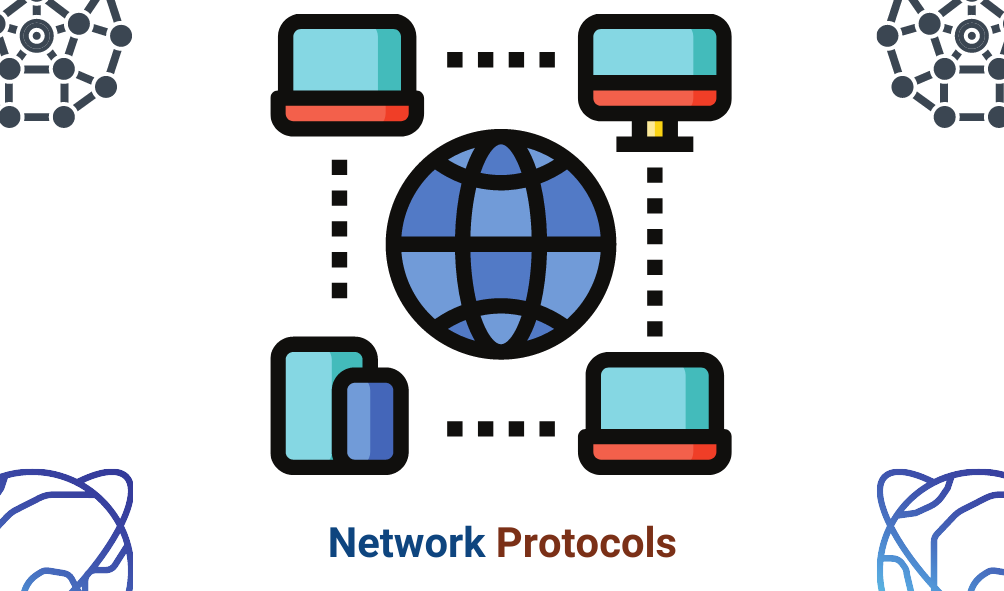
Communication protocols for IoT

By July Trendafilov

St. nr.: 4674693



Contents

[Short Introduction 3](#_Toc114053073)

[Overview of the protocols 3](#_Toc114053074)

[MQTT 3](#_Toc114053075)

[Overview 3](#_Toc114053076)

[Message transmitting 4](#_Toc114053077)

[Secure connection 4](#_Toc114053078)

[Subscribing/unsubscribing to topic 4](#_Toc114053079)

[Applications 5](#_Toc114053080)

[CoAP 5](#_Toc114053081)

[Overview 5](#_Toc114053082)

[Message Model 5](#_Toc114053083)

[Responses 6](#_Toc114053084)

[Tokens 6](#_Toc114053085)

[Methods 6](#_Toc114053086)

[Security 6](#_Toc114053087)

[Multicast CoAP 6](#_Toc114053088)

[Applications 6](#_Toc114053089)

[Use cases 7](#_Toc114053090)

[Conclusion 8](#_Toc114053091)

[Appendix A 8](#_Toc114053092)

[Bibliography 9](#_Toc114053093)

# Short Introduction

In embedded systems there is a variety of communication protocols for separate boards to communicate with each other. This report is focused on IoT and the communication with two of the heavily used protocols in Embedded systems: MQTT and CoAP. The report will clarify how each of these protocols works, how are messages transmitted and some examples where these find application.

# Overview of the protocols

Both protocols are in the Application layer and are used for M2M communication. And both are lightweight protocols. However, they are very different from each other.

## MQTT

### Overview

MQTT is a many-to-many protocol where the communication between clients is done through an MQTT broker a.k.a. as the MQTT server. The usual setup is one MQTT broker with multiple clients. This protocol is publish/subscribe type which means each client subscribes to one or more topics and publishes messages to the broker. The broker then redirects the messages to the subscribed clients. It uses internet connection for remote servers but has an option for local broker.

Diagram

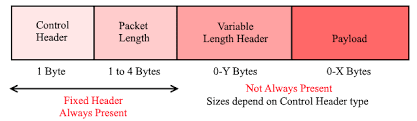
Description automatically generated

### Message transmitting

An MQTT packet consists of header (2B), variable header with options and payload. The maximal size for a single packet is 256MB. To send data, first there must be a connection between the broker and the clients. To do so, the clients request connection with given BROKER CREDIENTIALS (username and password for the broker) and then the broker must verify the connection by either rejecting it or accepting it. In case of approval the server returns CONNACK.

