Project 1. Lightweight publish-subscribe application protocol

Name: Xuan Zhang **Person Code**: 10878008
Name: Zixin Niu **Person Code**: 10892939

Tinyos code + TOSSIM simulation

Code Design

The NodeC.nc omponent implements node's application and communication logic. Each node contains a queue of messages managed with a FIFO policy, and the time at which the message is added in the queue is decoupled by the time is actually sent.

The fields of different kinds of messages are defined in the header file Message.h. We used 4 types of messages: ConnectMessage, AckMessage, SubscribeMessage, PublishMessage. Assuming that the QOS of publishing messages is always = 0, AckMessage only contains CONNACK and SUBACK which is distinguished by the field "code": code of CONNACK is 2, and code of SUBACK is 4, which defined in the Node.h.

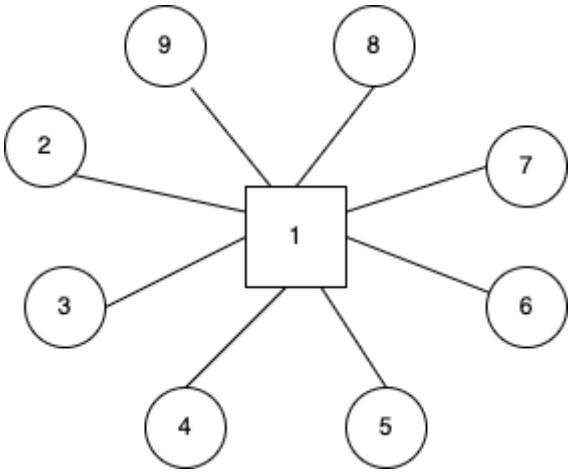
We used ConnectTimeoutTimer and SubscribeTimeoutTimer to retransmit messages, assuming that a message that has not been received for more than a certain period of time is lost. Moreover, we used three different types that is defined in Node.h to distinguish the state of nodes. For broker, after it boots, its state is NODE_ACTIVE. For clients, its state is NODE_BOOTED after booted, its state is NODE_CONNECTED after received CONNACK, and its state is NODE_ACTIVE after received SUBACK.

As regards publishing data, we used three Timer (TimerTemp,TimerHum,TimerLum) to publish messages periodically. The component Random generates the payload of publish messages.

According to the requirements on this project, the topics that each node will publish and subscribe to are initialized in Node.h to ensure that each node can publish data on at most one of the three aforementioned topics, and at least 3 nodes subscribing to more than 1 topic.

