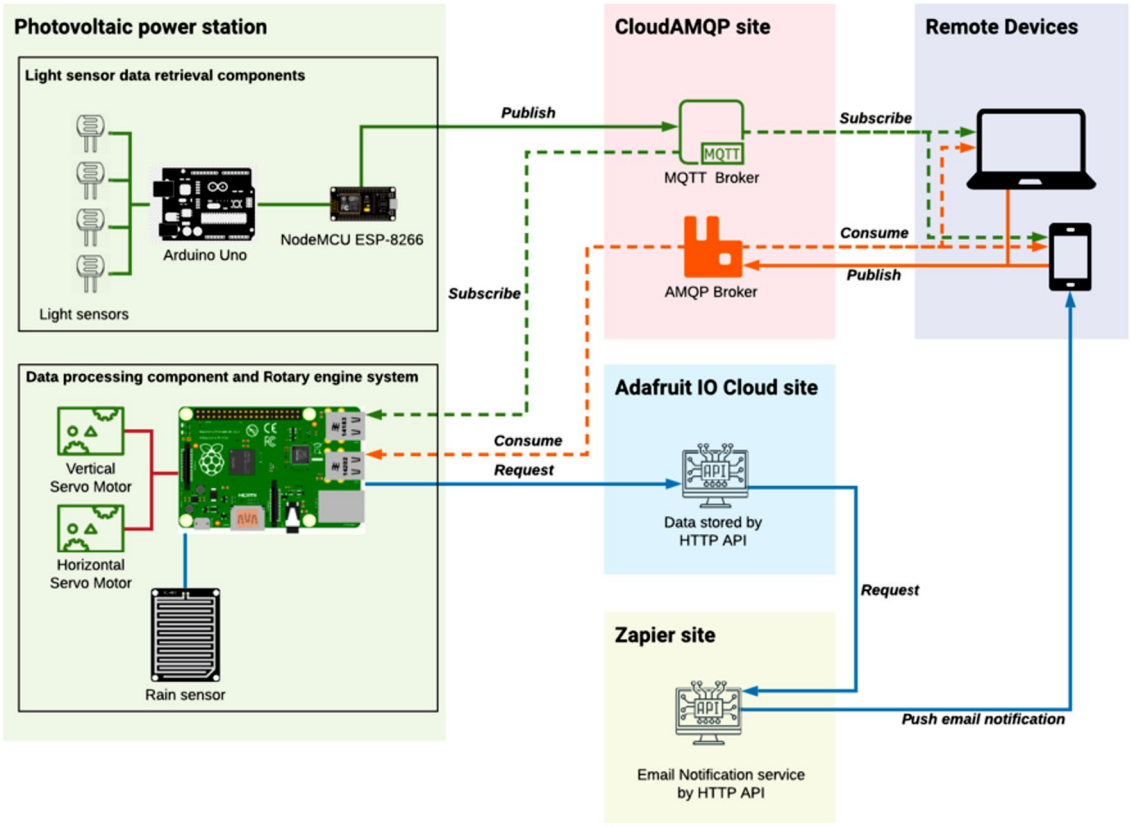

Fig. 6 The overall design model of the smart solar absorption system

Table 1 Devices and Services of the project

Devices and services	Number
Arduino Uno	1
Raspberry Pi 3	1
NodeMCU ESP8266	4
Light sensor	2
Servo motor	2
Rain sensor	1
MQTT service of CloudAMQP	
AMQP service of CloudAMQP	
HTTP service of adafruit io	

Table 1 lists all the devices and services that are used in the project.

In addition, an email notification service was developed that uses a Raspberry Pi3 device to collect data from a rain sensor. The collected data are then transmitted to the Adafruit IO cloud API service, which calculates and triggers an event that prompts Zapier to send an email message to the designated recipient. For transmitting optical sensor data in JSON format, the MQTT protocol is utilized. The NodeMCU device sends these data, which are received and processed by the Raspberry Pi 3. Acting as a processing and calculation unit, the Raspberry Pi coordinates the rotation of the solar panel, optimizing its alignment with sunlight. To take advantage of the message queue functionality of the AMQP protocol, a queue is created to store the status information of the pin plate navigation module. This application structure allows for dynamic changes to this status information, enabling the suspension or resumption of the system's operations as needed.

3.2 Evaluation of Applied Protocols

Over time, the HTTP API has a long history of continuous development, offering various features that make it a popular and widely used protocol. However, in recent years, the MQTT protocol has gained rapidly credibility and has proven to be highly applicable in many scenarios. Additionally, the AMQP protocol's Broker model, along with its Queue Message technique, ensures efficient and secure data transfer for specific types of application. Consequently, developers now have more suitable options for their ideas, not limited to IoT projects alone.

