

# Mesh Networking with NodeMCU ESP8266 and painlessMesh

Lab-04

Name: Md. Samir Hossain  
ID: 2022-3-60-161  
Course: CSE406 (Internet of Things)  
Section: 01  
Semester: Summer-2025

Dr. Raihan Ul Islam  
Associate Professor  
Department of Computer Science & Engineering

August 15, 2025

# Contents

<b>1</b>	<b>PainlessMesh Callback Messages Explanation</b>	<b>2</b>
1.1	New Connection . . . . .	2
1.2	Connection Change . . . . .	2
1.3	Adjusted Time . . . . .	2
1.4	Observed Serial Logs . . . . .	2
<b>2</b>	<b>Direct Message to a Specific Node in PainlessMesh</b>	<b>4</b>
2.1	Modifications to the Code . . . . .	4
<b>3</b>	<b>Multi-Hop Messaging with <code>painlessMesh</code> (<math>A \rightarrow B \rightarrow C</math>)</b>	<b>6</b>
3.1	Objective . . . . .	6
3.2	Hardware & Network Setup . . . . .	6
3.3	Procedure . . . . .	6
3.4	Brief code explanation . . . . .	7
3.4.1	Common library & debug . . . . .	7
3.4.2	Node A . . . . .	8
3.4.3	Node B — <code>NodeMCU-B-Middle.ino</code> (middle / relay) . . . . .	8
3.4.4	Node C — <code>NodeMCU-C-Receiver.ino</code> (final receiver) . . . . .	9
3.5	How message flow looks . . . . .	9
<b>4</b>	<b>Mesh vs. Star Topology: Advantages and Applications</b>	<b>9</b>
4.1	Advantages of Mesh over Star Topology . . . . .	9
4.2	Potential Applications . . . . .	9

# 1 PainlessMesh Callback Messages Explanation

This report explains the meanings of the callback messages observed in the provided PainlessMesh code, based on the serial monitor output and the PainlessMesh library documentation. The explanations are provided in my own words, with examples from the given serial logs where applicable.

## 1.1 New Connection

The `newConnectionCallback` is triggered when a new node establishes a direct connection with the current node in the mesh network. This event indicates that the mesh is expanding as a new device joins and becomes part of the network. The callback provides the `nodeId` of the newly connected node, allowing the current node to recognize and log the new participant in the mesh.

- Example from Serial Log: A log for this event would look like: `--> startHere: New Connection, nodeId = 3299373438`. This would indicate that a node with the ID 3299373438 has just joined and formed a direct connection with the current node.

## 1.2 Connection Change

The `changedConnectionCallback` is invoked whenever there is a change in the network topology, such as a node joining, leaving, or a connection link between nodes being established or broken. This event prompts the mesh network to update its internal routing table to reflect the new structure, ensuring efficient message routing across the network.

- Example from Serial Log: A log would be: `“Changed connections”`. This message would appear when, for instance, the node with ID 3050603722 joins or leaves, causing the mesh to reconfigure its connections.

## 1.3 Adjusted Time

The `nodeTimeAdjustedCallback` is called when the node's internal clock is synchronized with the mesh network's global time. PainlessMesh uses a mechanism similar to Network Time Protocol (NTP) to align the clocks of all nodes in the mesh, ensuring coordinated timing for tasks like scheduling messages. The callback provides the adjusted time and the offset applied to synchronize the node's clock.

- Example from Serial Log: A log would look like: `“Adjusted time 123456789. Offset = -200”`. This indicates that the node's clock was adjusted to the mesh time 123456789 with an offset of -200 microseconds to align with the network.

## 1.4 Observed Serial Logs

The provided serial logs show message reception events, which are handled by the `receivedCallback` function. These logs indicate communication between nodes in the mesh:

- From 1st ESP: “startHere: Received from 3299373438 msg=Hello from node 3299373438”  
This log shows that the first ESP received a broadcast message from the node with ID 3299373438, containing the message “Hello from node 3299373438”.
- From 2nd ESP: “startHere: Received from 3050603722 msg=Hello from node 3050603722”  
This log indicates that the second ESP received a broadcast message from the node with ID 3050603722, containing the message “Hello from node 3050603722”.

