Lab Report: An Analysis of painlessMesh Network Behavior

Name: Fatema tahsin anamika

ID:2021-3-60-170

An analysis into the core functionalities of the painlessMesh library, focusing on network event callbacks and targeted messaging protocols.

Task 1: Network Callback Events

This initial task focused on interpreting the automated serial output generated by the painlessMesh library to understand its event-driven nature.

Analysis of Callback Functions

1. **New Connection**
   - **Interpretation:** This callback serves as a notification that the local node has successfully formed a direct, point-to-point wireless link with another device. It is the most fundamental connection event, confirming that two nodes are now direct neighbors and can exchange data without an intermediary.
   - **Example Log:**
     --> startHere: New Connection, nodeId = 3486689028
   - **Significance:** The appearance of this log confirms the successful establishment of a new route in the mesh's physical layer.
2. **Changed Connections** 🌐
   - **Interpretation:** This event provides a higher-level alert to all nodes that the overall network structure has been modified. It is triggered by any change in the mesh's composition, such as a node joining or departing. This function is essential for the network's self-healing capability, prompting each node to re-evaluate its knowledge of the network topology.
   - **Example Log:**
     Changed connections
   - **Significance:** This callback is the trigger for recalculating routing tables across the entire mesh, ensuring network resilience and adaptability.
3. **Adjusted Time** 🕒
   - **Interpretation:** This callback indicates that the local node has successfully synchronized its internal clock with the network's master time. Within the mesh, one node is dynamically chosen as the authoritative time source. All other

nodes then align their clocks to this source, correcting for any local drift.

- **Example Log:**
  Adjusted time 1723404732. Offset = -3

- **Significance:** Time synchronization is vital for coordinating actions, creating accurate event timestamps, and managing network-wide scheduled tasks.

Task 2: Direct Messaging

The objective was to adapt the provided broadcast-based code to utilize unicast messaging, where a message is sent to a single, predetermined node.

Firmware Modification

The sendMessage() function was refactored. The mesh.sendBroadcast() call was replaced with mesh.sendSingle(targetNodeId, msg). To enhance robustness, a preliminary check, mesh.isConnected(targetNodeId), was introduced to confirm the target's presence in the network before attempting to transmit data.

```
// Define the specific Node ID you want to send a message to.
uint32_t targetNodeId = 3486689028;

void sendMessage() {
  String msg = "Direct message from Node ";
  msg += mesh.getNodeId();

  // Confirm the destination is reachable before sending
  if (mesh.isConnected(targetNodeId)) {
    // Transmit the packet to a single destination
    mesh.sendSingle(targetNodeId, msg);
    Serial.printf("--> Sent direct message to %u\n", targetNodeId);
  } else {
    Serial.printf("--> Target node %u not found. Message not sent.\n", targetNodeId);
  }
}
```

Validation of Results

The experiment was conducted with a sending node (ID: 2484433821) and a receiving node (ID: 3486689028).

- **Sender Serial Output:**
  --> Sent direct message to 3486689028

- **Receiver Serial Output:**
  Received from 2484433821: Direct message from Node 2484433821

Conclusion: The results validate the correct implementation of unicast communication. The message was exclusively delivered to the designated target, demonstrating a more efficient communication method compared to a network-wide broadcast.