Table 2 Comparison among HTTPS, MQTT, and AMQP protocols

	HTTPS	MQTT	AMQP
Abstraction	Request/Reply	Pub/Sub	Pub/Sub
Architecture	P2P	Brokered	P2P or Brokered
QoS	Provided by TCP	3	3
Transport protocol	TCP	TCP	TCP
Data Serialization	No	Undefined	AMQP type system or user defined
Security	SSL/TLS	SSL	TLS

All three protocols mentioned above can be employed for data transport applications, falling within the Application layer of the TCP/IP model. They use the TCP protocol to ensure accurate and reliable data transfer to the intended recipients. Operating in a client–server model, these protocols remain independent of the hardware components, operating systems, programming languages, and technologies used, offering versatility and ease of implementation. The comparison among those applied protocol are shown in Table 2.

3.2.1 Protocols Architecture

Figure 7 shows the operational architecture of the HTTP protocol. In the HTTP protocol, tasks must be executed sequentially, moving from one step to the next, and involve interactions with various software services and server communications. The operation revolves only around the request and the response. Therefore, if any point in the structure is interrupted or an error occurs, subsequent processes are not executed, leading to potential delays and inefficiencies. The HTTP protocol's reliance on multiple layers at the application layer contributes to a slower speed, as it requires traversing through various layers to achieve the final result.

Figure 8 illustrates the operational architecture of the MQTT and AMQP protocols. As indirect message transporters, all systems only need to receive data from the broker. This enables services to receive data from a single source and then process tasks without the need to follow a sequential model like the one used by the HTTP protocol. By adopting this distributed model, the system's operation speed increases as devices simultaneously receive data from a centralized source. This approach also improves system performance, as each component is responsible for specific functions. In case a service point fails or malfunctions, the remaining components continue to operate normally, making

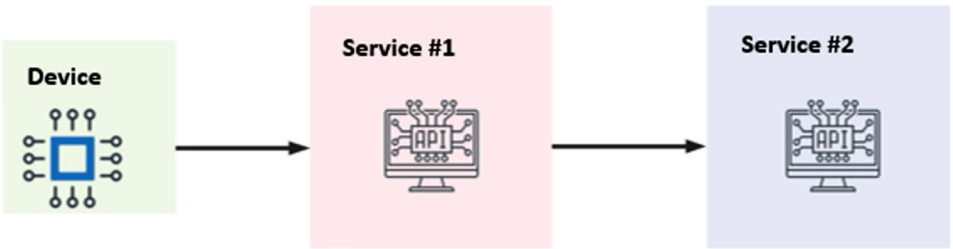

Fig. 7 Operation architecture of HTTP protocol

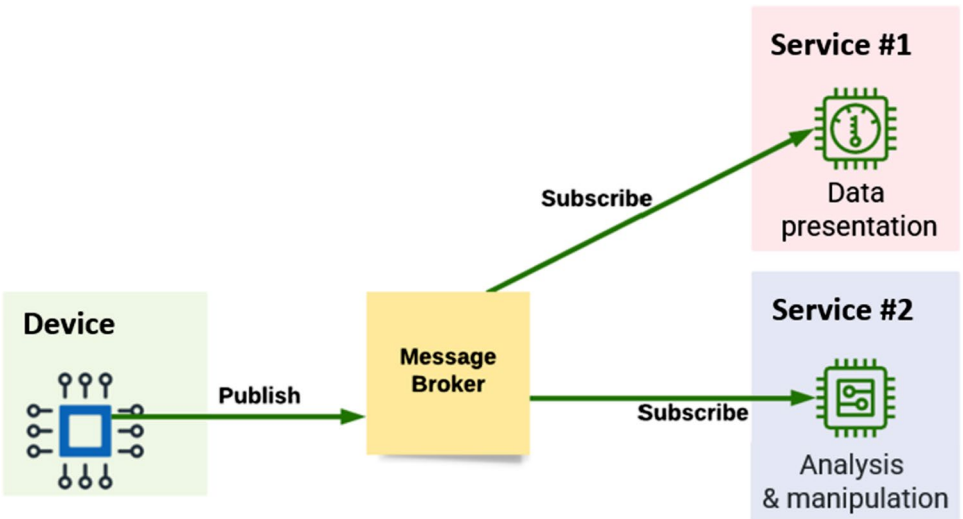

Fig. 8 Operation architecture of MQTT and AMQP protocols

system maintenance more manageable. The decentralized nature of the MQTT and AMQP protocols leads to better efficiency, resilience, and fault tolerance in IoT applications.

