LINFO2146 - Group Project

Nathan Collard, Liza Denis, Nijki Esemble

May 2024

1 Introduction

In this group project, we developed a system aimed at controlling the climatic conditions within greenhouses using a network of Internet of Things (IoT) devices. This system incorporates various components including light sensors, lights, irrigation controllers, and a mobile terminal, all interconnected through wireless technology.

The design and implementation processes are detailed in the following sections. We have leveraged advanced networking and control strategies to ensure efficient operation and management of greenhouse environments. The system is designed to be scalable, allowing for the integration of additional sensors and control devices as needed.

We provided you with Cooja Simulation, a basic and a complexe one. Keep in mind that the may take up to 30 seconds for the network to converge at the start.

For detailed insight into our development process and to access the complete source code, please visit our GitHub repository at the link provided below:

`https://github.com/Lizadns/ProjectLinfo2146.git`

2 Message format in the sensor network

In our smart greenhouse management system, we utilize a custom message format to facilitate communication between IoT devices and the server. The structure of the message is defined using a typedef struct named network_packet_t. Here's a detailed description of each field in the message format:

```
typedef struct {
    linkaddr_t src_addr;            // Source address
    linkaddr_t dst_addr;            // Destination address
    uint8_t src_type;               // Source type
    uint8_t dst_type;               // Destination type
    uint8_t type;                   // Message type: 0 = Routing, 2 = Data
    int8_t distance_to_gateway;     // Distance to the gateway
```

```
        char payload[64];               // Message payload
} network_packet_t;
```

In the following diagram, we present the structure of the message format used in Contiki OS. Each field in the message format is represented along with its respective size in bytes. This format facilitates communication between nodes in a wireless sensor network.

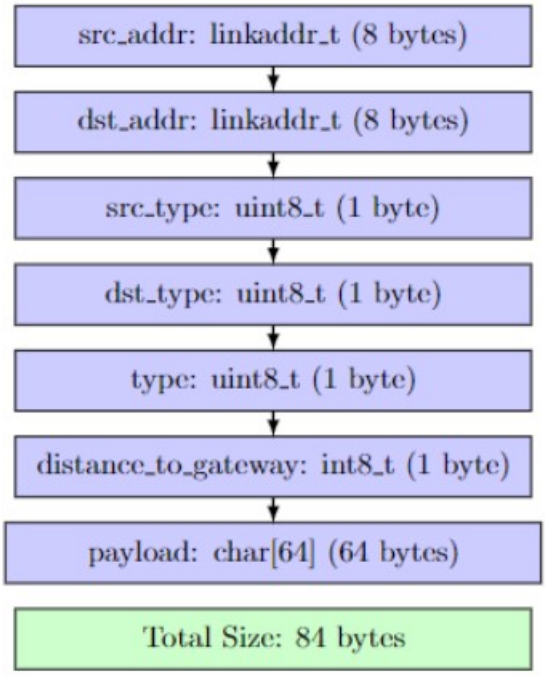

Figure 1: Size of the message format

3 Routing in the sensor network

3.1 Routing Protocol Implementation

The routing protocol operates on a simplified version of the tree-based routing topology, which does not use traditional network stacks like RPL (Routing Protocol for Low-Power and Lossy Networks). Instead, it requires each node to manage its routing independently, leveraging custom-built net libraries. These libraries facilitate specific routing functionalities tailored to our network's unique requirements, enabling more efficient management of data flow and node connectivity within the system.

3.2 Node Types and Roles

- **Gateway Node:** Serves as the root of the network tree and handles data aggregation and interaction with the server.
- **Sub-Gateway Nodes:** Act as intermediary nodes that aggregate data from sensor nodes before passing it to the gateway.
- **Sensor Nodes:** These are endpoint devices that collect environmental data and follow commands from the gateway or sub-gateway.

3.3 Dynamic Parent Selection

- Each node, upon activation or during regular intervals, broadcasts a "Node Hello" packet, which contains its identifier, type, signal strength, and distance from the gateway.
- Upon receiving multiple "Hello" packets, a node selects its parent based on the signal strength, the node type and the distance to the gateway, prioritizing sub-gateways over other sensor nodes to ensure a more stable connection to the network backbone.
- This mechanism allows the network to self-organize and adapt to changes, such as the addition of new nodes or the failure of existing ones.

3.4 Key Functions from the Custom Net Library

This file forms the core of the routing logic. Here are some critical functions and their roles:

- `set_radio_channel()`: Sets the communication channel for the node, ensuring that all nodes operate on a consistent frequency to avoid interference.
- `send_node_hello()` and `send_node_hello_response()`: These functions handle the broadcasting and responding of "Hello" messages. These messages are crucial for new nodes to announce their presence and for existing nodes to maintain their network awareness.
- `assign_parent()` and `leave_parent()`: These functions manage the parent-child relationships in the network. `assign_parent()` is called when a node decides on its best parent based on signal strength and node type, optimizing the routing path to the gateway. `leave_parent()` is used when a node needs to disconnect from its current parent, either due to a better parent becoming available or the current parent going offline.
- `assign_child()` and `unassign_child()`: Manage the list of child nodes for each parent node, ensuring that the routing tree remains updated as nodes join or leave the network.