simulation setting

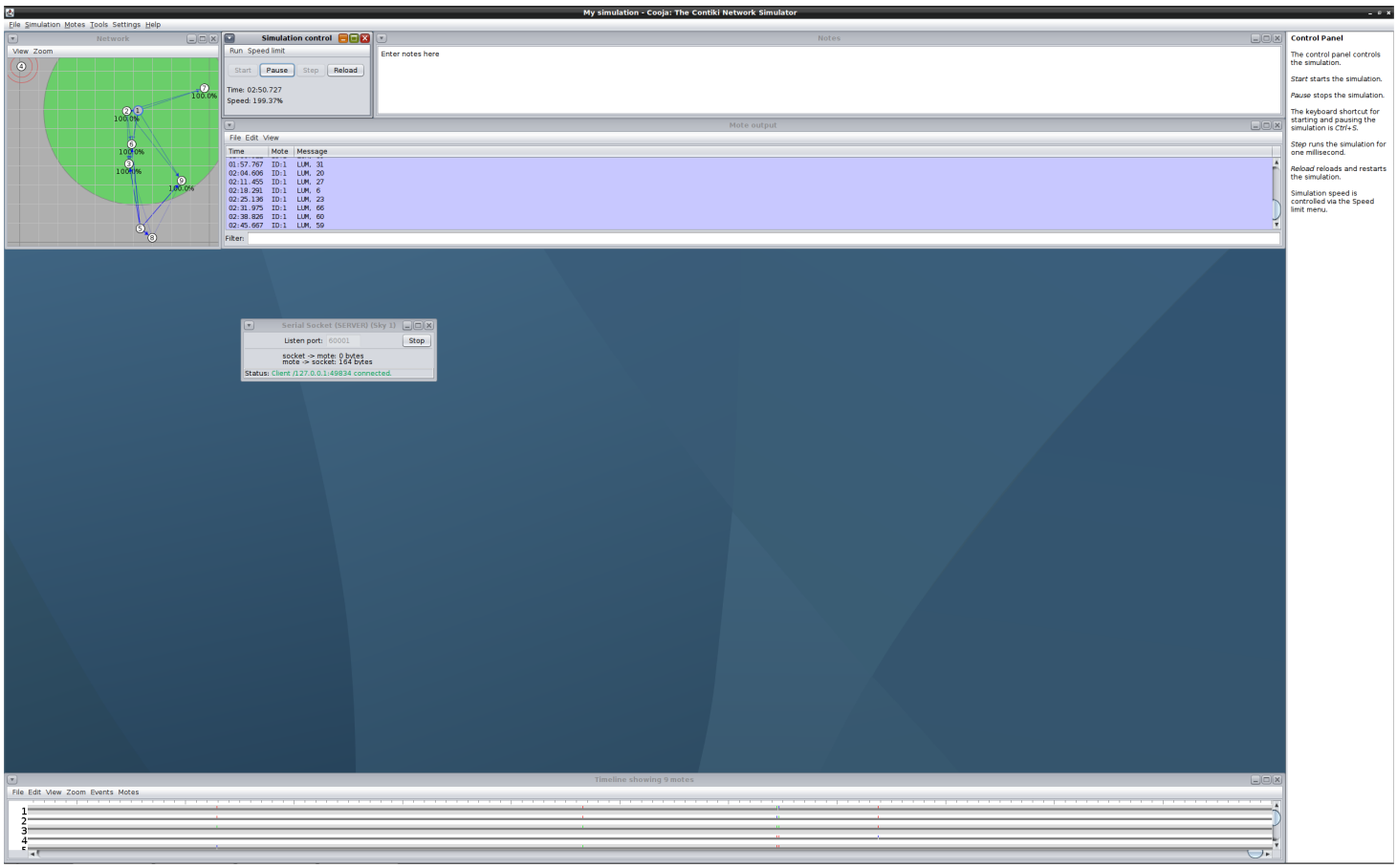
a star-shaped network topology as following is setted in the topology.txt with gain=-60.0 .



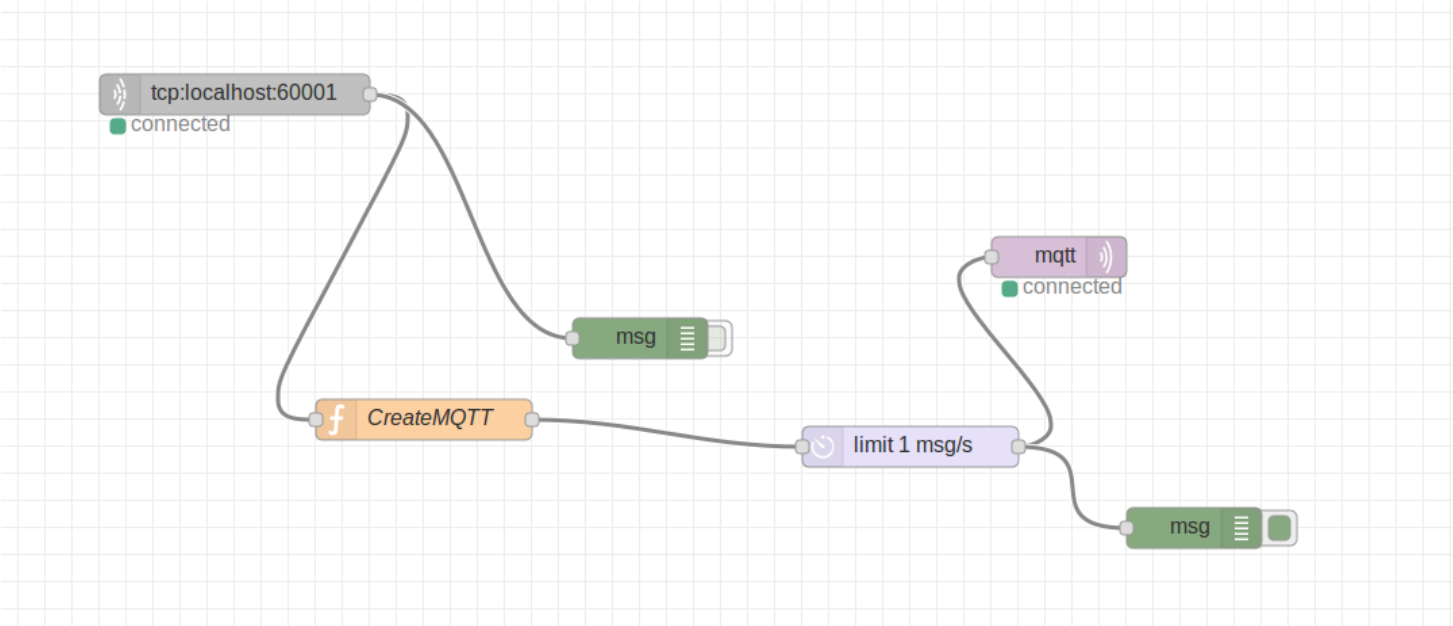
- node 1 as the broker
- node 2-9 as the clients

