Embedded devices project: IoT network and simulation with Cooja

Maeva De Keyser, Elias El Hathout

May 2024

1 Introduction

For the LINFO2146 course, we had to create a routing protocol, to establish a connection between diverse IoT devices and a server.
We decided to implement a routing protocol that creates a DODAG with the help of DIS, DIO and DAO messages. Then, the server is able to communicate to these devices through a gateway, belonging in the network.
Github link: https://github.com/maedekey/wireless_network

2 Routing

The routing is similar to the routing described in RPL. The motes can have two states, they are first not in a dodag, and then joins one and enters in the second state.

2.1 Joining DODAG

When a mote is not in a DODAG, it joins the DODAG following this sequence:

1. The mote trying to join the DODAG broadcasts periodically DIS messages (exception for the Gateway)
2. The motes in the DODAG receiving the DIS answers with a broadcasted DIO message containing its rank, and its Type.
3. The mote receives multiple DIO's and chooses the best one by first comparing their rank in the DODAG (the lower is the better), and secondly the signal strength if multiple DIO have the same rank. The mote then choose the sender of the best DIO as his parent. Note that Sub-Gateways can only choose the Gateway, and other motes can't choose the Gateway.

2.2 Periodic messages

After that, the mote enters in the second states and change its behaviour
The mote now does periodically the following actions:

1. The mote sends DIO messages at random intervals (increasing each sending) to inform children of possible changes
2. The mote sends and forwards DAO to its parent, and stores the source address and the mote type contained in the messages in its routing table which is a hash map.

2.3 Consistency Measures

The DODAG information propagates well using these messages, but to keep the DODAG consistent, we implemented also the following measures.

1. The mote have a timer starting when it receives a DIO from a parent. When it receives no DIO from its parent during the period (50 sec), the mote detach from the DODAG, sends a DIO with an infinite rank (255) to inform it's children and re enters the first state. The mote then restarts the protocol from the beginning.
2. The mote stores routes along with the time when the DAO has been received. If a child has not sent a DAO for more than 150 sec, it is deleted from the route table

2.4 Multicasting

The TURNON messages have to be delivered to every motes of a defined type (Water or Light Bulb).
In order to achieve this, we implemented a specific way to deliver these messages while generating the least possible traffic.
In our multicasting implementation, when the server orders to turn on a specific type of mote, the Gateway checks which child has a mote of this specific type in its sub-tree using its routing table, and sends them only one TURNON message. Each mote in the path to the specific types of motes acts the same way as the Gateway.
With this way, each mote forwards one message per neighbor concerned, and not one message per destination, which reduces significantly the traffic created per TURNON message

3 Messages

3.1 DIS message

DIS messages are broadcasted by every mote except the gateway trying to join a DODAG. The messages are a structure containing only the type of the mote

stored as a 8bit unsigned integer

3.2 DIO message

DIO messages are sent by every mote in the DODAG, in 3 cases :

1. The mote received a DIS mote, and answers in broadcast to share its position in the DAO, and its type
2. The mote detached from the DODAG, and inform its children by sending a DAO with an infinite rank (255 in our case)
3. The mote sends it periodically to ensure that its information are well known by the children

3.3 DAO message

DAO messages are sent by all the motes except the gateway. Their used to create routing tables on the motes. DAO are sent periodically when a mote is in a DODAG to it's parent, and are forwarded to each mote's parent until reaching the root-mote.
The message contains the source address of the mote, and each mote forwarding the DAO can save in it's routing table the source address of the DAO and the next hop (the child forwarding it)

3.4 Turnon message

TURNON messages are used by the gateway, upon receiving specific instructions from the server. They are typcially sent to sprinklers and to lightbulbs.
They are presented as structures containing:

1. The type of the message, as an 8 bits unsigned integer, representing a turnon,
2. The type of the targeted mote, also as an 8 bits unsigned integer.

The type of the message is used to identify the message upon reception. That way, motes know that they have to forward it until it reaches the targeted type mote.

3.5 Ack message

ACK messages are used by the sprinkler motes. They are sent before and after watering the plants. Such messages are actually structures containing:

1. The type of the message under the form of an 8 bits unsigned integer, representing an ACK,

2. The type of the mote sending the message, also as an 8 bits unsigned integer.

The type of the message is used to identify the message upon reception. That way, motes know that they have to forward it to the gateway, and it will know that it has to print "ACK received" on its serial port.
The type of the mote is used to identify which mote sent the ACK. It will be used by the gateway, by sending it to the server. That way, the server will be able to identify which type of mote sent the ACK.

3.6 Light message

Light messages are used by the light sensor motes. they are sent every x seconds, to the parents of the mote, which will relay it to the gateway.
Such messages are structures containing:

1. The type of message under the form of an 8 bits unsigned integer, representing a LIGHT,

2. The light value, as a unsigned integer of 16 bits.

The type of the message is used to identify the message upon reception. That way, motes know that they have to forward it to the gateway, and it will know that it has to print the light value received on its serial port.

3.7 Maint message

MAINT messages are sent by the mobile terminal after joining a DODAG. The terminal sends exactly 3 messages to it's parents, which routes it to the nearest light sensor. If the parent does not know the sensor, it is forwarded the parent of this parent.
The messages contains the source address of the mobile terminal to ensure that the light sensor knows where to send MaintAcks

3.8 MaintAck message

MAINTACK messages are sent as a reply to the MAINT messages received by light sensor. They are sent to the source address of the MAINT message.
This messages has a destination address field to facilitate the routing.

4 Server

The server is responsible for sending different instructions to different motes. It only communicates to the gateway, which relays the messages to the motes.

4.1 Watering plants

When the server is turned on, it immediately sends the "WATER" instruction to the gateway to water the plants. Upon reception, the gateway translates this message to a Turnon message, targeting the sprinkler mote type. It then sends this translation to the subgateway, which forwards the message to its children, until it reaches the sprinkler. Upon reception, the sprinkler sends an ACK message to its parent mote, which forwards it until it reaches the gateway. When it happens, the gateway prints on its serial port that the sprinkler mote type sent an ACK, to notify the server that the command to water the plants has been received.
After having watered the plants, the sprinkler sends an ACK again to its parents until it reaches the gateway, which notifies the server the same way.
The instruction to water the plants will be sent every 100 seconds, which is an arbitrary value.

4.2 Light sensing

When the light sensor mote is in place in the DODAG, it starts sending light messages to its parents, until it reaches the gateway. Then, the gateway prints on its serial port the light value, to transmit it to the server, by precising the type of the value. If it is greater than a certain value, the server sends a LIGHTBULBS message to the gateway. Upon reception, the gateway translates this message to a Turnon message, targeting the lightbulb mote type. It then sends this translation to the subgateway, which forwards the message to its children, until it reaches the lightbulb.
When the lightbulb finally receives the instruction, it turns on the light for a determined amount of time.