Homework 3 - Group 11

Feraud Elisa (s295573), Scalenghe Luca (s303452), Tchouamou Pangop Christelle Elsa (s295753)

Machine Learning For IoT, Politecnico di Torino, A.A.2022-2023

1 Exercise 1.3 Reporting

The MQTT protocol is generally the most popular choice for IoT applications and that is for the following reasons:

- MQTT is designed for applications that require limited resources, thus in this case it is the better choice since the monitoring of the battery status needs limited resources. Furthermore, REST is not designed for this type of limited resource applications: REST is better suited for web services that communicate with the HTTP protocol.
- It uses a publish-subscribe paradigm, while REST uses a client-server. The first is better suited for this type of application because sensors can send update to a central broker when they have sufficient power resources. This broker keeps the messages for future resources that want to access them, and this is done by subscribing to the specific topic.
- The topic is a way to separate information coming from different sources and also a way to organize them. This is very practical in IoT applications where you have to minimize the waste of energy. A device only subscribes and receives messages about the topics it strictly needs, thus does not receive unnecessary messages.
- MQTT is an asynchronous protocol, while REST is synchronous. The synchronous protocol implies that a client that sends a request waits also for a response before proceeding with other tasks. Specifically for our application the asynchronous protocol is better suited that is because the latency of the connection is not predictable and so the time to wait for messages can be much longer then what is needed.
- MQTT supports real-time communications, while REST is not designed for this type of application. Because of this, in order to monitor the battery status every one second, the choice of MQTT is clearly better because reaction time is very small.

2 Exercise 2.3 Reporting

HTTP, which is a request-response protocol in a client-server architecture, has different functions that can be used by the client that through them makes questions to the server. The HTTP protocol has typically these types of requests: GET, POST, PUT, DELETE. The client can only write the header of the request, the body is then filled by the response of the server with the necessary information. In our code we implemented the methods reported in 1. In the first case the choice is straightforward because you have to retrieve the list of the monitored devices and this can only be done with a GET. In the second case there was also the specific path for the GET that you had to consider and for the specific MAC address you do a query to retrieve only the data specified in the query parameters of the HTTP request. The final method is a DELETE request. Contrary to the other two methods explained before this is a request that changes the state of the server because it deletes the specified elements in the path parameter.

Table 1: Reporting 2.3

Method	Endpoint	Description
GET	/devices	Retrieve the list of MAC addresses of the monitored devices
GET	/device/{mac_address}	Retrieve battery status information of the device with the specified MAC address in the specified time range
DELETE	/device/{mac_address}	Delete the timeseries associated to the specified MAC address