LINFO2146 - Group Project

March 5, 2024

1 Practical Info

- Deadline: 17/05/2024.
- Group size: Maximum 3 people.

2 Introduction

We consider the (fictive) scenario of smart greenhouse fields. Several smart greenhouses are scattered across a large plain. Each greenhouse contains IoT devices to control the climatic conditions for the plants inside the greenhouse:

- A sub-gateway.
- A light sensor and two light bulbs.
- One device connected to the global irrigation system.

Each device is connected to an external server responsible for managing the greenhouses.

2.1 Sub-gateway

The devices from a same greenhouse are part of a sub-network with a sub-gateway handling all the communications for this sub-network.

2.2 Light management

The light sensor sends periodically the current greenhouse light level to the server. If the light is below an user-defined threshold, the server will command all the light bulbs in the greenhouse to turn on for X minutes. (You can choose the value of X).

2.3 Irrigation system

At a specific time of the day, the irrigation system is run for all the greenhouses. The server triggers the irrigation system by sending a command to all devices connected to it in each greenhouse. Upon receiving this command, the devices send a acknowledgement to the server and start watering the plants for Y minutes (Again you can select the value of Y). Each device sends a message to the server at the end of the watering process confirming that the process was run successfully.

2.4 Mobile Terminal

From time to time, a greenhouse operator comes into a greenhouse with a mobile and interact with the light sensor for maintenance. it will send Z messages to the light sensor and receive a answer for each message.

2.5 Architecture

We call these devices "sensor nodes" in the following. The sensor nodes communicate over a wireless IEEE 802.15.4 multi-hop network. Figure 1 shows an example of a network topology.

Concretely:

1. The server communicates with all the sensors but the mobile. Communication are either periodic (light sensor) or when a triggering condition is met (light bulbs and irrigation system).
2. All communication are one-sender-to-one-receiver except when the server starts the irrigation system or turns on the light bulbs of a greenhouse which is a one sender-to-many-receiver communication.
3. The operator may move from one greenhouse to an other thus disconnecting and connecting regularly to a greenhouse sub-network.
4. Communications between the mobile an any device in a sub-network stay totally local in that sub-network. It means that communications do not go higher than a sub-gateway in the network tree.

3 Implementation

You have to implement the system described in the previous section.

3.1 Sensor nodes and wireless network

Nodes communicate over IEEE 802.15.4. The nodes must organize themselves into a tree with the gateway node as root of the tree. Additionally, the direct children of the gateway must be sub-gateways. To select a parent node, a new sensor node:

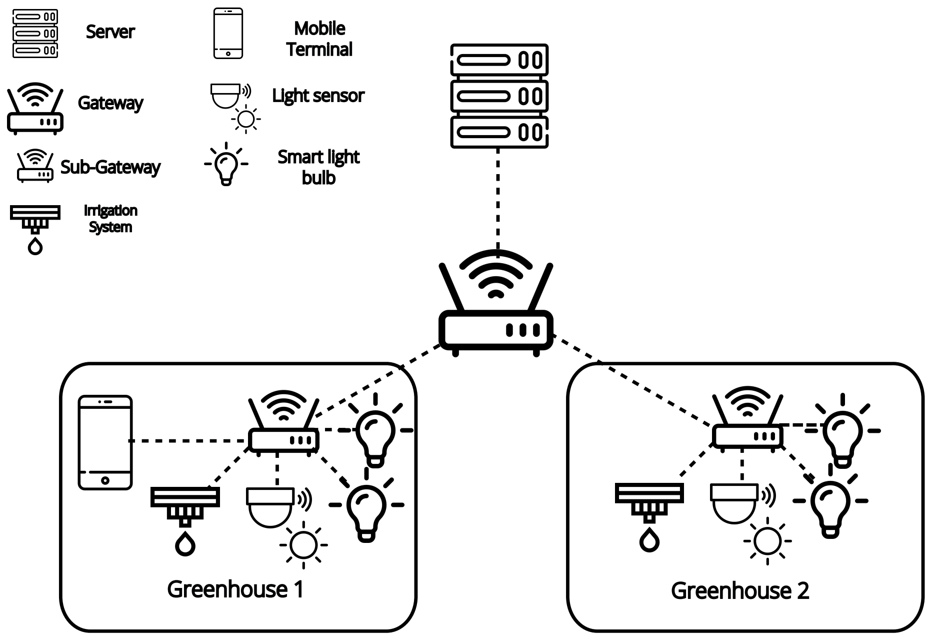

Figure 1: Network Topology [1]