Cooja + Node-red + Thinkspeak

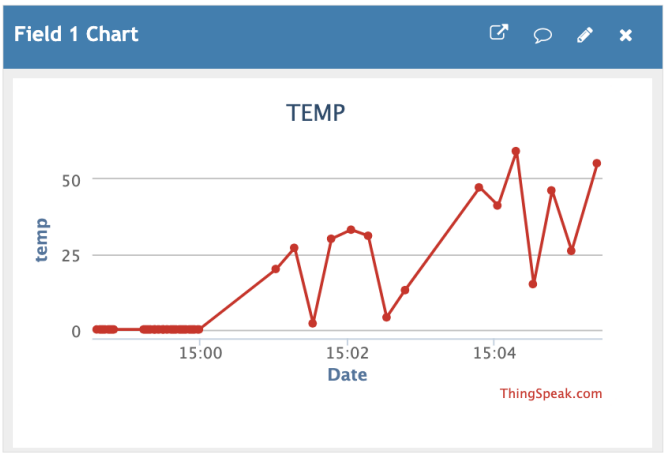
First, we used the simulator Cooja to run the built project using 9 sky motes with a serial socket(server) on mote 1 with the purpose of sending data to node-red.

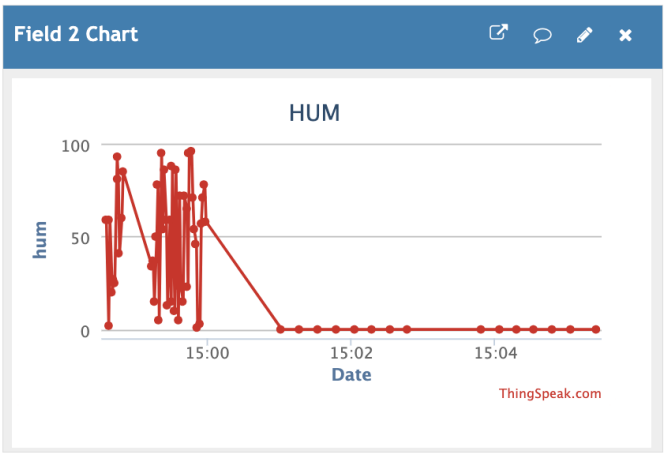


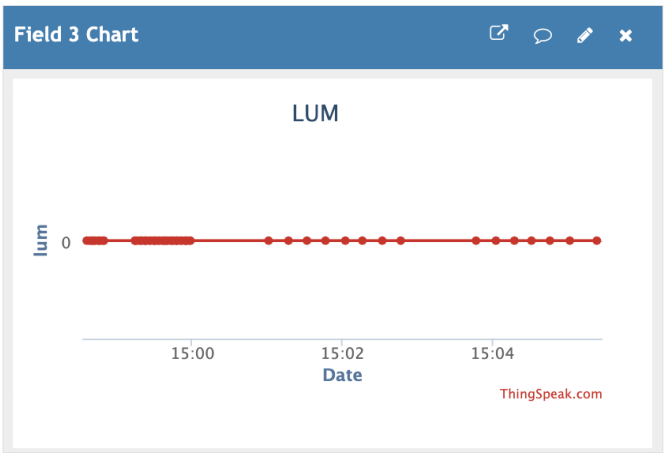
Then, in node-red we used an input TCP to get the data from Cooja. We create MQTT packets and sent them to our public Thinkspeak channel setted using mqtt connection. We used a delay node to ensure messages sent periodically.



The Thinkspeak channel where the data has been sent to is available at https://thingspeak.com/channels/2209664







Results

1. log of the simulation: **simulation.txt** contained in the delivery, relevant Cooja screenshots showed in this report.
2. the Node-red code report: **flow.json** contained in the delivery.
3. the ThingSpeak public channel URL as shown above.