UCLouvain

Université Catholique de Louvain

LINFO2146
Mobile and Embedded Computing
Project Report

Huberty Nicolas 54961900
Cammarano Vincent 53912100
Arnita Andrew 04882100

nicolas.huberty@student.uclouvain.be
vincent.cammarano@student.uclouvain.be
andrew.arnita@student.uclouvain.be

Professor : Prof. Ramin Sadre
Course Assistant : Gorby Kabasele Ndonda
Academic year 2022 - 2023

0.1 Introduction

In this project, we consider the scenario of building management system where a large number of wireless sensor nodes, are put across various areas of a building. Firstly, we will explore the message format in the sensor network, which will involve understanding how the sensor nodes communicate with each other. Then, we will investigate how routing in the sensor networks works. We will see how data packets are transmitted through the network from the sensor nodes to the external server via the coordinator nodes. Finally, we will look at how to coordinate the sensor nodes. The coordinator nodes were used to achieve this, it schedules the transmission of data for a set of sensor nodes to the border-router. We worked on github to share/test each others code.
https://github.com/NicolasHuberty/LINFO2146

0.2 Message format in the sensor network

To implement the functionality of having nodes transmitting information in a network, we need to define the message type that it sends to it neighbours, in this case sensor and coordinator nodes. We implemented a message format that is used by every node in the network, it goes as follows:

- **RSSI:** each node sends its own RSSI which defines how strong its signal strength is
- **Node Type:** We have 3 different types of nodes in the network: Border_Router, Coordinator, and sensor. This information is important to be shared with other nodes to inform which type they're working with
- **Message Type:** We consider 10 types of messages that can be transferred from one node to the other, these types will be discussed later on.
- **Data:** In a wireless network, node share data between, in this case the data is random numbers that are calculated that represent the motion sensor data.

Here, we are going to discuss the different types of messages that can be sent by all nodes across the network:

1. **HELLO_TYPE:** a message type that initiated the communication between nodes. This helps the neighbouring nodes within the range of reach, to know how many nodes and their types that are present in the network. When it is sent, a RESPONSE_HELLO_MSG message from the nodes that received it is sent back to the initiator.
2. **ASK_CLOCK_TYPE:** a message type used to request information about the clock from other nodes in the network.
3. **GIVE_CLOCK_TYPE:** a message type used to provide information about the clock to other nodes in the network.
4. **SET_CLOCK_TYPE:** a message type used to set the clock and store the time slot using RSSI.
5. **ALLOW_SEND_DATA:** a message type used by the coordinator to allow a sensor node to send its data.

6. **DATA:** a message type used to transmit data between nodes in the sensor network.
7. **CHOSEN_PARENT:** a message type used by a sensor node to notify its chosen parent in the network.
8. **RESPONSE_HELLO_MSG:** a message type used to respond to a hello message sent by another node.
9. **TRANSFER_DATA:** a message type used by the coordinator to transfer data to the border router.
10. **NOT_MY_DATA:** a message type used by the sensor parent node of a child sensor node to forward the child's data to the coordinator node.

We will discuss the implementation process that is used to have all the nodes in a sensor network communicate with each other and transfer messages. From the perspective of a sensor node, when the simulation starts, it sends a HELLO_TYPE of message to the nodes that are within reach based on its range. This will initiate the communication with its potential parents and the rest of the nodes that are in the network. When the other nodes in the network receives the message from the sensor node, the process of choosing a parent begins. We need to know if a coordinator node is present or just sensor nodes are. In this case, 4 different scenarios are present, which are as follows:

- **Coordinator Node Selection:** in this scenario, if among the nodes that received the HELLO message from the sensor node, is in fact a coordinator node, then the sensor node itself chooses this node as its parent so it can use it to send data to the server.
- **Sensor Node Selection:** in this scenario, we check if the type of nodes present in the range are all sensors, if this is the case, the sensor node chooses another sensor node that has the best RSSI as its parent. This helps the child sensor node to forwards its data to the coordinator node via its parent sensor. A chain of parent/child sensor nodes can happen, so as long as there is a sensor node within range, sensor node that has the best RSSI is chosen to forward the data to the coordinator node.
- **Coordinator or Sensor Selection:** in this scenario, if within the range of the sensor nodes we have a mix of coordinator nodes and sensor nodes, the sensor node chooses as its parent the coordinator node, regardless if a sensor node that has a better RSSI than the coordinator is present within its range.
- **Disappearing Parents:** in this scenario, we might encounter a failure/disappearance of a parent node, this parent node can be a coordinator or a sensor node. In order for a sensor node to choose a different parent if the initial parent failed to stay alive, the sensor senses that its parent is lost, removes that parent from the list of parents, then it starts again the process of choosing a parent within its range.
- **Disappearing Sensors:** in this scenario, it is the other way around, where the parent either a coordinator or a sensor detects that a child has been lost. To be able to know that information, the parent node sends a message to its child up to 3 times until a response is sent back, if no responses were detected

by the parent, this sensor node is considered as lost and the parent node removes this lost node form its list of children nodes

