UNIVERSITÉ LIBRE DE BRUXELLES

Group project

INFO-Y118 - Mobile and embedded computing

DETRY Pierrick (000545842)
LAENEN Maximilien (000543889)
MANSOURI Zacharia (000392784)

May 17, 2024

Contents

Format packet

Our packet format contains 5 data fields:

- `RANK`: the rank of the node (mote) in the network: `GATEWAY`, `SUBGATEWAY` or `SENSOR`. For a `SENSOR`, even though the following piece of data is not present in the packet but handled differently, it has a sub-rank in the following: `IRG_SYS` (irrigation system), `MOB_TER` (mobile temrinal), `LGT_SEN` (light sensor) or `LGT_BLB` (light bulb). These are assigned in *cooja* in real-time using the serial port of the mote where it is possible to send data. Sending `a`, `b`, `c` or `d` will respectively assign `IRG_SYS` , `MOB_TER`, `LGT_SEN` or `LGT_BLB` as a sub-rank of the `SENSOR` mote. Each rank/sub-rank can be viewed by activating the *Mote attributes* in the *View* of the *Network* in the *cooja* simulation where a distinct colour is applied for each.
- `PACKET CATEGORY`: the category of the packet that is sent. Three categories are used for routing: `HELLO`, `HELLO_ACK`, `CHILD_DISCONNECT`, and the last category, `APPLICATION`, is used for all packets that carry information to handle the functionalities provided by the network, such as light bulb lighting when the light sensor of a network detects a too low brightness, irrigation and communication between a mobile terminal and the light sensor of a greenhouse.
- `APPLICATION CATEGORY`: used by `APPLICATION` packets, they handle
  - `APP_LGT_LVL`: the sending of the light level from the light sensor to the server;
  - `APP_LGT_ON`: the lighting of all the light bulbs in a specific greenhouse;
  - `APP_IRG_ON`: the sending of the activation command to the irrigation systems;
  - `APP_IRG_ACK`: the opening/closing acknowledgements from the irrigation systems;
  - `APP_MOB_LGT_SEN`: the communication between a mobile terminal and the light sensor in a greenhouse.
- `VALUE`: an integer value to provide additional information such as the light level, the duration for the irrigation systems to stay active, etc. When needed, this field can be modified by any mote that handles the packet.
- `SOURCE`: the source link layer address of the packet. This field is filled by the mote that creates the packet and is modified by the sub-gateways for packets sent by light sensors.

Network concepts

2.1 Setting up the network

Our protocol is based on a simple concept, once you are connected to the gateway or to a node which is connected to the gateway (and so on), you are considered *"in the network"*.

The gateway is therefore the first to be considered in the network. For others to connect to him, it broadcasts `HELLO` messages. Any sub-gateway that might be present in the area can acknowledge (`HELLO_ACK`) the connection and add the gateway as parent. This process is iterative for each node connecting to the network as illustrated below:

Figure 2.1: Initialization of the network (G = gateway, SG = sub-gateway, S = sensor)

If a node has multiple parents in range, rules described in the project requirements are applicable. A sensor should prioritize sub-gateways and best signal strength.

2.2 Maintenance of the network

Strictly speaking, the network does not implement `ALIVE` packets. However, we based our time-to-live regarding parents/children on two distinct timers:

- The first timer is dedicated to sending `HELLO` messages, as seen in the previous section. Those messages are broadcasted by all the nodes present in the network. Therefore, a parent will always receive `HELLO` packets from his children and vice-versa. The timer will, for example, be set up to send `HELLO` messages every 5 minutes. Whenever a child responds to it with a `HELLO_ACK`, the parent will mark it as alive.
- The second timer will be devoted to removing children that were not marked as alive. This timer will therefore rely on a bigger delay than the first one. Any child that does not receive `HELLO` messages from its parent will mark itself as out of the network after a while using its own timer.

If the node removed from the network was a leaf, it will now be open to accept a new parent. If the node was not a leaf, all of its children will undergo the same process.

There is still another aspect that needs to be addressed and it is related to the roaming of a node. When a child discovers a new parent providing better specifications such as better better rank or a signal strength, the child will send a `HELLO_ACK` to the new parent node but will also send a `DISCONNECT` packet to the old parent otherwise the previous parent would still consider the child as its own.

2.3 Routing

The routing protocol is separated into two classes: upward traffic (packets coming from the sensors and going to the gateway/python server) and downward traffic (packets coming from the gateway/python server and going to the sensors).

The upward traffic is the simplest to implement. All the traffic is merely forwarded to the parent of the node emitting which in turn, forwards it again to his parent, etc. The python server, on the other hand, is just connected via the serial interface of the gateway.

Regarding downward traffic, we distinguish two scenarios:

- The activation of the irrigation system using multicast
- The activation of the smart light bulbs of a single greenhouse

The activation of the irrigation system is done by sending a multicast packet from the python server to all greenhouse's gateway. Upon receiving this packet, the sub-gateways will broadcast the packet to reach the correct sensor.

The smart light bulbs activation is achieved by responding to the light sensor packet. We therefore know from which sub-gateway the packet comes from. The process of reaching the light bulbs from there is the same as for the irrigation system.

For clarity purposes, here is an example of the light sensor sending its brightness level. It will first send it to its parent until reaching the server. Once the packet has been received, the server will check if the brightness level requires a light switch or not. In this example, the percentage is lower than 70% so the server sends a message to turn on the lights of that greenhouse.

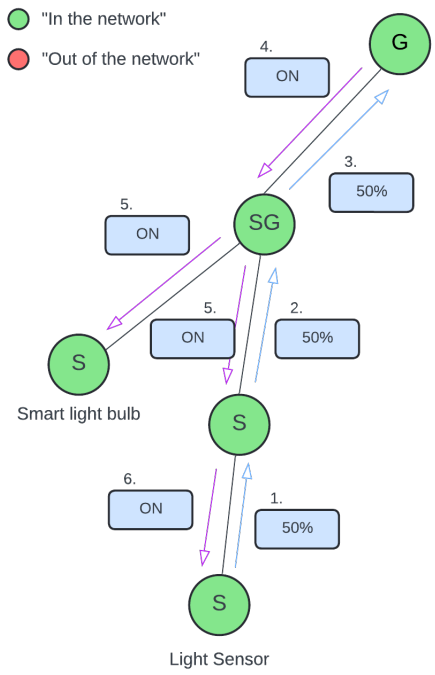

Figure 2.2: Path of a packet

A mobile terminal can enter the network and send packets to (then receive responses from) the light sensor of a greenhouse. For that purpose, the mobile terminal first sends a packet with a specific app category and a value of `0`. That packet is routed by all parent sensors to the sub-gateway that increases its value field and then broadcasts it to the light sensor of the greenhouse. It also increases its value field and sends it back to the sub-gateway that once again increases its value field and broadcasts it to the mobile terminal, that therefore receives it as an answer from the light sensor.

2.4 Additional information

As mentioned in the assignment, to be able to communicate with more than two children, you need to change the line: `#define CSMA_MAX_NEIGHBOR_QUEUES 6`
in the file: `contiki-ng/os/net/mac/csma/csma-output.c`.

In our project, the maximum number of neighbors a node can have is 6. If we moved out to a building and needed for example a sub-gateway per room, we would need to change this value again for a greater one.