3.2.2 Design Characteristics, Reliability and Interoperability

Based on measurements performed in the 3 G network, the MQTT protocol demonstrates significantly higher throughput compared to HTTP, being approximately 93 times faster. This efficiency can be attributed to MQTT's lightweight design, where data are transmitted as a byte array. The message header in MQTT comprises only 2 Bytes, while the message content can be as large as 256 Megabytes. Similarly, AMQP also provides a byte-by-byte data transfer mechanism with an 8 Bytes header capacity, whereas HTTP protocol is known for its support of very long header and message content editing.

Regarding the trust attribute, the MQTT protocol is ranked highest due to its reliable and efficient data transmission capabilities. However, it has the lowest interoperability compared to HTTP. MQTT is primarily designed for Publish/Subscribe communication, which may not cover all use cases in the diverse IoT field. On the other hand, the HTTP protocol, especially when integrated with RESTful technology, is well suited for strong interaction on web services and is known for its ease of client–server interaction.

Interoperability remains crucial for all types of IoT field protocol. The limited scope of MQTT in communication patterns and message formats may pose challenges in certain IoT scenarios. On the contrary, the AMQP protocol utilizes serialization techniques such as Protocol Buffers, Message Packs, and JSON for data transmission design, providing more flexibility to handle diverse data formats.

In summary, the MQTT protocol stands out for its speed and trustworthiness, while the HTTP protocol excels in interoperability and strong web service interaction. Each protocol has its strengths and weaknesses, making it essential for developers to choose the most suitable protocol based on the specific requirements of their IoT projects.

3.2.3 Energy Consumption and Resource Requirements

The MQTT protocol is purposefully designed for efficient bandwidth consumption and minimal utilization of device resources, making it highly suitable for 8-bit controllers with limited memory, typically around 100 Bytes. However, the AMQP protocol requires a slightly higher level of resource capacity due to the inclusion of additional activities aimed at ensuring reliability and redundancy. In comparison, the HTTP protocol requires a larger source of energy and resources to perform the same actions as MQTT and AMQP, making it less favorable for resource-constrained IoT devices.

3.2.4 Bandwidth and Latency

Among the three protocols mentioned above, the HTTP protocol exhibits the highest bandwidth usage and latency, followed by the AMQP protocol and then the MQTT protocol. Latency and bandwidth requirements are significantly influenced by the use of the TCP protocol, which unfortunately does not support improved latency. During the initial stages of connection establishment, TCP does not fully utilize the available network bandwidth, as it adopts a cautious approach to avoid network congestion. As a consequence, the TCP packet sender gradually increases the congestion level and doubles the number of packets per signal transfer round over time. This behavior can contribute to higher latency and increased bandwidth usage when using the HTTP and AMQP protocols, affecting overall performance.

3.2.5 Used in IoT, M2M and Standardization

While the MQTT protocol enjoys widespread usage among various organizations, it has not yet reached the status of a global standard. On the other hand, HTTP is a well-established global Web standard protocol, although its suitability for the IoT industry is somewhat limited. MQTT, being a machine-to-machine connection establishment protocol, is widely supported by renowned organizations such as IBM, Facebook, Cisco, RedHat, and Amazon Web Service (AWS).

Additionally, AMQP stands out as one of the most successful protocols in the IoT field, finding application in major projects such as the Oceanographic Monitoring Project in the mid-Atlantic mountains and NASA's Nebula Cloud Platform. In particular, AMQP is organized by the Organization for the Advancement of Structured Information Standards (OASIS) and is recognized as an international standard (ISO/IEC 19464: 2014).

In summary, the use of the HTTP protocol in the IoT domain is limited due to its heavy and slow performance. MQTT is emerging as a promising and practical protocol for IoT applications and is gaining traction among various organizations. Meanwhile, the AMQP protocol stands as a well-established and successful choice, being an international standard for IoT deployments.

3.3 Analyze the Obtained Results

We utilized Wireshark software for packet capture and analysis to study the behavior of each type of protocol. Wireshark provides a powerful toolkit that enables us to