Messages can be sent to the broker and on every message if the broker receives and processes it, it returns PUBACK. However, there are 3 possible ways to send a message with QoS (Quality of Service): 0 – 2; QoS 0 (at most once), QoS 1 (at least once), QoS 2 (exactly once). These three levels are sequenced fastest to slowest. Another important feature is the RETAIN flag. It tells the broker to remember the last sent value. These two variables are kept in the variable header.

Another feature is the Last-Will-Message. This message is sent when the client disconnects from the broker.



### Secure connection

The MQTT protocol relies on TCP and sends the data in plain text. However, the data can be encrypted using TLS with certificates and keys. To have secure connections with the broker, the broker must have a certificate and a key. The client must have a certificate to connect to the specified broker. The server’s key approves of the client’s certificate by checking compatibility. Still there is an option for unsecured connection if there are no certificates.

### Subscribing/unsubscribing to topic

To subscribe to a topic, a client must send a subscribe request to which the server returns SUBACK. To unsubscribe the client must also send request but then the server return UNSUBACK.

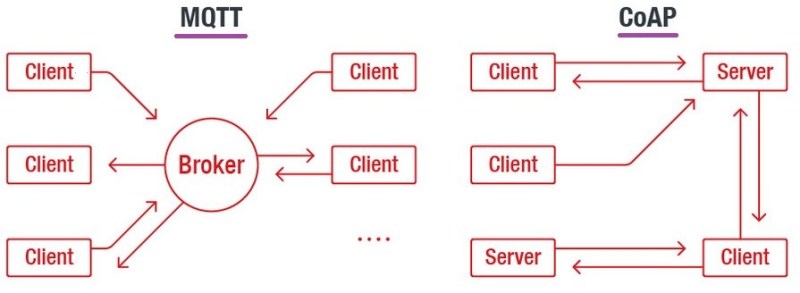
### Applications

Because MQTT is many-to-many connection and encourages live data, this makes it perfect for solutions including automobiles, thermostats, monitoring systems, etc. It has energy efficient. In general, it is great for SCADA (look at [Appendix A](#_Appendix_A)) software. Such are all monitoring systems for data acquisition.

## CoAP

### Overview

In contrast to MQTT, CoAP is a one-to-one communication protocol. It is a client/server model of HTTP. In M2M interactions CoAP acts as server and client at the same time. In terms of usage, CoAP uses the same requests as HTTP but works asynchronous over UDP. It is an HTTP for restricted devices. CoAP uses the internet to connect to other Nodes in a constrained network.



### Message Model

CoAP messages are sent between two endpoints over UDP. The packet has a 4-byte header, optional binary options and an optional payload. Each message contains a Message ID used to detect duplicates and for optional reliability. Reliability is done by marking a message as confirmable. This message is retransmitted after timeout until an ACK with the same Message ID is returned. When a recipient can’t process a Confirmable message, it returns an RST (Reset message) instead of ACK.

Another message type is Non-confirmable (NON). They are not acknowledged but have a Message Id to detect duplicates. When the recipient cannot process the message, it might return RST.

### Responses

There are specific types of responses such as:

* Piggybacked response
  + The recipient returns the Confirmable Message with the ACK and if the ACK is lost, the client will retransmit the request again.
* Separate response
  + The client sends a request, but the server is not ready to respond. To evade extra requests, the server sends back an EMPTY ACK to stop the client from retransmitting and when the Message is ready it sends it to client and the client returns an ACK.

If the message is Non-confirmable, then the server might return either a Non-confirmable message or a Confirmable.

### Tokens

When a packet is sent, a token is also present. It is different from the Message ID because it is used to match a response with a request. It is intended for concurrent requests and especially for differentiating them. Might also be called Request ID. The client MUST generate at all cost a unique token for the given endpoint.

### Methods

Just like the HTTP, CoAP has also GET, POST, PUT, DELETE methods. The payload depends on the Method of the message.

### Security

To secure the connection on UDP, the CoAP must use DTLS for datagram connections. It is certification-based system.

### Multicast CoAP

CoAP supports requests to multicast group. This way the client can discover and join more servers and listen to the default CoAP port. Multicast requests might also be received by multicast addresses.

