LINFO2146 - Mobile and Embedded Computing

Project
Simulation Of Air Quality Sensor Network

Team Member Names:
Igor Nunes Ferro - NOMA: 07092100
Mojgan Giahi - NOMA: 09692100

Professor:
Prof. Ramin Sadre
Course Assistant:
Gorby Kabasele Ndonda

A report submitted in partial fulfilment of the requirements of the
UCLouvain for the degree of
Master of Science in *Computer Science and Engineering* and *Data Science*

Academic Year: 2021-2022

1 Introduction

Abstract

In this project, we are going to implement a simulation of the air quality sensor network. The assumption is that there is a network of wireless devices, known as sensor nodes, in a building, each of which is attached to an air quality sensor as well as a motorized valve for air ventilation. Since the wireless devices are not powerful enough to do the data analysis by themselves, they send their latest sensor value to the server outside the wireless network, and the "open valve" command will be sent from the server to the devices based on the sensor data analysis. To this end, sensor values are read once per minute. The valve is opened for 10 minutes if the slope of the line generated by a least-squares fit to the past thirty sensor values is greater than a specific threshold. These devices communicate in a tree-shaped network following our designed routing protocol. To avoid network traffic due to a large number of sensor nodes, some computation nodes are represented in the network, having more resources than the sensor nodes. When they receive data from a sensor node, these computation nodes can do the analysis and request to open the valve themselves. However, this happens completely transparently, meaning that the sensor nodes still think that they send their data to the server. To avoid overloading the computation nodes, a limited number of sensor nodes has been assigned to each computation node for analysis. In case the limit is exceeded, it will forward their messages to the server.

In the following report pages, we will outline the message types that are exchanged in the network and with the server. Then, the routing protocol used in our sensor network will be presented. After that, the computation nodes function will be explained. Finally, we will present a conclusion of our work. Following is the link to our GitHub repository, which includes all commented source code of the code for our nodes (sensor nodes, computation nodes, border router) and the server:

`https://github.com/igorferro1/LINFO2146_project`

2 Message Format in the Sensor Network

2.1 Explanation

For the whole network, there are 4 message types:

1. **Announcement messages**: This message is dedicated to announcing that a node can be the parent of another node; *"I can be your parent"*. The announcement messages are sent every 1.5 minutes to each node.
2. **Parent Down messages**: This message is dedicated to informing the neighboring nodes that a node lost its parent, and thus requests to rebuild a tree if a node is lost; *"I lost my parent"*. A message like this is sent periodically from the Root node, and then it will propagate down from the Root node and it will just re-create the tree in case there are some lost nodes in the way.
3. **Data messages**: This is the message that carries the data that is being sent once per minute by each sensor node to the server with the values that are read;
4. **Open Valve messages**: This is just a command sent by the server or a computation node to a sensor node to request opening the valve of the sensor for 10 minutes;

All types of messages start with a number, encoded on 8 bits, that represent the type of the message. This number has the value 0 for Announcement, 1 for Parent Down, 2 for Data, and 3 for Open Valve messages.
The structure of our created messages will be described in more depth in the following section.
The message is structured with:

- The message type
- The variable that tells whether it is a *Broadcast* or *Unicast*
- The rank of the sender: the immediate neighbor who sends the message
- The address of the node sending the data so that in case it is a Data message we have the address of the node that sends the data in order to reply with the "open valve" message
- The value read, which is the value that is sent on the Data message from the node to the server

2.2 Implementation

The sensors react to a few kinds of messages based on the following explanations:
For receiving the Broadcast message type: Sensors react to the Announcement messages so that if a new node is better than the current parent of a node, based on the RSSI of the last message, it will change the parent node. Any node can become a parent if some other node does not have any parent. Besides, whenever the sensor receives messages from a "down parent" message, it announces to the other nodes that a parent is lost. So, if there are nodes under the tree's current node, they will be reconnected.
For receiving the Unicast message type: Whenever the unicast message comes from other sensors with data, if it is the data intended to send to the server, the sensor will receive the data from the other sensors and just pass the data without any modifications. The same applies to the commands. That is to say, once the sensor receives the command, it checks whether the command to open the valve is for the sensor, and if it belongs to the sensor, it opens the valve for 10 minutes; otherwise, it will just keep passing the command to the next hub.

3 Routing Function in the Sensor Network

3.1 Explanation

The routing protocol used by the nodes to build a tree-shaped network is inspired by *RPL*, with some simplifications. Unlike *RPL*, a node can only have one parent at a given time. This section will describe the protocol in detail.