0.3 Routing Functionality in Sensor Networks

0.3.1 Implementation

The Border Router manages the routing within the network using a custom routing protocol that organizes the coordinator nodes into a tree structure, with the Border Router at the root.

0.3.2 Functionality Description

The Border Router manages the routing functionality as follows:

1. Coordinator nodes organize themselves into a tree structure, with the Border Router as the root.
2. Upon joining the network, new coordinator nodes send a 'HELLO' message to the Border Router.
3. The Border Router, upon receiving the 'HELLO' message, acknowledges the new coordinator node and adds it to the network tree.
4. If a coordinator node stops responding at a 'CLOCK_REQUEST' or disappears from the network, the Border Router updates its network tree and reassigns all coordinator identifiers sending them a new 'HELLO' message.

This routing protocol ensures efficient data transmission and provides robustness to the network by allowing it to adapt to changes dynamically.

0.3.3 Berkeley Time Synchronization Algorithm

The Berkeley Time Synchronization Algorithm is implemented to maintain time synchronization between the Border Router and the coordinator nodes. The Border Router acts as the master node in this algorithm. The algorithm works as follows:

1. The Border Router queries each coordinator node for its time.
2. Each coordinator node responds with its current time.
3. The Border Router calculates the average time from the received times and its own.
4. The Border Router then sends the average time back to each coordinator node.
5. Each coordinator node adjusts its clock to the received average time.

This algorithm ensures that all coordinator nodes are synchronized with the Border Router, facilitating the efficient scheduling of data transmission slots.

0.3.4 Network Architecture

The network architecture consists of a Border Router, multiple coordinator nodes, and sensor nodes. The Border Router serves as the root of the network tree. Each coordinator node, acting as an intermediary, manages the transmission of data from the sensor nodes to the Border Router. The Border Router then forwards the data to the external server for further analysis.

This architecture is scalable, allowing new coordinator nodes to join the network at any time, and robust, as the routing protocol can adapt to changes within the network, such as the addition or removal of coordinator nodes.

0.4 Coordinator/Sensor nodes

In order to make the transmission of the data from the sensor, we implemented a version of time slot technique that allow sensors node to forward their data to the coordinator during a certain interval of time. This sensor slot is calculated thanks to the window of the coordinator sent by the border routeur divided by the number of sensor that the coordinator has.

The coordinator send a message to the sensors during their respective time slot that allow them to send data.By doing that we don't need to update and synchronize the clock of the sensor like for the coordinators. Since its the coordinator itself that that trigger the time slot of the sensors and the sensors doesn't know his time slot. If a sensor doesn't answer at the end of his time slot, it is considered like not alive device and his removed from the coordinator child. Since the bordeur routeur is storing the whole infrastructure, he is also notified that the sensors is not alive anymore.

0.5 Testing Code

To facilitate code testing, we have created a sample simulation in Cooja called **finalSimu.csc** located in the project's directory.

0.6 Conclusion

To conclude, this project focused on exploring the functionalities of a sensor network for a building management system. We had first to implement the messages format in the network, the implementation process allowed the border routeur, coordinators and sensors to communicate effectively and transfer messages based on different scenarios.

Furthermore, we delved into the routing functionality within the sensor network, where the Border Router played a crucial role in managing the network tree and organizing coordinator nodes. The custom routing protocol ensured efficient data transmission, and the Berkeley Time Synchronization Algorithm facilitated time synchronization between the Border Router and coordinator nodes.

The network architecture consisted of the Border Router, coordinator nodes, and sensor nodes, working together to transmit data from sensors to the external server. By implementing a time slot technique, the sensor nodes were able to forward their data to the coordinators during designated intervals, ensuring effective data transfer.

In conclusion, this project has provided a solid foundation for understanding and implementing sensor networks in building management systems, showcasing the potential for advanced monitoring and data collection in various domains.