MQTT & CoAP project

Alessandro Bottazzi

July 2024

1 Abstract

The objective of this project is to highlight the **differences between the MQTT and CoAP protocols**, showcasing their distinct natures; MQTT is an event-driven protocol, in contrast with CoAP's nature, which follows a request-response model. The goal is to implement a system where this two different protocols coexist and work together. MQTT has lightweight publish-subscribe architecture, making it highly efficient for scenarios requiring minimal bandwidth and power consumption, such as IoT and M2M communications. CoAP, is designed for constrained devices and networks, using a client-server model similar to HTTP but optimized for low overhead and simplicity.

1.1 From the "abstract" to the concrete

To better understand the characteristics of the individual protocols and the way in which they can operate, a pseudo-realistic structure was created in order to achieve periodic data collection, analysis, and control of actuators, with immediate anomaly notifications. In the following document we will talk about a **remote-controlled greenhouse**.

2 Software and Libraries

The project code is written in Python. The used libraries are: paho-mqtt and aiocoap for MQTT and CoAP protocols implementation.

3 General structure overview

To ensure that attention is not distracted from the main purpose of the project (i.e. an intelligent use of the two protocols), the entities are identically single (entity topic will be discussed later) because he addition of individual entities would increase the complexity of the architecture without however bringing improvements of the project.

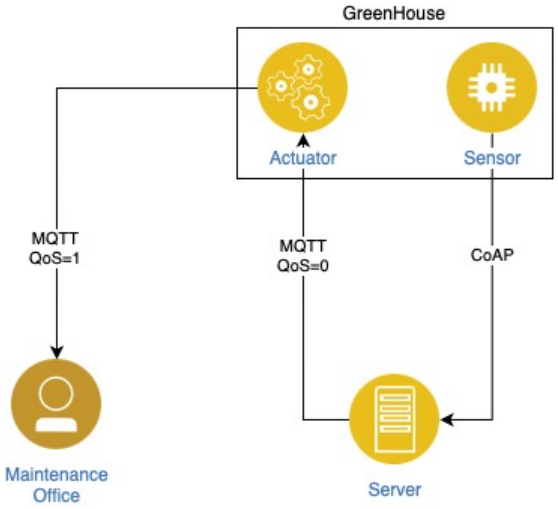

Figure 1: SingleGreenhouse layout example

3.1 Entities

The entities in game are:

- Sensor: it's a CoAP Server because it holds information.
- Actuator: it's an MQTT Subscriber and Publisher node.
- Server: CoAP Client and MQTT Publisher.
- Maintenance office: MQTT Subscriber node. Actuator and Sensor are in the same physical place, Server and the Maintenance office don't necessarily.

The aim of the project is the creation of a system in which there is a central server that analyzes the various data and is able to query sensors and manage actuators, the actuators in turn are equipped with a system for the detection of a possible fault and the possibility of sending a warning signal to the maintenance office. In more detail we are talking about a remotely controlled greenhouse.The sensors must read data such as humidity, temperature, and light, the server must analyze them and control the actuators which are a water pump and a lamp (Fig. 1).

3.2 Requirements

Let's talk about the requirements of the system:

- The server is responsible for reading the data periodically, analyzing it and consequently controlling the actuators. It's the brain of the system.

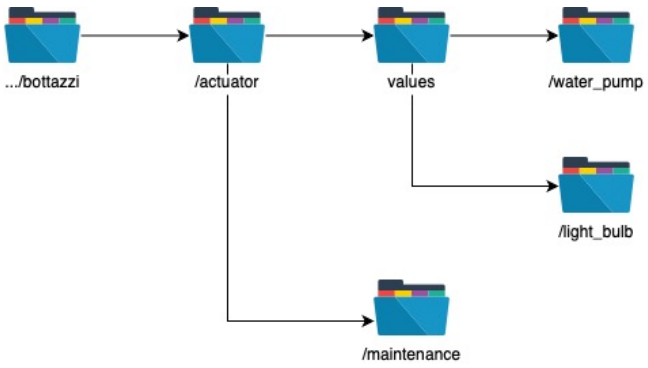

Figure 2: MQTT structure

- Sensors must provide the information to the Server.
- Actuators are able, in case of anomalies, to notify to the maintenance office.

4 More about MQTT and CoAP

Let's now talk about further details that go more specifically into the characteristics of the two different protocols. For MQTT the main **topics** are: maintenance, actuator, the last one is divided into water_pump and light_bulb. the server publishes to *actuator/water_pump* and *actuator/light_bulb* (Fig. 2), "True" or "False" to indicate whether an actuator should be activated or not. The maintenance topic is instead used to indicate by sending a string which type of actuator has broken. Hence the dual Publish-Subscribe nature of the actuator. Given its importance, the signal in case of quasi has QoS = 1, therefore single delivery is guaranteed.
CoAP resources reside under the headings: *coap://localhost/humidity, coap://localhost/temperature, coap://localhost/light.* The CoAPs Observe option is made useless due to the presence of MQTT.

5 Conclusion

In conclusion, we observe how the binding between entity and protocol is strictly linked to their purpose and what you want to achieve. Looking at the code is clear that actuator and sensor codes are simple and reduced without the need for huge resources for calculation and communication. The use of MQTT and therefore the need for a TCP connection could create overhead but at the same time eliminates all the weak points of CoAP. CoAP does the same with MQTTs weaknesses. The simultaneous use of these two protocols and their lightness makes them perfect for IoT and M2M applications.