3.2 Implementation

For unicasts, it is necessary to route the received messages. Considering only sensor nodes, if a received message is just passing data, the node will basically send it to its parent without having to check anything else, but before sending it, it will add the original node that sends the data to its routing table. If the received message is a command message, then the node is going to look at its routing table. Each node has a routing table, and this routing table comprises the entries related to each of the nodes in the network tree branch that is connected to the respective node. When the node receives a command message, it checks the message's destination node and passes it to the next hub based on the existing routing table. For instance, we assume that there is a sensor node along the top of the tree, and there is also a sensor node in the middle. In this case, if the middle sensor node receives a command from the sensor node on the top of the tree while the command's destination is the further down sensor node, it passes the command in the direction of that lower destination node based on its routing table. Hypothetically, there are 5 nodes connecting as follows:

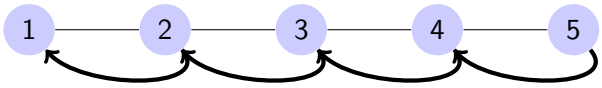

Figure 1: Sensor Node Tree

As shown in Figure 1, sensor node 5 is sending the data to node 1. in this situation, node 4 will receive the data first, and it will perceive that it is the data coming from node 5, so it will write down in its routing table that node 5 is directly connected to it and if it had any message to node 5, it could simply send it to node 5 from this path. Likewise, the same applies to node 3. Once node 3 receives the message, it perceives that it received it from node 5, so it will update its routing table, stating that if it had any message to node 5, it should send it to node 4. The same applies to node 2 and node 1.
Therefore, each node has a routing table, and the routing table has multiple entries. Each entry of the routing table has two parameters; the target node, which is the end node (in Figure 1 it is node 5), as well as the next hub, which specifies which node you have to send the data in order to reach the target node. Moreover, this is an indicator of whether the entry is alive. As a result, the implementation is basically strict.

In this small section, a brief of the implementation of the Border router node (Also known as the Root node) is explained. Since the computation nodes calculate the value and send the command only for 5 nodes, so if there are not enough computation nodes in the network, the data will eventually reach the Root node. Once it reaches the Root node, it will be sent to the server, and the server will keep the values and do the calculations. Thus, the Border router node is basically a node connected to the server, and it gets the data from the sensors and sends it to the server if the calculation is not made in any of the calculation nodes in the middle.

4 Computation Nodes Function

4.1 Explanation

Concerning the computation nodes, only the data plane is different from the sensor motes. Indeed, they do not need to create data since they are not linked to a sensor, and they can intercept data going to the root to do the computation instead of the server. This lightens the workload of the server and can be significantly beneficial either if some nodes are far from the root or if the network consists of many nodes. Since the computation nodes also have physical limitations (physical/virtual memory), they can not choose to do the computation for every node that goes through it when sending data to the server. The number of sensor nodes for which the computation node can do computations is thus limited (here the limit for the number of sensor nodes is 5).

4.2 Implementation

Each minute, the sensors get a renewed value of data and then forward it to their parents. The parent could be another Sensor node, a Computation node, or the Root node. In case the parent is a sensor node, this sensor parent will just forward the data that the sensor is sending so that the data will be sent to the next parent until it finds a computation node or the root node that can calculate the value of this loop and then send the command to open the valve.

In this section, we explain in detail what happens when the data that the sensor is sending reaches a computation node and is read by the computation node.
In this situation, the computation node will check whether it is calculating the data for the sensors that it is in charge of, since each computation node can only analyze data for a limited number

of 5 sensors (In order to not overload the computation nodes). Taking into account the limited number of sensor nodes assigned to each computation node, if the arrived data is from the 5 sensors that exist in the list of the computation node and that sensor is down in the tree for that specific computation node, it will just calculate the loop and check whether it is needed to open the valve and if necessary it will send the message to the respective sensor to open the valve. Otherwise, if the sensor is not in the list to calculate the loop in that corresponding computational node, the computational node will simply forward the data to its parent. The parent may be a sensor node or computation node. In case it is a sensor node, as explained, the sensor node will forward the message. If it is another computation node, it will check if it is calculating for the sensor node from its 5 node list that each computation node calculates, and if it was among that 5 node list, it would send the command; if not, it will forward the data. It just keeps continuing, and ultimately it routes to the server and will be calculated in the server.

We also found a mechanism to handle the situation when a node goes down so that the nodes that are connected to this node could be able to search for new parents. Our approach is that from time to time (every 5 minutes), the Root node sends a message, and this message is of the Broadcast type that is going to spread, and consequently, the message requests the nodes to rebuild the tree.

5 Challenges

One of our biggest challenges was to implement a large system without having message conflicts. Also, implementation of the nullnet was complicated since there was not enough available references for the nullnet on the internet.
There was only one problem that we suppose was related to the system that we used. The nodes forwarded the messages but based on our observations, sometimes they did not forward some of the messages. It usually happened when the speed of our simulation was set to 1000 percent. The reason for setting it to 1000 percent was that the data was sent once a minute, and we wanted to fasten the waiting period. Anyhow, we believe that this was related to our system and is not a problem in the project. One other remark is that we tried the communication between Cooja and Z1 motes, but we couldn't, so we decided to use Z1 motes as all types of nodes.

6 Conclusion

The network we have built can meet all the requirements of the project. The sensor nodes can find their parents correctly and open the valves if they receive the appropriate data. In addition, the network is adequately checked by the Root node so that if the node is lost in the network, the overall structure and performance of the network will not be distorted, and eventually, the network does its operations appropriately. We can perform other similar operations by establishing a proper connection between the nodes and building a suitable tree. The most crucial factor in this network is sending the correct data and the proper routing protocol in order to send data between nodes.