UCLouvain

LINFO2146 - Mobile and Embedded Computing

Final Project

Amirhossein Mohammadkarimi - NOMA: 10372201

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

Academic Year: 2022-2023

1 Introduction

Abstract

his project focuses on a fictional building management system with a large number of wireless sensor nodes deployed throughout different areas of a building. Each sensor node is equipped with a motion sensor to measure activity in its coverage area. Whenever motion is detected, a counter inside the sensor node is incremented. The sensor nodes communicate wirelessly using the IEEE 802.15.4 protocol and form a multi-hop network. To efficiently manage the network, coordinator nodes are introduced, which are devices directly connected to the border router. Their role is to schedule data transmission from a set of sensor nodes, allowing only one node to send data at a time.

Following is the link to my GitHub repository, which includes all commented source code of the code for my nodes (sensor nodes, coordinator nodes, border router) and the server:

`https://github.com/amir-karimy/LINFO2146.git`

The project involves the following components:

2 Sensor Nodes:

2.1 Explanation

These wireless devices continuously monitor motion and send their counter values to an external server for further analysis. The sensor nodes need to organize themselves using a scheduling mechanism implemented in the C script.

2.2 Implementation

Upon initialization, the sensor node generates a random address for the coordinator node to which it will send its data. This random generation is only for demonstration purposes, and in a real-world scenario, the coordinator address would likely be predefined or determined through a network protocol.
Once initialized, the sensor node enters a continuous operation loop where it first waits for the sensor reading period to expire. This period can be considered the time it takes for the sensor node to acquire new data. After this period, the sensor node calculates a new time slot based on the current time.
The sensor node then waits for its assigned time slot before sending its data to the coordinator. The time slot mechanism ensures that sensor nodes do not transmit data simultaneously, reducing the chance of data collisions on the network. Upon reaching its time slot, the sensor node generates random sensor data and sends this data to the coordinator node.

3 Border Router:

3.1 Explanation

The border router is the central node in the network and acts as a bridge between the wireless network and an external server. It assigns time-slots to coordinator nodes, synchronizes time using the Berkeley Time Synchronization Algorithm, and facilitates communication with the server.

3.2 Implementation

The Border Router code starts by setting up an input callback to handle incoming packets and a random time slot for sending packets. Within a loop, it checks if a timer has expired. If the periodic

timer expires, it triggers the backoff timer, which after a delay, sends a packet to the server. If the time window timer expires, it updates the time slot and sends a new packet to the server.

4 Coordinator Nodes:

4.1 Explanation

These nodes are responsible for managing a set of sensor nodes and ensuring coordinated data transmission. They receive time-slots from the border router and forward data from the sensor nodes during their assigned time-slots. The scheduling mechanism can be implemented using either a polling or time slot approach.

4.2 Implementation

I included necessary header files for the Contiki OS, nullnet networking, logging, packet buffers, timers, the network stack, and a random library. Logging Setup sets up a logging module named "Coordinator Node" with an info-level log level. A constant is defined for controlling the intervals between sending packets (SEND_INTERVAL). The script defines a struct called `time_slot_t`, which holds the network address (`linkaddr_t type`) of a coordinator node and a time slot (`uint16_t type`). An array of this struct, time_slots, is created to store multiple time slots. The `input_callback` function is currently a placeholder for implementing logic on receiving data. The `send_packet` function assigns a time slot to the nullnet buffer and sends it to the border router. A process called `coordinator_node_process` is defined and is set to autostart. Inside the `coordinator_node_process` thread, process Thread starts by setting the nullnet input callback to `input_callback` and setting a periodic timer to `SEND_INTERVAL`. Then, within an infinite loop, it waits for the `periodic_timer` to expire. When the timer expires, it randomly chooses a time slot, assigns it to a coordinator in `time_slots`, sets a callback timer `backoff_timer` to send the packet, and then resets the `periodic_timer`.

5 Server:

An external application running on a Linux machine receives messages from the sensor nodes via the border router. The server is implemented in Python. Bridging functionality is needed between the border router and the server, and a Serial Socket connection is established to facilitate communication.

6 Challenges

I tried to prevent the case where sensor nodes send message to each other, and tried to implement it in a way that nodes only send message coordinator node and not to each other whenever the coordinator node is close to them with a strong signal. However, I could not succeed to do so.

7 Conclusion

In the provided network, the sensor node's operation is relatively simple, as its primary purpose is to gather data and send it to a coordinator node following a specific schedule. The use of time slots is a common method to prevent data collisions and maintain a smooth and efficient data flow in the network. Despite its simplicity, this script contains the essential components of a sensor node and can be further developed to suit more specific applications.