TẠP CHÍ KHOA HỌC ĐẠI HỌC TÂN TRÀO

ISSN: 2354 - 1431

http://tckh.daihoctantrao.edu.vn/

COAP AND MQTT PROTOCOLS IN THE INTERNET OF THINGS

Thuy Dung Nguyen, Thi Nguyet Vu

[1]Tan Trao University, Vietnam

Email address: vtnguyet@ictu.edu.vn

https://doi.org/10.51453/2354-1431/2024/1147

Article info

Received:12/5/2024

Revised: 15/6/2024

Accepted: 25/8/2024

Keywords:

MQTT, CoAP, application layer protocols, Internet of Things, IoT, wireless sensor network.

Abstract:

Alongside the explosion and development of technology, the Internet of Things (IoT) - the network that connects everything to the Internet - can be considered the trend and future of the world. For devices to communicate with each other, they will require one or more protocols, which can be seen as specialized languages to solve specific tasks. This paper aims to survey the two most promising protocols for the application layer in the IoT network: MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol). Based on research results on architecture, transmission methods, publish/subscribe mechanisms, response/acknowledgment, security, and Quality of Service (QoS), a comparison between the two protocols will be conducted, followed by observations on their application scopes for IoT designs.

TẠP CHÍ KHOA HỌC ĐẠI HỌC TÂN TRÀO

ISSN: 2354 - 1431

http://tckh.daihoctantrao.edu.vn/

GIAO THỨC COAP VÀ MQTT TRONG MẠNG LƯỚI KẾT NỐI VẠN VẬT

Nguyễn Thuỳ Dung, Vũ Thị Nguyệt

[1]Đại học Tân Trào, Việt Nam

Địa chỉ email: vtnguyet@ictu.edu.vn

https://doi.org/10.51453/2354-1431/2024/1147

Thông tin bài viết

Ngày nhận bài: 12/5/2024

Ngày sửa bài: 15/6/2024

Ngày duyệt đăng: 25/8/2024

Từ khóa:

MQTT, CoAp, giao thức ứng dụng, Internet vạn vật, IoT, mạng cảm biến không dây.

Tóm tắt

Bên cạnh sự bùng nổ và phát triển của công nghệ thì Internet of Things (IoT) – mạng lưới vạn vật kết nối internet có thể coi là xu hướng và tương lai của thế giới. Để các thiết bị có thể giao tiếp với nhau, chúng sẽ cần một hoặc nhiều giao thức, có thể xem là một thứ ngôn ngữ chuyên biệt để giải quyết một tác vụ nào đó. Bài báo sẽ tiến hành khảo sát hai giao thức triển vọng nhất cho lớp ứng dụng trong mạng IoT là MQTT (Message Queuing Telemetry Transport) và CoAP (Constrained Application Protocol). Dựa trên các kết quả nghiên cứu về kiến trúc, phương thức truyền tải, cơ chế xuất bản/đăng ký, phản hồi/đáp ứng, độ bảo mật, Chất lượng dịch vụ (QoS) để tiến hành so sánh hai giao thức với nhau và cuối cùng đưa ra nhận xét về phạm vi ứng dụng của chúng cho các thiết kế IoT..

1. Introduction

The Internet of Things (IoT) is a scenario in which every object, including humans, is provided with a unique identifier and the ability to transmit and exchange information and data over a single network without the need for direct interaction between humans or human-computer interaction. IoT has evolved from the convergence of wireless technology, microelectronics, and the Internet. In simple terms, it is a collection of interconnected devices with the capability to connect to each other, to the Internet, and to the external world to perform specific tasks.

The basic architecture of IoT consists of three main layers: the Application Layer, the Network Layer, and the Perception Layer.

The Application Layer provides the interface between user applications and the lower layers through which messages are transmitted. Application layer protocols are commonly used to exchange data between programs running on source and destination devices. This layer serves as a gateway for processing application programs to access network services. It represents direct support for user applications, such as message switching software, database access, email, etc. IoT requires standardized protocols for this layer.

Two of the most prominent and widely used protocols today are MQTT and CoAP, which share the following characteristics:

• They are open standards.

• They are more suitable for constrained environments compared to HTTP.

• They provide asynchronous message transmission mechanisms.

• They run on the Internet Protocol (IP).

This paper covers relevant aspects of these promising protocols and conducts a comparison between them, concluding with observations on their application scope for IoT designs.

2. CoAP Protocol

Architecture: The CoAP protocol was developed by the Constrained RESTful Environments (CoRE) research group, as an application layer protocol for IoT networks. CoAP is defined as a REST-based web transfer protocol, which is commonly used for communication between computers (such as personal computers and web servers) in managing resources on the internet. CoAP is designed to be compatible with HTTP and large-scale RESTful webs through simple proxies. CoAP operates in a client/server model, where clients perform GET, PUT, POST, and DELETE methods on resources. CoAP is based on datagram-based data packaging and can be used at the datagram header and other packet data transmission protocols.

