Module 10 - Mesh

10.1 Introduction

Module 10: Mesh

Author: YP

Learning Objectives

After completing this module, students are expected to be able to:

- **Explain** the concept, advantages, and basic architecture of IoT mesh networks.
- **Implement** a simple mesh network using ESP32 with the `painlessMesh` library.
- **Build** a *gateway* (*Root Node*) to forward data from the mesh network to a server.

10.2 What is Mesh?

10.2.1 What is a Mesh Network?

image

A **mesh network** is a topology where each device (*node*) is interconnected, creating multiple paths for data. Unlike traditional networks that rely on a single central point (such as a router), a mesh network is **decentralized**.

How It Works

In a mesh network, each ESP32 node acts as both a sender and receiver, functioning as a repeater. This differs from a star network where communication only goes through one central hub. A mesh network is more decentralized, thus improving coverage and network reliability.

In this setup, one ESP32 acts as the root node, which connects the mesh network to an external network, while other nodes can serve as intermediate parents that forward data or as leaf nodes that only send and receive their own data. Data in the network is automatically routed through available nodes until it reaches the final destination, either another node within the mesh or an external network via the root node. This creates an efficient, flexible, and fault-tolerant network.

Key characteristics include:

- **Self-Healing**: If one *node* fails, data is automatically rerouted through another available path.
- **Wide Coverage**: Each *node* functions as a *repeater*, significantly extending signal range.

10.2.2 Topology Comparison: Mesh vs. Star

To understand the advantages, let’s compare it with the commonly used Star topology in home Wi-Fi.

- Mesh
  Mesh Topology
  Connects devices directly, creating multiple paths for data. A mesh network is more flexible than a star network and requires fewer gateways to communicate with the same number of devices. However, it is more complex and costly compared to a star network. Mesh networks also lack universal standards, which may cause compatibility issues between devices from different vendors.
- Star
  Star Topology
  Connects devices through a central hub. A star network is simpler and cheaper than a mesh network but depends heavily on the central hub. Device failures in a star network do

not affect the rest of the system, but if the hub fails, the entire network goes down. Star networks are best suited for devices that need to communicate directly with a central node, such as office equipment, security cameras, and medical devices.

Component	Mesh Topology	Star Topology
Resilience	**Very High**. Supports self-healing.	**Low**. If the hub/router fails, the entire network fails.
Coverage	**Wide and flexible**. Easy to expand.	**Limited** by the hub/router's range.
Complexity	More complex to configure.	Simple and easy to set up.

10.2.3 Types of Nodes in a Mesh Network

In this lab, we will learn about 4 types of nodes:

- **Root Node**
  The root node is the highest-level node in a Wi-Fi mesh network and acts as the only link between the mesh network and the external IP network. It connects directly to a conventional Wi-Fi router and forwards data packets between the external network and the mesh nodes. There must only be one root node in a mesh network, and it can only have one upstream connection—to the router. The root node is crucial in ensuring that all mesh network data can be accessed by external devices.
- **Leaf Nodes**
  Leaf nodes are nodes that have no child nodes (no downstream connections). They can only send or receive their own data packets and do not forward data from other nodes. Typically, leaf nodes are located at the farthest edge of the mesh where no new downstream connections are possible. If a node only has a station interface (station-only node) and lacks a softAP interface, it is designated as a leaf node since downstream connections require a softAP.
- **Intermediate Parent Nodes**
  Intermediate parent nodes are neither root nor leaf nodes. They have one upstream connection (to a parent node) and may have multiple downstream connections (to child nodes). These nodes can send, receive, and forward data from both upstream and downstream connections. Unlike leaf nodes, they can still form downstream connections in the future. They serve as bridges in the mesh network, enabling data to flow between different layers.
- **Idle Nodes**
  Idle nodes are nodes that have not yet joined the mesh network. They attempt to establish an upstream connection with an existing intermediate parent node or try to become the root node if certain conditions are met (for example, when no root node exists in the network). Idle nodes remain passive until they successfully join or are integrated into the mesh network.

