Group Project

1. Introduction

In the course we have seen two ways to send sensor data from IoT devices to MQTT subscribers:

1. With MQTT on TCP: The IoT device connects to an MQTT broker and publishes its sensor data.
2. With MQTT-SN on UDP: The IoT device sends the sensor data to an MQTT-SN gateway that forwards it to an MQTT broker.

Both solutions require an IP network (for example IPv6 with 6LoWPAN and RPL) on the device side. Despite being much more lightweight than a full IPv6 implementation, 6LoWPAN and RPL are still relatively complex and it is not always easy to obtain good network performance, as you might have noticed in the exercises.

In this project you should build an IoT network using Rime where the sensor data is published through an MQTT-Rime gateway to normal MQTT subscribers. To do this, you have to implement

- a tree-based routing protocol using Rime
- an MQTT-Rime gateway

This is explained in more detail in the following.

2. System description and requirements

2.1 The sensor network

The sensor network consists of Zolertia embedded systems communicating via IEEE 802.15.4. The sensor nodes should organize themselves into a tree with the gateway node as root of the tree. To select a parent node, a new sensor node either uses signal strength[1] or number of hops as criterion (should be configurable in your implementation). New nodes can join the network at any time and nodes can also disappear. The routing should adapt to such changes. To simplify things, unlike RPL, a node selects exactly one parent, i.e., there is exactly one path from the root node to another node.

Important: In your implementation, you are only allowed to use the Rime[2] modules for

- single-hop (reliable) unicast, and
- best effort local area broadcast.

You are not allowed to use other protocol stacks nor the existing Rime modules for routing, multi-hop communication etc.

You can choose whatever message format you like inside the sensor network. Efficiency is important, of course. See what we said about MQTT-SN in the course. Your sensor nodes should produce at least three kinds of sensor data.

2.2 The MQTT-Rime gateway

The gateway is a normal PC (e.g. running Linux). It receives the data packets from the IEEE 802.15.4 sensor network, translates them into MQTT publish messages and sends them to an MQTT broker.

Sensors <-----> Gateway (PC) <-----> MQTT broker <-----> Subscriber

The gateway should also have some simple command line interface that allows sending configuration messages to the sensor nodes. It should be at least possible to configure nodes to either

a) send their sensor data periodically

or

b) send their sensor data only when there is a change.

[1] You can find several examples in the Internet how to get the signal quality of the last received packet in Contiki.
[2] See http://contiki.sourceforge.net/docs/2.6/a01798.html and the exercise of lecture 5 for more information on Rime.

2.3 Bridging

Your laptops probably do not have an IEEE 802.15.4 network interface; therefore some bridging functionality is needed in order to allow the gateway to communicate with the sensor network. As in the exercises, you can connect one of the Zolertia nodes to your PC via USB and use it as the physical root of the sensor network tree. There are several ways how the PC can communicate with the bridge node:

- The easiest is probably to use the "serialdump" tool shipped with Contiki. As explained in lecture 7's exercise, that tool just forwards everything on stdin (PC→ node) and on stdout (node → PC). If you write your gateway software in Java, you can start the serialdump tool with Runtime.exec(…) and access stdin and stdout as input/output streams.
- If you are familiar with C, an alternative can be to write your own tunnel software. If you take a look at the source code of the serialdump tool you can see that it does not do much more than opening the USB device as a serial interface.

Unfortunately, we can only give you three devices per group. This is enough to do some general tests. However, to test your solution for large sensor networks, you have to simulate the sensor nodes in Cooja. There are two ways how your gateway can communicate with a simulated bridge node:

- Over a network connection to Cooja. To this end, you have to start a Serial Socket Server on the simulated bridge node in Cooja (see experiment #4 in lecture 6's exercise).
- By installing the serial2pty plugin in Cooja: https://github.com/cmorty/cooja-serial2pty
  This plugin allows you to create a serial device for a simulated node in Cooja that you can access from outside. In that way, your simulated gateway node appears as a serial device on the PC like a real device connected over USB.

2.4 Optimization

In MQTT, publishers always send data to the broker, even if there is no subscriber. This is a waste of energy and network bandwidth in sensor networks. Your gateway should automatically tell sensor nodes to stop sending data if there is no subscriber for the kind of data produced by the nodes.

If you use mosquitto as broker, the gateway can subscribe to the logging information of the broker and receive a message when a subscriber arrives or leaves. This is explained on

http://www.steves-internet-guide.com/mosquitto-logging/

That is a simple method for the gateway to keep track of subscribers. However, it is also unreliable: What happens if a subscriber connects to the broker before the gateway has been started? In addition to the logging approach described above, implement a solution where subscribers publish messages about the data they are interested in, so that the gateway can know what data is needed.

3. Deliverables

You have to deliver a zip file containing

- A short report in PDF format (~4-5 A4 pages, 10 or 11pt) describing
  - the general structure of your system,
  - the message format in the sensor network,
  - how the network organization and the routing in the sensor network work,
  - how the optimization works.
- All source code produced by you (sensor nodes and gateway).