### Applications

Because CoAP is one to one connection with requests of client/server type, it is not suitable for live data although it has CoAP observe to request regularly the state. It is used for M2M communication and mostly smart energy and building automation.

# Use cases

Those two protocols are IoT and can be used in one practice. However, in different applications they perform differently. Also depends on the use of the application. Some of the use cases are:

1. Smart lighting within the R10 building
   1. Depending on what the control of the system is we could choose both.
      1. If it is manual control, we could use CoAP because we just turn on or off the system, increase the brightness manually and we shouldn’t keep the connection open but rather request connection when we need to change something. This way we safe energy.
      2. If it is light dependent, we could use MQTT because light can change gradually over time and live data will change the brightness according to the light readings. The connection is open because its live data and consumes more energy.
2. Counting wildlife in a nature reserve
   1. This one can only be obtained by live data, and we must use CoAP. To count if new life is born or one ended, we could use CoAP because it is not regular for new life to be born in a wildlife reserve this frequently. It is energy efficient.
3. Air quality reporting in Eindhoven
   1. Depending on the use of this application we could use both
      1. If the environment where these readings are made is dependent on the readings, we must use live data (MQTT) to regulate the air. For example, clean rooms for electrodes and such (HIGH IMPORTANCE)
      2. If it just gives readings to people for the air, CoAP is better because its energy efficient and changes the value only periodically if there is a change and does not need endless connection. Energy efficient.
4. Measuring glacier size
   1. This reading could be used with CoAP and MQTT as well depending on the importance.
      1. If the readings must be precise and time sensitive, MQTT is more suitable.
      2. If we notify once the digits have changed a little more and periodically, we could use CoAP to send the data to the client.

# Conclusion

Both protocols are designed for M2M communication, they are lightweight and have a TLS security. However, there are more differences starting from that they use different transport layer protocols (TCP, UDP). Furthermore, it is proved that CoAP is more energy efficient as it does not keep the connection open at all times as MQTT does. Also, when using Wireshark, it seems that the packets of CoAP are smaller compared to the MQTT.

# Appendix A

"Supervisory control and data acquisition (SCADA) is a control system architecture comprising computers, networked data communications and graphical user interfaces for high-level supervision of machines and processes. It also covers sensors and other devices, such as programmable logic controllers, which interface with process plant or machinery."

*Source: SCADA -* [*https://en.wikipedia.org*](https://en.wikipedia.org/wiki/special:search/SCADA)

"CoAP is a service layer protocol that is intended for use in resource-constrained internet devices, such as wireless sensor network nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity."

*Source: CoAP -* [*https://en.wikipedia.org*](https://en.wikipedia.org/wiki/special:search/CoAP)

"Wireshark is cross-platform, using the Qt widget toolkit in current releases to implement its user interface, and using pcap to capture packets"

*Source: Wireshark -* [*https://en.wikipedia.org*](https://en.wikipedia.org/wiki/special:search/Wireshark)

"MQTT (originally an initialism of MQ Telemetry Transport) is a lightweight, publish-subscribe, machine to machine network protocol. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth. It must run over a transport protocol that provides ordered, lossless, bi-directional connections—typically, TCP/IP."

*Source: MQTT -* [*https://en.wikipedia.org*](https://en.wikipedia.org/wiki/special:search/MQTT)

# Bibliography

Dobbelsteen, J. (n.d.). *Github*. Retrieved from https://git.fhict.nl/technology/t-sem3-db/-/tree/master/communication/Toolbox

Dobbelsteen, J. (n.d.). *GitLab*. Retrieved from https://git.fhict.nl/technology/t-sem3-db/-/blob/master/communication/Challenges/1%20Communication%20protocols%20for%20IOT.pdf

HiveMQ. (n.d.). *HiveMQ*. Retrieved from https://www.hivemq.com/docs/hivemq/4.8/user-guide/introduction.html

Shelby, Z. (2014, June 14). Retrieved from The Constrained Application Protocol (CoAP): https://www.rfc-editor.org/rfc/rfc7252#page-38

Spiess, A. (n.d.). *YouTube*. Retrieved from https://www.youtube.com/watch?v=pfG8uEDZj5g&t=687s

*Wikipedia*. (n.d.). Retrieved from https://www.wikipedia.org