10.3 Example Code

10.3.1 Root Node

```
#include <Arduino.h>
#include <painlessMesh.h>
#include <WiFi.h>

// --- Konfigurasi Jaringan ---
#define MESH_PREFIX     "jaringan_mesh_saya" // HARUS SAMA dengan semua node
#define MESH_PASSWORD   "password_mesh"      // HARUS SAMA dengan semua node
#define MESH_PORT       5555

painlessMesh mesh;

// Callback ketika Root menerima pesan dari Leaf
void receivedCallback(uint32_t from, String &msg) {
  Serial.printf("Pesan diterima dari node %u: %s\n", from, msg.c_str());
}

void setup() {
  Serial.begin(115200);

  // Inisialisasi Mesh
  mesh.setDebugMsgTypes(ERROR | STARTUP | CONNECTION);
  mesh.init(MESH_PREFIX, MESH_PASSWORD, MESH_PORT);

  // Set sebagai ROOT
  mesh.setRoot(true);
  mesh.setContainsRoot(true);

  // Pasang callback untuk menerima pesan
  mesh.onReceive(&receivedCallback);

  Serial.println("Mesh dimulai sebagai ROOT");
  Serial.print("Root ESP32 SoftAP IP: ");
  Serial.println(WiFi.softAPIP());
}
```

```
void loop() {
  mesh.update();
}
```

10.3.2 Leaf Node

```
#include <Arduino.h>
#include <painlessMesh.h>

// --- Konfigurasi Jaringan ---
#define MESH_PREFIX     "jaringan_mesh_saya" // HARUS SAMA dengan semua node
#define MESH_PASSWORD   "password_mesh"      // HARUS SAMA dengan semua node
#define MESH_PORT       5555

painlessMesh mesh;

// Ganti sesuai nomor leaf
String leafName = "Leaf1";

void sendMessage() {
  String msg = "Hello guys im " + leafName;
  mesh.sendBroadcast(msg);
  Serial.println("Sent: " + msg);
}

void setup() {
  Serial.begin(115200);
  mesh.setDebugMsgTypes(ERROR | STARTUP | CONNECTION);
  mesh.init(MESH_PREFIX, MESH_PASSWORD, MESH_PORT);
}

void loop() {
  mesh.update();
  static unsigned long lastSend = 0;
  if (millis() - lastSend > 5000) {  // kirim tiap 5 detik
    lastSend = millis();
    sendMessage();
  }
```

```
}
```

10.3.3 Intermediate Parent Node

```
#include <Arduino.h>
#include <painlessMesh.h>

// --- Konfigurasi Jaringan ---
#define MESH_PREFIX     "jaringan_mesh_saya" // HARUS SAMA dengan semua node
#define MESH_PASSWORD   "password_mesh"      // HARUS SAMA dengan semua node
#define MESH_PORT       5555

painlessMesh mesh;

// Callback ketika node menerima pesan
void receivedCallback(uint32_t from, String &msg) {
  Serial.printf("Pesan diterima dari %u: %s\n", from, msg.c_str());
  // Intermediate node tidak perlu kirim ke server, cukup teruskan pesan
  mesh.sendBroadcast(msg);
}

// Callback saat node terhubung
void newConnectionCallback(uint32_t nodeId) {
  Serial.printf("Node baru terhubung: %u\n", nodeId);
}

// Callback saat node terputus
void changedConnectionCallback() {
  Serial.println("Perubahan koneksi terjadi!");
}

// Callback saat node time sync
void nodeTimeAdjustedCallback(int32_t offset) {
  Serial.printf("Waktu sinkronisasi, offset = %d\n", offset);
}

void setup() {
  Serial.begin(115200);
```

```
  // Inisialisasi mesh
  mesh.setDebugMsgTypes(ERROR | STARTUP);
  mesh.init(MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT);

  // Callback
  mesh.onReceive(&receivedCallback);
  mesh.onNewConnection(&newConnectionCallback);
  mesh.onChangedConnections(&changedConnectionCallback);
  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
}

void loop() {
  mesh.update();
}
```

10.3.4 Idle Node

There is no code because this is just a status if the node is not yet connected to the mesh.