CoAP enables small, low-power devices to perform computation, exchange information, and interact in RESTful applications. CoAP can be divided into two sublayers: the message layer and the request/response layer. The message layer detects duplicates and provides reliable communication over UDP transport layers since UDP does not have built-in error recovery mechanisms. The request/response layer handles REST connections. CoAP uses four types of messages: confirmable, non-confirmable, reset, and acknowledgment. The reliability of CoAP is achieved through the combination of confirmable and non-confirmable messages. Separate responses are used when the server needs to wait for a specific time before responding to the client. In non-confirmable mode, the client sends data without waiting for an ACK message. The server responds with a reset message (RST) when the message is erroneous or encounters issues in communication.

Flexibility - Content Negotiation: Similar to HTTP, CoAP is a document transfer protocol, but it is designed for constrained devices. CoAP packets are much smaller compared to HTTP/TCP messages. Bit fields and integer mappings are widely used to save space. The simplicity of CoAP packets allows for in-place generation and parsing without requiring additional RAM in constrained devices. CoAP also supports content negotiation like HTTP. The client uses the Accept option to indicate the desired representation of a resource, and the server responds with the Content-Type option to inform the client about its available representations. Similar to HTTP, this enables independent evolution of the client and server, allowing the addition of new resources without affecting each other. CoAP requests can utilize query strings in the form of ?a=b&c=d. This can be used to provide search capability, pagination, and other features for clients.

Security: CoAP runs on UDP (User Datagram Protocol) rather than TCP (Transmission Control Protocol). Clients and servers communicate through connectionless packets. By eliminating TCP, full IP networking can be achieved on small microcontrollers. CoAP allows the use of UDP broadcast and multicast communication. Since it does not run on UDP, SSL/TLS (Secure Sockets Layer/Transport Layer Security) is not available for providing security. DTLS (Datagram Transport Layer Security) provides similar security guarantees as TLS but for transmitting

data over UDP. DTLS-enabled devices will support encryption algorithms such as RSA (Rivest-Shamir-Adleman) and AES (Advanced Encryption Standard) or ECC (Elliptic Curve Cryptography) and AES.

Network Address Translation (NAT) Issues: In CoAP, a sensor node is often a server, not just a client, although it can be both. Sensors or actuators provide resources that can be accessed by clients to read or modify their state. When sensors act as servers, they need to be able to receive incoming packets. To function correctly behind NAT, the first device needs to send a request to the server, as done in Lightweight M2M (LWM2M), allowing routers to establish the connection. Although CoAP does not require IPv6, it is the simplest protocol to use in an IP environment where devices are directly routed.

Some important features of CoAP:

• Resource Observation: Allows for registering to monitor desired resources using a publish/subscribe mechanism.

• Block-wise Transfer: Enables the exchange of large data between client and server without requiring full data updates, reducing communication costs.

• Resource Discovery: Servers use the CoRE link format based on web links in URI paths to discover resources for clients.

• Interaction with HTTP: The flexibility of CoAP allows for easy interaction with HTTP through proxies, making it compatible with various devices using common REST architectures.

• Security: CoAP is built on top of Datagram Transport Layer Security (DTLS) to ensure message integrity and security.

3. MQTT Protocol

Architecture: MQTT is a publish/subscribe protocol used for low-bandwidth, high-reliability IoT devices, and is suitable for unstable networks. MQTT follows a client/server model, where each sensor is a client that connects to a server (broker) using the Transmission Control Protocol (TCP). MQTT is a message-oriented protocol, where each message is a discrete piece of data, and the broker is not aware of its contents. Each message is published to an address, which can be seen as a channel. Clients subscribe to one or more channels to receive/send data, known as subscriptions. Clients can subscribe to multiple channels, and they receive data when any other station publishes data to the subscribed channels. When a client sends a message to a channel, it is called publishing.

Quality of Service (QoS): In the MQTT protocol, there are three QoS options for publishing and subscribing:

• QoS0: At most once delivery: The broker/client will deliver the message once.

• QoS1: At least once delivery: The broker/client ensures that the message is delivered at least once, requiring at least one acknowledgment from the endpoint, which may result in multiple acknowledgments for the same message.

• QoS2: Exactly once delivery: The broker/client guarantees that the message is delivered exactly once, with the transmission process confirmed by the TCP/IP protocol.

A message can be sent with any Quality of Service (QoS), and clients can also subscribe with any desired QoS requirement. This means that the client will select the maximum QoS it can receive messages with.

Last Will and Testament (LWT): MQTT clients can register a custom message to be sent by the broker if the client disconnects unexpectedly. These messages can be used to notify registered subscribers.

Message Retention: MQTT supports storing messages within the broker for message persistence.

When publishing messages, clients can request the broker to retain the messages. Only the latest messages are stored. When a client subscribes to a channel, any stored messages related to that subscription will be sent to the client. Unlike a message queue, MQTT brokers do not allow message persistence to back up to the server.

Security: MQTT brokers can require username and password authentication from clients to establish a connection. To ensure security, the TCP connection can be encrypted using standard security technologies, such as Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols for secure communication between the web server and the browser.