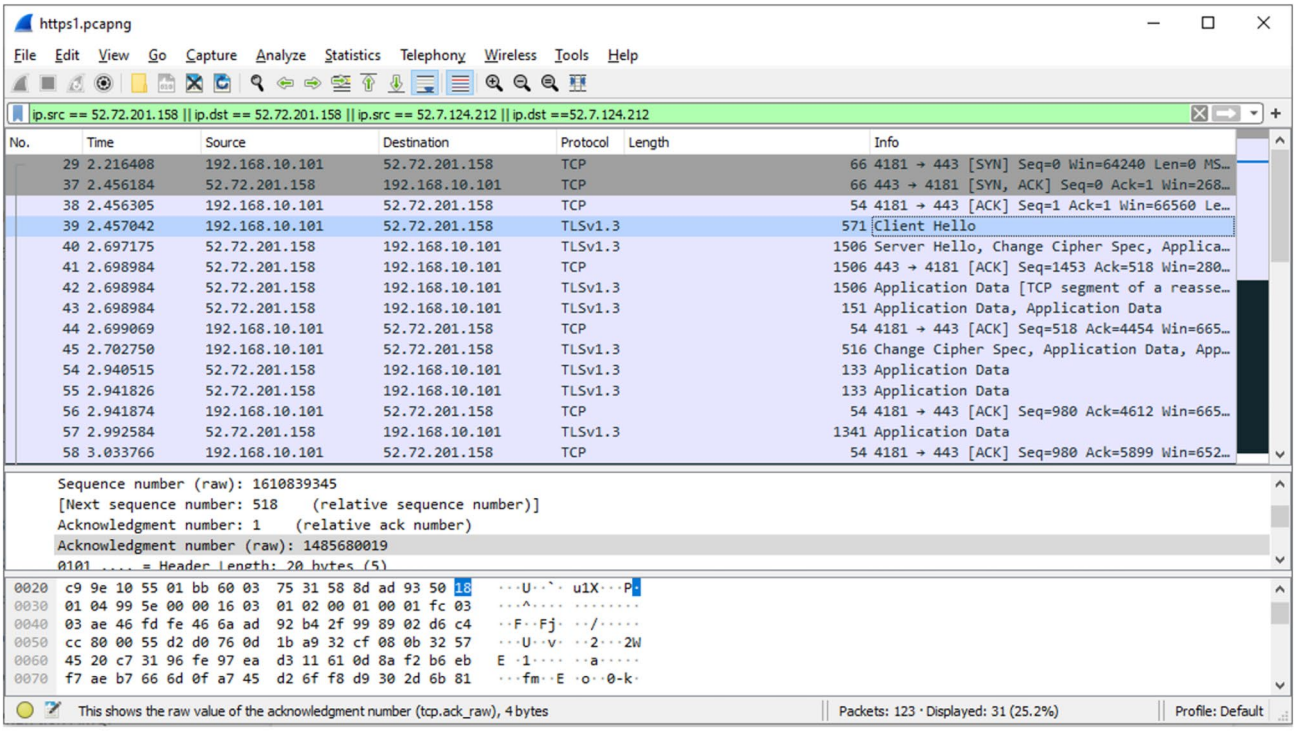

Fig. 9 Packets exchanged over HTTPS protocol

examine packets exchanged between devices and servers in real time. By capturing and inspecting these packets, we gained valuable information on the communication patterns, message structures, and efficiency of the HTTP, MQTT, and AMQP protocols. Wireshark allowed us to visualize the headers and contents of the packet, providing a detailed view of the data transmission process for each protocol. Through this analysis, we were able to identify the key factors that influence the speed, bandwidth usage, and latency of the protocols, allowing us to make comparisons and make informed evaluations. Figures 9, 10, and 11 show the packet analysis results of the three protocols HTTPS, MQTT, and AMQP, respectively. These results were captured and examined using Wireshark software.

Based on Fig. 9, the HTTPS protocol follows the following steps during communication: - Three-way handshake event. - Client greeting with TLSv1.3 security and acknowledgment from the server. - The server sends greetings and receives acknowledgment. - Confidential Information Exchange. - Data Exchange. For the data sample provided, the exchanged packets are encrypted, making it impossible to view their content without decryption on the server. The time taken to complete a request-response process is calculated as 3.033766 – 2.216408 = 0.817358 s. This duration is slightly longer due to the addition of key-exchange steps and an increase in packet size. However, the advantage is that the security of the system is ensured.

When a client connects to the server using the MQTT protocol, it must undergo the three-way handshake and user authentication process. Once the client is authenticated, it can start receiving signals from the MQTT Broker as soon as a message is published. Referring to Fig. 10, the server continuously sends messages to our machine at a frequency of every half second. The delivery of a 94-bytes packet takes only 0.039863 s, calculated as 0.584867 - 0.545004 s. This high speed allows the system enough time to efficiently handle tasks such as navigating the solar panels or manipulating intensity information on the program display.

Based on the packet capture of the AMQP protocol's operation, shown in Fig. 11, we can calculate the operation times as follows: - Complete connection setup, including three-way handshake, Connection, Channel, Queue, and Consume initiation. The duration is temporarily calculated as: 4.741508 - 3.832052 = 0.909456 s. - Obtaining the Basic.Get system status signal takes: 4.828093 - 4.742299 = 0.085794 s. - Publishing a data packet takes: 4.913548 - 4.828748 = 0.0848 s. - Compliment of message delivery during consumption with the Basic.Deliver package: 4.958620 - 4.916574 = 0.042046 s.

