Mobile Embedded Group Project

Ilias Boulif, Jérémy Bricq, George Rusu

May 4, 2019

Contents

1 Introduction

The aim of this project is to propose an implementation of an IoT network using *Rime* where the sensor data is published through an MQTT-Rime gateway to normal MQTT subscribers. To achieve that we have implemented as requested by the project's statement, a tree-based routing protocol using Rime and an MQTT-Rime gateway.

Firstly, we are going to explain the general structure of our system. Secondly, we are going to present and explain the different messages exchanged in a normal run. Then, we will explain the organization and the routing in our environment. We will speak briefly about our gateway and MQTT subscriber implementation. And finally, we will discuss about the optimizations we managed to implement.

The implementation source code of this project is available at [1].

2 General structure of our system

Our infrastructure is divided in 2 sides: over air network (root and sensors) and wired network. This infrastucture is illustrated in Figure 1.

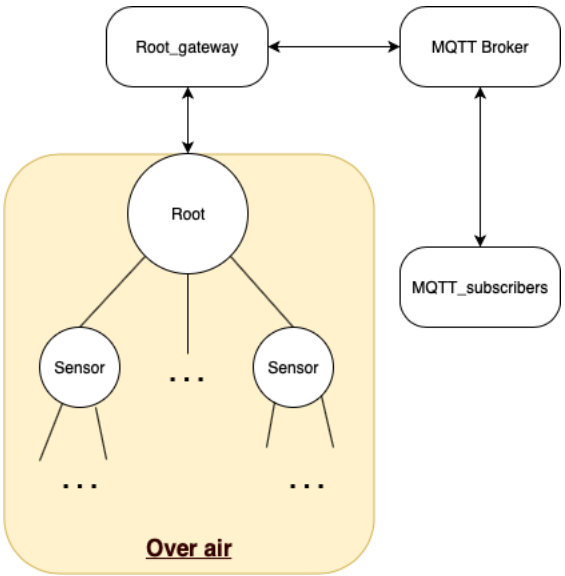

Figure 1: Representation of our structure

The over air side (yellow part) is composed of one root node and a number of sensor nodes. When the sensor nodes have some collected data to publish, they will forward the information all the way up to the root. In order to accomplish that, there exists only one path from each sensor to the root. This is achieved by the fact that, each sensor chooses a parent according to its rank. In our system, the rank corresponds to the number of hops from the node to the root. Hence, when choosing a parent, each sensor will look for the parent with the lowest rank, in other words, with the shortest path to the root.

Once arrived at the top of the tree, the information is forwarded to the rime gateway which will publish all the sensor's information to the MQTT broker. The most important component is the root of the tree because he is responsible for the transmission of the data from the air network to the wired network. Once the data has been published in the broker, any MQTT subscriber can subscribe to topics in order to receive data of interest.

In our system, the nodes communicate together by sending different kind of messages that will be explained later in this document (see Section 3). Some of these messages are used to ensure a good working of the system. Indeed, the network is built as such to keep sending information up to the root even if a node is moved or deleted, this will be explained in the Section 4.

Regarding the collection of data from the sensors by the root, there are two modes which describe the system's behaviour. The root receives some configuration messages, respectively the mode and the

number of subscribers through the gateway. He forwards the received configuration to the sensor nodes by the mean of the discovery and alive response packets, this will be explained in the Section 5.

Lastly, we will explain in Section 6 our MQTTsubscriber implementation which is basically a simple tool to subscribe to several or all topics in the broker.

3 Message format

In our implementation we have 3 packets structures. The node interprets the packet according to his type. We will explain each kind of packets and discuss the different types used by nodes to communicate between them.

Description of *packet*

1. Type : [DISCOVERY_REQUEST, DISCOVERY_RESPONSE, ALIVE_REQUEST, ALIVE_RESPONSE]
2. Rank : The rank of the transmitting node
3. Mode : The sent mode [DATA_ON_CHANGE, DATA_PERIODICALLY] - by default, data periodically
4. HaveSubscriber : Boolean that indicates if someone is subscribed to some topics

Description of *data_packet*

1. Type : [SENSOR_DATA]
2. NodeSrc : The source node transmitting the data
3. NodeRank : The rank of the node source
4. DataTemp : The temperature data
5. DataADXL : The accelerometer data

Description of *data_packet_aggregate*

1. Type : [SENSOR_DATA_AGGREGATE]
2. numberPacket : The number of contained packets
3. packet1 : The first packet
4. packet2 : The second packet

DISCOVERY_REQUEST When a node starts, his rank is equal to 0. The node will start to search for a network, and thus search for a parent to attach to. He will send in broadcast a discovery request to find a parent connected to that network. In order to have this work properly, we define that the root node will always have the same rank which is 1. Even when connected to a parent, the node will still keep sending discovery request in order to search the environment for a better parent, if there is one. The lowest the rank is (except a rank of 0), the nearest the node is to the root.

DISCOVERY_RESPONSE When a node receives a discovery request packet, he will reply with a discovery response containing his rank in unicast. The node that has made the request receive now the response and assign the his rank the value of the rank of his parent + 1.

ALIVE_REQUEST Each node has to keep sending an alive request in order to know if his parent is still available in the network. After 4 alive requests without any response, the child considers that he has no parent. If he has no more parent, his rank return back to 0 and he has to research for a new one.

ALIVE_RESPONSE When a node receives an alive request, he has to replies with an alive response to his child to validate his presence. Each alive response is accompanied with the rank of the parent. In this way, if the parent is disconnected from the network (rank = 0), the child can also put his rank to 0 and can disconnect.

SENSOR_DATA A sensor data packet is a packet that contains some information collected by the sensor. This packet travels from his sensor to the subscribers by going forwarded by the nodes on the path to the root.

SENSOR_DATA_AGGREGATE A sensor data aggregate packet is a packet that contains many sensors data packets in one. This packet will be explained in details in the Section 7 : Optimizations.

4 Network organization and routing

5 Gateway

6 MQTT-Subscriber

7 Optimizations

8 Conclusion

References

[1] Github repository, *MobileEmbeddedComputing*, Available at : `https://github.com/georgesrusu/MobileEmbeddedComputing`.