The two main versions of MQTT are MQTT v3.1 and MQTT-SN (formerly known as MQTT-S) V1.2. Although MQTT v3.1 is designed to be lightweight, it has two limitations that restrict devices: each MQTT client must support TCP and typically maintains an open connection to the broker at all times. This can be a challenge for environments with high message loss or limited computing resources. MQTT channel names are often long strings that are not suitable for the 802.15.4 standard. This limitation is addressed by the MQTT-SN protocol, which defines a UDP mapping of MQTT and adds the ability to index channel names for the broker.

4. Comparison between CoAP and MQTT

Both MQTT and CoAP are useful application layer protocols for IoT, but they have fundamental differences:

MQTT is a many-to-many communication protocol for transmitting messages between multiple clients through a central broker. It separates publishers and subscribers by allowing clients to send messages and brokers to decide the routing and duplication of messages. MQTT has some support for message persistence and works well as a direct data bus.

CoAP primarily serves as a one-to-one protocol for transmitting state information between clients and servers. It supports resource observation and is best suited for state transfer models rather than solely event-based ones.

MQTT clients establish a long-lasting TCP connection to the broker, which is not ideal for devices behind NAT. Both CoAP clients and servers send and receive UDP packets.

MQTT does not support message labeling with types or other data to help clients understand them. MQTT messages can be used for any purpose, but all clients must know the message format to enable communication. In contrast, CoAP supports content negotiation and discovery, allowing devices to discover each other for data exchange.

When comparing performance between MQTT and CoAP in terms of end-to-end transmission latency through a shared middleware, research [3] indicates that MQTT has lower latency than CoAP when the packet loss rate is low. Conversely, when the packet loss rate is high, CoAP transmits faster than MQTT. In the case of small-sized messages and a loss rate below 25%, CoAP performs faster than MQTT in generating less overhead. This performance difference can be leveraged to improve network efficiency when choosing the protocol based on current network conditions.

Comparing the two protocols for smartphone sensor applications:

- Qualitative comparison shows that MQTT is more suitable for applications requiring advanced functionality, precise decision-making, and sending messages exactly once. CoAP has limitations in supporting secure Unicast transmission, while MQTT is a better solution when Multicast safety requirements are needed.

- Preliminary analysis shows that CoAP achieves better results in terms of bandwidth usage and round-trip time. CoAP is a suitable choice for

the efficient development of applications using limited resources.

- In terms of reliability, MQTT performs well with good congestion control. However, significant differences are only observed when data needs to be exchanged frequently. For applications that do not require high data transmission frequency, no noticeable difference is seen in reliability between CoAP and MQTT.

According to a survey of Internet protocols using gateway-constrained devices for sharing sensor data with applications [5], the analysis shows that CoAP performs better than MQTT in terms of energy consumption and bandwidth utilization when considering request-response and resource observation requirements.

Summary comparison table of some basic features between the MQTT and CoAP protocols

Protocol	CoAP	MQTT
RESTful	Yes	No
Transport	UDP	TCP
Publish/Subscribe	Yes	Yes
Request/Response	Yes	No
Security	DTLS	SSL/TLS
QoS	Yes	Yes

5. Conclusion

The Internet of Things (IoT) and the proliferation of smart devices, along with technological advancements, have the potential to completely transform human life in the future. With everything being "internet-enabled," users can fully control, monitor, and communicate with these devices from anywhere. Therefore, selecting a suitable protocol for IoT networks is a critical issue for developers and designers. This article surveyed two dominant protocols in the IoT application layer: CoAP and MQTT. Based on the survey results and research on architecture, transmission methods, Publish/Subscribe, Request/Response capabilities, security, and Quality of Service (QoS), the article provides an overview of the two protocols, each with its own advantages and disadvantages. The choice of protocol depends on the specific application and constraints. While CoAP is designed for interacting with websites, MQTT utilizes low bandwidth in high-latency environments, making it ideal for Machine-to-Machine (M2M) applications.

In the future, to validate the suitability of each protocol for different applications, the research team will continue to execute experiments to provide empirical results.

REFERENCES

Ala Al-Fuqaha; Mohsen Guizani; Mehdi Mohammadi; Mohammed Aledhari; Moussa Ayyash, *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications*, IEEE Communications Surveys & Tutorials 2015, Pages: 2347 – 2376.

Dinesh Thangavel; Xiaoping Ma; Alvin Valera; Hwee-Xian Tan; Colin Keng-Yan Tan, *Performance evaluation of MQTT and CoAP via a common middleware*, IEEE Conference Publications, Pages: 1 – 6, 2014.

N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, *Comparison of two lightweight protocols for smartphone-based sensing*, inProc. IEEE 20th SCVT, 2013.

S. Bandyopadhyay, A. Bhattacharyya, *Lightweight Internet protocols for web enablement of sensors using constrained gateway devices*, International Conference on Computing, Networking and Communications (ICNC), 2013.

Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, Jesus Alonso-Zarate, *A Survey on Application Layer Protocols for the Internetof Things*, Transactionon IoT and Cloud Computing, 2015