LINFO2146 - Group Project

March 21, 2025

1 Practical Info

- Deadline: 16/05/2025, 20:00.
- Group size: Maximum 3 people.

2 Scenario

We consider the (fictive) scenario of a building management system: There is a large number of wireless devices distributed in a building, where each device is directly attached (by cable) to an air quality sensor and to a motorized valve for the air ventilation. Based on the analysis of the sensor data, the device should open or close the valve. We assume:

- Sensor values are read once per minute.
- If the slope of the line obtained by a least-squares fit to the last thirty sensor values is above a certain threshold (you can choose the value), the valve is opened for 10 minutes.
- Devices are not powerful enough to do the data analysis by themselves. Instead they send their latest sensor value to a server outside the wireless network which does the analysis and then sends a command to the device to open the valve if necessary.

We call these wireless devices "sensor nodes" in the following.

The sensor nodes communicate over a wireless IEEE 802.15.4 multi-hop network. However, a straight-forward implementation would result in too many communications if the number of sensor nodes is very large. To solve this problem, we introduce "computation nodes" in the wireless network. These are special devices that have more resources than the sensor nodes and that can do the data analysis for them. This should happen completely transparently: The sensor nodes still think that they send their data to the server. If there is a computation node on the path from a sensor node to the server, the computation node will catch the data message (i.e. not forward it to the server), store the data locally, do the data analysis, and then send the outcome of the analysis

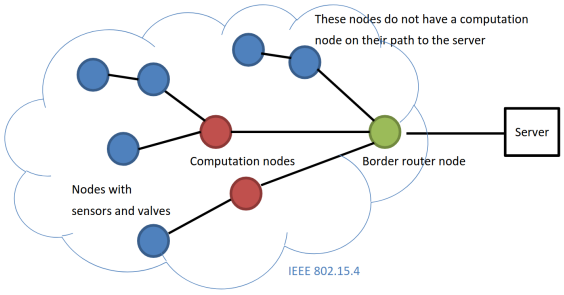

Figure 1: Example of Topology

("open valve) to the sensor device if necessary. Figure 1 shows an example of topology.

In order to not overload the computation nodes, a computation node should only do the data analysis for a limited number (e.g. 5) of sensor nodes. If there are more nodes, it will forward their messages to the server. Note that sensor nodes can appear and disappear, i.e. the system does not know in advance for which sensor nodes a computation node will be responsible. If a computation node has not received new data from a sensor node over a longer period, it assumes that the node does not exist anymore and that any stored data from it can be removed to make space for data from other nodes.

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 border router node as root of the tree. New nodes can join the network at any time and nodes can also disappear. The routing must adapt to such changes. Your protocol must be efficient in terms of routing, meaning adapted to changes in the network but also efficient in terms of energy. For example the amount of control messages sent by a node can depend on the stability of the network or the parent selection process can take into account energy metrics sent by the nodes. Typically, nodes running low on energy should not have too many children. Nodes should choose a new parent (if available) before the parent runs out of energy. Those are some examples. The routing protocol that you design should guarantee that the messages sent reach their destination while saving as much energy as possible on the mote. To measure the impact of your design on the energy, you can use the module Energest from Contiki-NG: https://docs.contiki-ng.org/en/master/doc/tutorials/Instrumenting-Contiki-NG-applications-with-energy-usage-estimation.html.

Important: In your implementation, you need to implement your own routing protocol meaning 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. You can also check the Contiki-NG Documentation:

https://docs.contiki-ng.org/en/master/doc/programming/NullNet.html

The network should be emulated in Cooja. For the sensor nodes, you can use the Z1 mote type. Since we don't have air quality sensors and valves in Cooja, the sensor nodes should produce fake sensor data. You can represent the status of the (fake) valve by a LED. For the computation nodes, you can use devices of the type "Cooja mote". These are powerful virtual devices that are directly executed on the CPU of the host computer without resource limitations.

3.2 Border router and server

The server is an application running on Linux. It receives and replies to messages from the nodes. You can write the server in C/C++, Java or python. Some bridging functionality is needed between the border router (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. You must first create a Serial Socket on brigde node in Cooja, then the server application can connect to the port of the Serial Socket Server.

In that way, your Border router 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. 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, connection to the mote (given the required argument) and sending "test".

4 Comments

You must make an implementation such that it is easy for a user to understand what is happening during the simulation in Cooja. Messages from the motes on the Mote Output Message Tool from Cooja must be human-readable and easy to understand. They must provide information about network events like when a node selects a parent or when a node disappears, or even when a valve has been turned on. On the server side, the messages must be explicit as well like the data being received by a mote or when a command has been sent by the server.

5 Deliverables

To submit your project, you must create a Github repository read-accessible to the assistant. The repository name must start with `LINFO2146_2025`. It must contains:

- The source code of your project for the nodes (sensor, computation) and the server.
- A PDF file of your report.
- A README containing the name of the students in the group and their NOMA.

Important: The Github account of the assistant to add to the repository is: **gkabasele**.

5.1 Report

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

- the message format in the sensor network,
- how routing in the sensor network works,
- the decisions to make the protocol more energy efficient,
- how the computation nodes work.

Keep the report short and only give the essential information, i.e., your design decisions, not source code. In your report, we ask you to choose one feature of your protocol implemented to save energy and include a graph showing how this feature impact the energy usage of the network. The metric used (power consumption on a Z1, time before the first node of the network dies, etc) is left to your appreciation. **Don't forget to put the names of all group members on the first page of the report.**

6 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.