- Checks if among the parent candidates, there is a sub-gateway node. If yes, the sub-gateway node become the parent.
- If there are multiple sub-gateway nodes as potential parent, the sensor nodes uses the signal strength to decide.
- If there are only sensor nodes as parent, the nodes uses the signal strength to decide. It can happen if a node is to far from the sub-gateway and as to use an other sensor as relay.

Except for the gateway and sub-gateway, nodes can join the network at any time and can also disappear. The routing must adapt to such changes.To simplify the design, unlike RPL, a node only chooses one parent at a given time, i.e., there is exactly one path from the root node to any other node.

Important: In your implementation, you need to implement your own routing protocol, which means that you should not use a network stack (no RPL). You need to manage the routing by yourself. To that end, you will need single-hop unicast and broadcast. There are two examples in contiki-ng/examples/nullnet that can help you.

The network should be emulated in Cooja. For the nodes, you can use the Z1 mote type. The light sensor nodes should produce fake sensor data. How you produce the data is left to you. You don't need to do anything very complex, random values are enough.

You will find on Teams two files showing you an example of such communication:

- serial_test.c : A program running on a node in Cooja that prints the message it receives on its serial port
- server.py: An external python program connecting to the mote (taking the IP address of the docker and the Serial Socket Server port as argument), sending 20 times "test" and printing the output of the mote.

3.2 Gateway and server

The server is an application running on Linux. It receives messages from the nodes. You can write the server in C/C++, Java or python.

Some bridging functionality is needed between the gateway (i.e. the node at the root of the wireless network) and the server. The easiest way to allow an external application to talk to a node in Cooja is via a network connection as seen in the exercise[2]. You must first create a Serial Socket on the bridge node in Cooja, then the server application can connect to the port of the Serial Socket Server. In that way, your gateway node can send messages to the server by writing on its serial interface and receive messages from the server by reading from its serial interface.

3.3 Multicasting

To start the watering process in every greenhouse, the server sends a command to every devices connected to the irrigation system. Networking-wise, it means that your protocol should support multicasting. It means that the server sends only one message but it is received by every device connected to the irrigation system. Multicasting must also be used when the server want to turn on the light bulbs in a specific greenhouse. The idea behind multicasting is to reduce the number of messages that must be sent over the network, think about how it can be achieved for those two scenarios. For this context, you can simulate multicasting by replicating message at specific point in the network tree.

4 Deliverables

You have to deliver a short report in PDF format(5 A4 pages) describing:

- The message format in the sensor network
- How the routing in the sensor networks works
- How you implement the multicasting.

[2]You will see it soon

Keep the report short and only give the essential information, i.e., your design decisions, not source code. The report must contain a link to a Github repository read-accessible to us with the commented source files of the code for your nodes (sensor nodes, gateways nodes,...) and the server. The Github accounts of the assistant are: **wiauxb** and **gkabasele**. You must add both of us to your repository. Don't forget to put the names of all group members on the first page of the report.

5 Problem you may encounter

By default in the simulator, a node can no more than 2 neighbor before exhibiting strange behavior. This is a limitation of the CSMA implementations, it creates a buffer queue per neighbor and limits the number of queue to 2. You need to change the file `contiki-ng/os/net/mac/csma/csma-output.c`. You must change the 2 on line

```
#define CSMA_MAX_NEIGHBOR_QUEUES 2
```

to the number you want.