The package exchange times of the three protocols include five phases from connection establish, authorization,

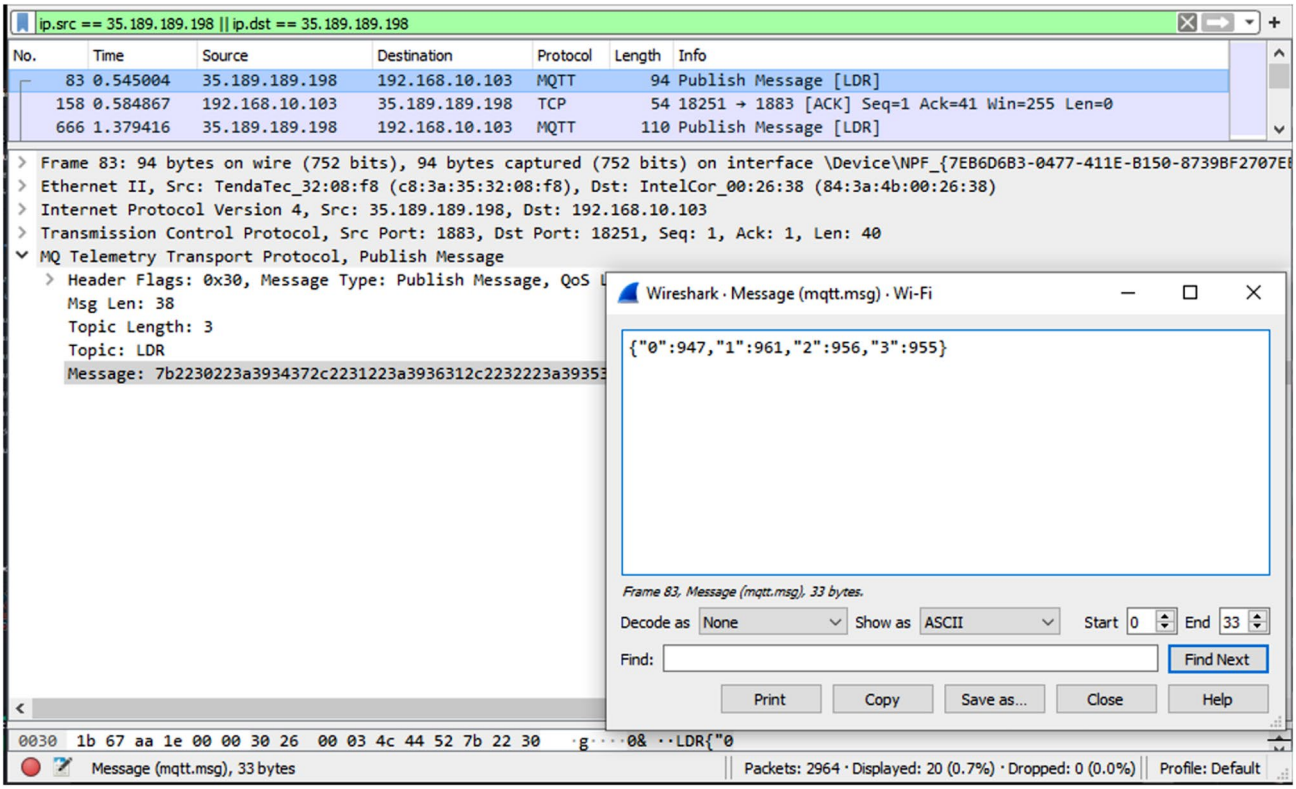

Fig. 10 Analyze the packet received by the client from the Broker MQTT

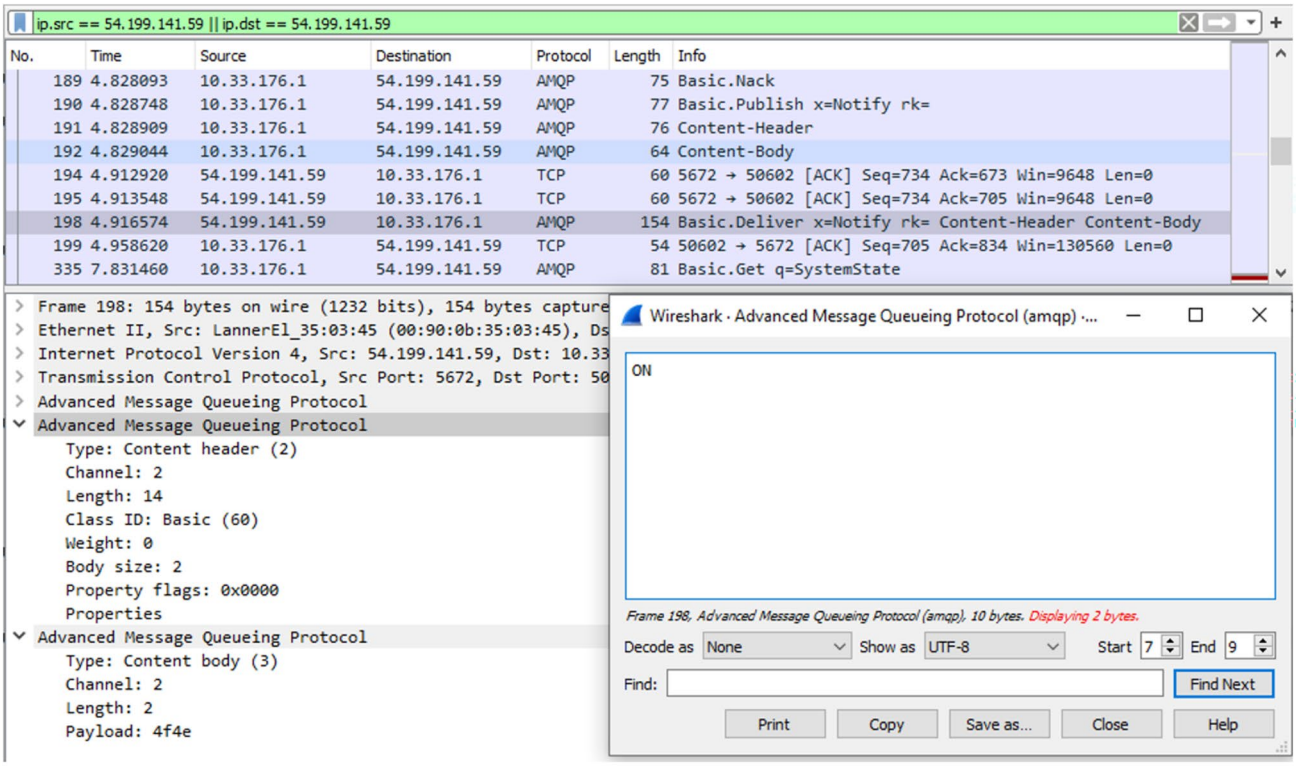

Fig. 11 Analyze Consumer packets from Broker AMQP

configuration, sending data, and receiving response. In Fig. 12, it becomes clear that the MQTT protocol exhibits a faster speed than the other protocols during the connection establishment and data transmission phases. However, the AMQP protocol involves more data due to the requirement of declaring numerous setup settings before sending and receiving messages, making its communication more optimized. For the HTTP protocol, each data request requires sending a bulky header and requires authentication with every request. Consequently, in terms of actual transport speed, MQTT outperforms both AMQP and HTTP, with AMQP trailing behind MQTT and HTTP in descending order of speed.

The speed of operation of the three protocols is also influenced by their respective message design characteristics. Table 3 presents the Header and Body sizes of each protocol packet. It is evident that, while the data transfer may be relatively small, the HTTP protocol includes a significant portion of its packet size dedicated to the header, and the packets sent from the server are also large due to adherence to API rules. In contrast, the remaining two protocols adopt a minimalist packet design, and message transport does not generate additional data. This streamlined approach contributes to the overall efficiency and speed of the MQTT and AMQP protocols.

Table 3 Header and Body sizes of each protocol packet

Protocol	Header of sent msg	Body of sent msg	Header of received msg	Body of recieved msg
HTTP	337	25	1043	194
MQTT	2	41	2	41
AMQP	22	11	22	11

4 Conclusions and Future Works

In summary, this paper has provided an overview and evaluation of three widely used and well-known protocols. HTTP, MQTT, and AMQP, followed by their practical application in real projects. The comparison between these protocols revealed some similarities in characteristics. To gain a complete understanding of their strengths and weaknesses, a deeper analysis was conducted based on specific criteria. By reevaluating their ranking for each criterion, a direct and concise evaluation was achieved, allowing a more informed selection decision for IoT projects and systems. The study has presented a comprehensive comparative picture of these protocols in data transport applications, supported by empirical evidence from actual projects. However, it is essential to acknowledge that these protocols are subject to rapid evolution and may gain support in other fields, potentially leading to changes in their features and functionalities in the future. Consequently, there will be opportunities for further analysis, evaluation, and realistic judgments about these protocols in the ever-evolving IoT ecosystem. In conducting this