3.5 Multicasting vs Unicast

- **Multicasting**: Utilized mainly by the gateway and sub-gateway nodes for commands that should be executed by multiple nodes simultaneously, such as activating irrigation or lighting systems. This is managed by checking if the destination address matches the multicast address and then forwarding the message to all child nodes.
- **Unicast**: Used for specific commands or data responses, where messages need to be directed to a specific node. This includes responses to the server's requests or command acknowledgments.

3.6 Routing Process

1. **Initialization:** Each node initializes its network settings and starts by sending a "Node Hello" using `send_node_hello()`. This helps in establishing its presence in the network.
2. **Parent Selection:** Upon receiving hello packets from multiple nodes, each node evaluates potential parents using `assign_parent()`, choosing based on signal strength and the hierarchical position (prefer sub-gateways over other sensor nodes).
3. **Data Transmission:** Once the network hierarchy is established, nodes begin their designated operations (e.g., sensing, actuation). Data and commands are routed through the tree, from sensors up to the sub-gateways, and then to the gateway.
4. **Maintaining Connectivity:** Nodes periodically resend "Hello" messages to adapt to any changes in the network topology. This dynamic adaptation helps in maintaining network efficiency and robustness.

3.7 Challenges

- **Dynamic Network Conditions:** The system effectively adapts to changes such as node failures or the introduction of new nodes without requiring manual reconfiguration.
- **Energy Efficiency:** By optimizing the frequency of routing messages and using intelligent parent selection, the network minimizes unnecessary communications, which conserves energy.

4 Implementation of the multicasting

4.1 Overview

Multicasting in your network facilitates efficient group communication by allowing a single message to be received by multiple nodes simultaneously. This

is particularly important for operations such as activating all light bulbs or irrigation systems in a group of greenhouses without the need to send individual messages to each node, thereby reducing network traffic and conserving energy.

4.2 Multicast Address Configuration

- **Static Multicast Address**: In your implementation, a static multicast address (`0xFFFF`) is defined in `network_greenhouse.h`. This address is recognized by all nodes as the destination address for multicast packets, which are intended for multiple receivers within the network.

4.3 Code Implementation

- **Address Handling**: The multicast address is set up as a constant in the network configuration header, ensuring that it is accessible throughout the network codebase.
- **Packet Handling**: When creating packets intended for multicasting, nodes set the destination address of the packet to the multicast address. This indicates that the packet should be processed by multiple nodes.

4.4 Node-Specific Handling

- **Gateway and Sub-Gateway Nodes**: These nodes play a crucial role in the multicasting process. When they receive a packet with a multicast destination address, they forward this packet to all connected child nodes. This is done by iterating over the list of child nodes and sending the packet to each node using the network stack's output function.
- **Sensor Nodes**: Upon receiving a multicast packet, sensor nodes check the destination type within the packet (e.g., irrigation system, light bulbs). If the type matches their function, they execute the command encoded in the packet, such as turning on the irrigation system or the lights.

5 Description of the server

Our server is written in Python and is designed to manage communication with the sensor nodes of the greenhouse system.
It communicates every 120 seconds with all the irrigation systems in the greenhouse using the multi casting, asking them to switch on. This periodic function runs on a separate thread.
The main loop continuously reads the messages sent to it by the gateway. In order to do this, the gateway prints them on the serial interface.
Using this main loop, the server can confirm that it has received acknowledgement from all the irrigation systems. It can also extract the light intensity and source (greenhouse ID) from the messages coming from the light sensor to check

whether the light intensity in a greenhouse is below a certain threshold, 65 degrees in our case. If it is, the server will send a command to turn on the lights in that corresponding greenhouse.

6 Description of sensor nodes

6.1 Irrigation System

Irrigation is activated when it receives a message from the server. It then sends an acknowledgement to the server, sprays the plants in the greenhouse for 20 seconds and finally sends a message to the server to say that irrigation has stopped.

6.2 Mobile Terminal

When the mobile terminal is inserted in a greenhouse, it sends 3 control messages to the lighting modes.

6.3 Light Sensor

The light sensor produces random value corresponding to the light intensity in the greenhouse. When it has find it's parent, a subgateway, it can send the information to the server.
When it's received a message from the mobile terminal, it's answer with it's current light intensity.

6.4 Lights

When they receive the message from the server to switch on, they light up for 29 seconds and then switch off.

7 Conclusion

Our project has created a robust system to effectively manage greenhouse conditions. Using custom messaging, dynamic routing and multicasting, we enabled wireless communication between devices.