LINFO2146 - Group Project

March 29, 2023

1 Practical Info

- Deadline: 19/05/2023.
- Group size: Maximum 3 people.

2 Introduction

We consider the (fictive) scenario of a building management system. There is a large number of wireless devices scattered across the several areas of the building. Each device is connected to a motion sensor which measures the activity in its covered area. We assume:

- Every time someone passes next to the sensor, a counter inside the sensor is incremented.
- The sensors send their counter values to a server outside the wireless network which will do some analysis with the data.

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

The sensor nodes communicate over a wireless IEEE 802.15.4 multi-hop network. Because there can be a large number of sensor nodes, the sensor nodes need to organized themselves via some kind of scheduling mechanism. To this end, we introduce *coordinator nodes*. Those are special devices directly (i.e., one hop away in the wireless network) connected to the border router and their role is to schedule the transmission of data for a set of sensor nodes such that only one node is allowed to send its data at a given time. Figure 1 shows an example of a network topology.

Concretely:

1. The border router assigns a time-slot to each coordinator node.
2. The coordinator node assigned to the current time-slot will inform the set of sensor nodes in its area.
3. The sensor nodes send their counter value to the external server via the border router.

4. The time slot moves to the next coordinator node.

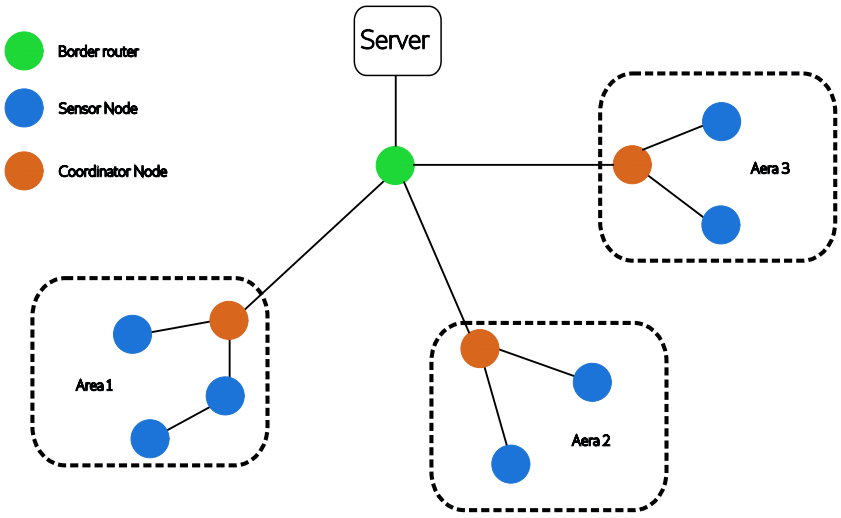

Figure 1: Network Topology

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. To select a parent node, a new sensor node:

- Checks if among the parent candidates, there is a coordinator node. If yes, the coordinator node become the parent.
- If there are multiple coordinator 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.

New sensor nodes can join the network at any time and nodes 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. We don't have motion sensors in cooja, so the 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.

3.2 Border router 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 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 the bridge 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 (not necessarily needed).

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) and sending "test".

3.3 Scheduling

As mentioned in Section 2, you have to implement a scheduling mechanism such that only one sensor node can send data to the server at a given time.

3.3.1 Border Router and Coordinator Node

The border router has a fixed-size time window that is divided into time-slots whose duration depends on the current number of coordinator nodes. You can choose the duration of the time window and the duration of the time slot. During a time slot, only the sensor nodes managed by the coordinator node to which the time slot has been assigned can send data. This approach requires that the border router and the coordinator nodes are timely synchronized. To achieve this synchronization, you will implement the Berkeley Time Synchronization Algorithm. This algorithm works as follow:

1. A node is selected as the master node, in this case it is the border router.

2. The master node periodically collects the clock time of each neighbor, in this case it would be the coordinator nodes, and its own.

3. The master node computes the average time difference between its clock time and the ones it receives. Then it adds this average time difference to its own clock time. It then forwards the new clock time to the neighbors.

4. Each neighbor updates its clock time with the clock time received by the master node.

To implement this algorithm, you will need the clock library of contiki[1]. This library has the function `clock_time` which gives the current clock of the device in systems ticks. This clock can not be modified via contiki, so you can not use it as such, you will have to implement a new clock based on this one.

When a coordinator starts, it looks for the border router and once connected, the border router assigns the time slot and the time synchronization algorithm is run. The elapsed time between two runs of the algorithm must not be more than 5 seconds.

3.3.2 Coordinator Node and Sensor Node

During its assigned time slot, a coordinator node can forward the data of the sensors node. Since only one node can send at a given time, a scheduling mechanism is needed. We present two possible solutions but you can come up with your own solution.

Polling: The coordinator asks a sensor node if it wants to send something. If the node has something to send, it directly sends its data and when it has nothing more to send, it notify the coordinator with a specific message. The coordinator performs this process with each sensor node in a round-robin fashion.

Time slot: This is similar to the scheduling between the border router and the coordinator nodes. The coordinator node assigns time slots to each of its sensor nodes.

4 Deliverables

You have to deliver a short report in PDF format(4 A4 pages) describing:

- The message format in the sensor network
- How the routing in the sensor networks works
- How you coordinate the sensor nodes.

[1] `https://docs.contiki-ng.org/en/master/_api/group__clock.html`

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 or Bitbucket repository read-accessible to us with the commented source files of the code for your nodes (sensor nodes, coordinator nodes, border router) and the server. Don't forget to put the names of all group members on the first page of the report.