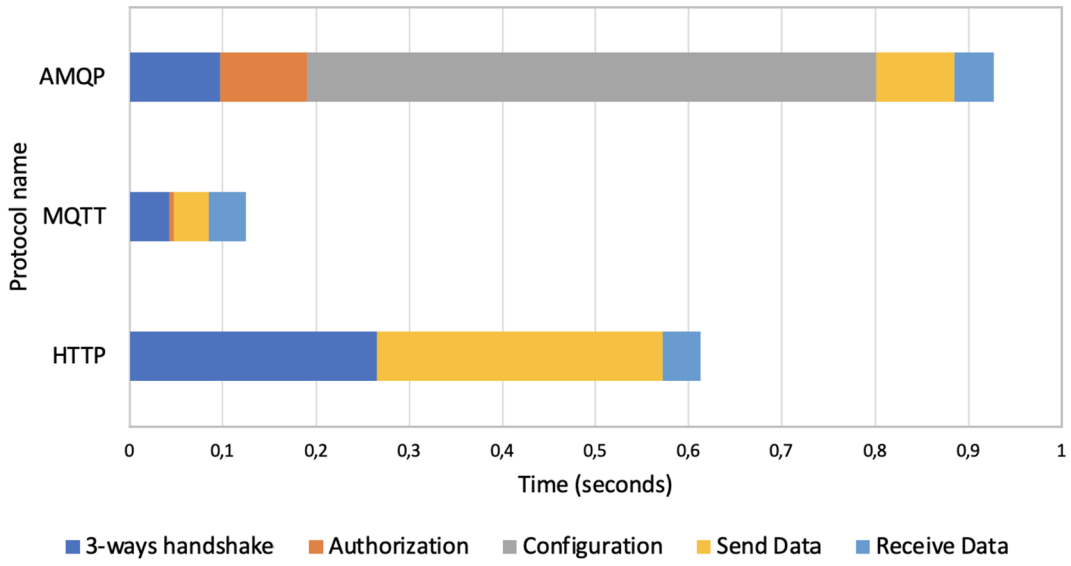

Fig. 12 Timeline of packet exchange

research, we have provided valuable information on the strengths and limitations of HTTP, MQTT, and AMQP in various IoT scenarios. As the IoT landscape continues to evolve, researchers and developers will have the opportunity to revisit and re-assess these protocols to ensure optimal choices for their specific IoT systems and applications.

In our future endeavors, our goal is to evaluate other metrics and adjust the settings of each protocol to suit various network environments. Furthermore, it will be beneficial to observe the behavior of these protocols under higher performance conditions in specific IoT systems.

Author Contributions APN and NNP contributed to the design of the system and the communication of data. KTTM and PDT wrote the article and performed the final correction.

Funding The authors declare that no funds, grants, or other support was received during the preparation of this manuscript.

Data Availability The experiment data used to support the findings of this study are available on request from the corresponding author.

Code Availability Project code is available from the corresponding author on request.

Declarations

Conflict of Interest The authors declare that there are no conflicts of interest with respect to the publication of this paper.

Ethical Approval This research work does not involve human and/or animal subjects. Traffic signs are collected from street cameras. The model is built on training data and has been tested through the test data set.

Consent to Participate All authors have agreed to participate to contribute to the project.

Consent for Publication All authors have read and agreed to the published version of the manuscript.

References

1. Shah SH, Yaqoob I (2016) A survey: Internet of things (iot) technologies, applications and challenges. In: 2016 IEEE Smart Energy Grid Engineering (SEGE), pp 381–385. https://doi.org/10.1109/SEGE.2016.7589556
2. Balaji S, Nathani K, Santhakumar R (2019) Iot technology, applications and challenges: a contemporary survey. Wirel Pers Commun 108(1):363–388. https://doi.org/10.1007/s11277-019-06407-w
3. Li M, Gu W, Chen W, He Y, Wu Y, Zhang Y (2018) Smart home: architecture, technologies and systems. Procedia Comput Sci 131:393–400. https://doi.org/10.1016/j.procs.2018.04.219
4. Khoa TTM, Minh NCA, Hau NT (2021) Internet of things enables real time smart home monitoring system. J Sci Technol 50(2):257–267
5. Dhanvijay MM, Patil SC (2019) Internet of things: a survey of enabling technologies in healthcare and its applications. Comput Netw 153:113–131. https://doi.org/10.1016/j.comnet.2019.03.006
6. Senoo EEK, Akansah E, Mendonça I, Aritsugi M (2023) Monitoring and control framework for iot, implemented for smart agriculture. Sensors. https://doi.org/10.3390/s23052714
7. Gupta P, M IOP (2021) A survey of application layer protocols for internet of things. In: 2021 International Conference on Communication Information and Computing Technology (ICCICT), pp 1–6. https://doi.org/10.1109/ICCICT50803.2021.9510140
8. Kraijak S, Tuwanut P (2015) A survey on iot architectures, protocols, applications, security, privacy, real-world implementation and future trends. In: 11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015), pp 1–6. https://doi.org/10.1049/cp.2015.0714
9. Moraes T, Nogueira B, Lira V, Tavares E (2019) Performance comparison of iot communication protocols. IEEE Press, pp 3249–3254. https://doi.org/10.1109/SMC.2019.8914552
10. Uy NQ, Nam VH (2019) A comparison of amqp and mqtt protocols for internet of things. In: 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), pp 292–297. https://doi.org/10.1109/NICS48868.2019.9023812
11. Yassein MB, Shatnawi MQ, Aljwarneh S, Al-Hatmi R (2017) Internet of things: Survey and open issues of mqtt protocol. In: 2017 International Conference on Engineering & MIS (ICEMIS), pp 1–6. https://doi.org/10.1109/ICEMIS.2017.8273112
12. Mansour M, Gamal A, Ahmed AI, Said LA, Elbaz A, Herencsar N, Soltan A (2023) Internet of things: a comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions. Energies. https://doi.org/10.3390/en16083465
13. Yudidharma A, Nathaniel N, Gimli TN, Achmad S, Kurniawan A (2023) A systematic literature review: messaging protocols and electronic platforms used in the internet of things for the purpose of building smart homes. Procedia Comput Sci 216:194–203. https://doi.org/10.1016/j.procs.2022.12.127
14. Krishna CS, Sasikala T (2019) Healthcare monitoring system based on iot using amqp protocol. In: Smys S, Bestak R, Chen JI-Z, Kotuliak I (eds) International Conference on Computer Networks and Communication Technologies. Springer, Singapore. pp 305–319
15. Wang H, Xiong D, Wang P, Liu Y (2017) A lightweight xmpp publish/subscribe scheme for resource-constrained iot devices. IEEE Access 5:16393–16405. https://doi.org/10.1109/ACCESS.2017.2742020
16. Wang V, Salim F, Moskovits P (2013) Using messaging over WebSocket with STOMP. Apress, Berkeley, CA, pp 85–108. https://doi.org/10.1007/978-1-4302-4741-8_5
17. Saputro AK, Anditya AR, Ulum M, Sukri H, Alfita R, Ibadillah AF (2020) Application of lora (long range access) in optimizing internet of things using mqtt (message queuing telemetry transport) for fish feed monitoring. In: 2020 6th Information Technology International Seminar (ITIS), pp 224–228. https://doi.org/10.1109/ITIS50118.2020.9321021

18. Pierleoni P, Concetti R, Marzorati S, Belli A, Palma L (2023) Internet of things for earthquake early warning systems: a performance comparison between communication protocols. IEEE Access 11:43183–43194. https://doi.org/10.1109/ACCESS.2023.3271773
19. Gemirter CB, Şenturca Çağatay, Baydere Şebnem (2021) A comparative evaluation of amqp, mqtt and http protocols using real-time public smart city data. In: 2021 6th International Conference on Computer Science and Engineering (UBMK), pp 542–547. https://doi.org/10.1109/UBMK52708.2021.9559032
20. Moraes T, Nogueira B, Lira V, Tavares E (2019) Performance comparison of iot communication protocols. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), pp 3249–3254. https://doi.org/10.1109/SMC.2019.8914552
21. Mishra B, Kertesz A (2020) The use of mqtt in m2m and iot systems: a survey. IEEE Access 8:201071–201086. https://doi.org/10.1109/ACCESS.2020.3035849
22. Dobbelaere P, Sheykh Esmaili K (2017) Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: industry paper, pp 227–238. https://doi.org/10.1145/3093742.3093908
23. Hunkeler U, Truong HL, Stanford-Clark A (2008) Mqtt-s – a publish/subscribe protocol for wireless sensor networks. In: 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08), pp 791–798. https://doi.org/10.1109/COMSWA.2008.4554519
24. Naik N (2017) Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In: 2017 IEEE International Systems Engineering Symposium (ISSE), pp 1–7. https://doi.org/10.1109/SysEng.2017.8088251
25. Fernandes JL, Lopes IC, Rodrigues JJPC, Ullah S (2013) Performance evaluation of restful web services and amqp protocol. In: 2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN), pp 810–815. https://doi.org/10.1109/ICUFN.2013.6614932
26. Krishna C, Sasikala T (2019) Healthcare Monitoring System Based on IoT Using AMQP Protocol. In: ICCNCT 2018:305–319. https://doi.org/10.1007/978-981-10-8681-6_29