Figure 1: Serial Log from id: 3299373438

Figure 2: Serial Log from id: 3050603722

## 2 Direct Message to a Specific Node in PainlessMesh

This report details the modifications made to the provided PainlessMesh code to send a direct message to a specific node using `mesh.sendSingle()` instead of broadcasting to all nodes. It explains the changes to the `sendMessage()` function, how the code handles cases where the target node is not connected, and the process for testing and verifying the direct messaging functionality using serial monitor output. The code for both the first and second ESP nodes is included to demonstrate the setup for a two-node mesh network.

### 2.1 Modifications to the Code

The primary change was made to the `sendMessage()` function to enable direct messaging to a specific node. The original code used `mesh.sendBroadcast()` to send messages to all nodes in the mesh network. For Task 2, this was replaced with `mesh.sendSingle(targetNodeId, msg)` to send a message to a single node identified by its `nodeId`. Below are the specific changes:

1. Target Node ID:

- A hardcoded `targetNodeId` (e.g., 3299373438 for the first ESP, 3050603722 for the second ESP) I added to specify the recipient node. This was based on the node IDs observed in the provided serial logs.
- The task allowed for hardcoding the node ID, so no serial input was implemented, though this could be added for dynamic selection.

2. Custom Message:

- The message content was updated to “Direct hello from node [ID]” for the first ESP and “Direct greeting from node [ID]” for the second ESP to distinguish messages in the serial logs.
- The message includes the sender’s `nodeId` using `mesh.getNodeId()` for identification.

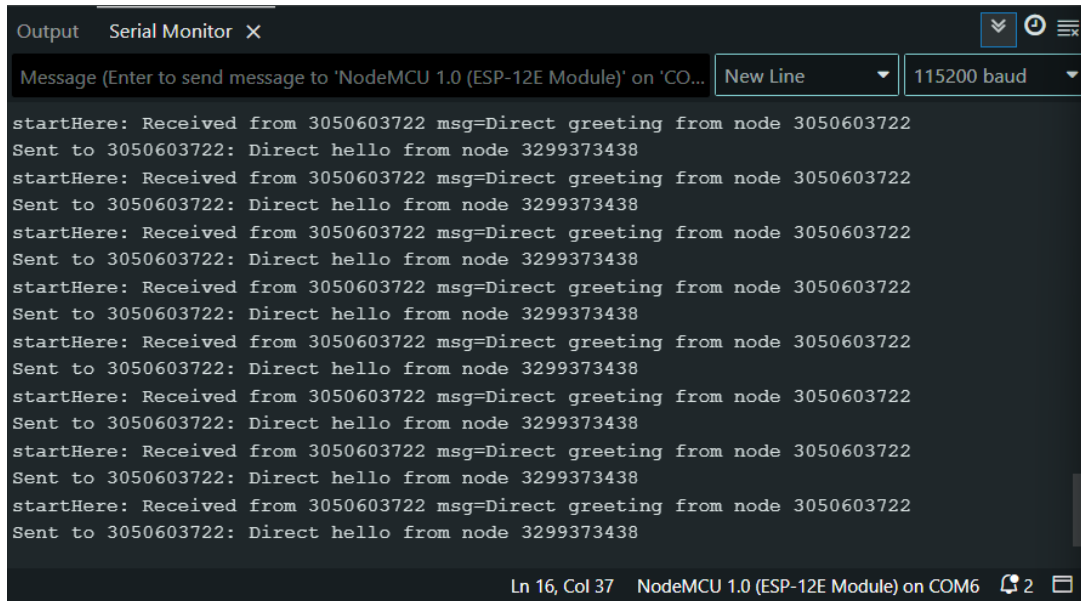
3. Connection Check:

- Before sending the message, the code checks if the target node is connected using `mesh.isConnected(targetNodeId)`.
- If the target node is connected, the message is sent, and a confirmation is printed to the serial monitor (e.g., “Sent to [ID]: [message]”).
- If the target node is not connected, a message is logged (e.g., “Target node [ID] is not connected”), preventing unnecessary send attempts.

4. No Changes to Other Functions:

- The `setup()`, `loop()`, and callback functions (`receivedCallback`, `newConnectionCallback`, `changedConnectionCallback`, `nodeTimeAdjustedCallback`) were left unchanged, as they handle network setup, message reception, and connection events, which are still relevant for direct messaging.

- The receivedCallback logs incoming messages, allowing verification of direct messages on the target node's serial monitor.

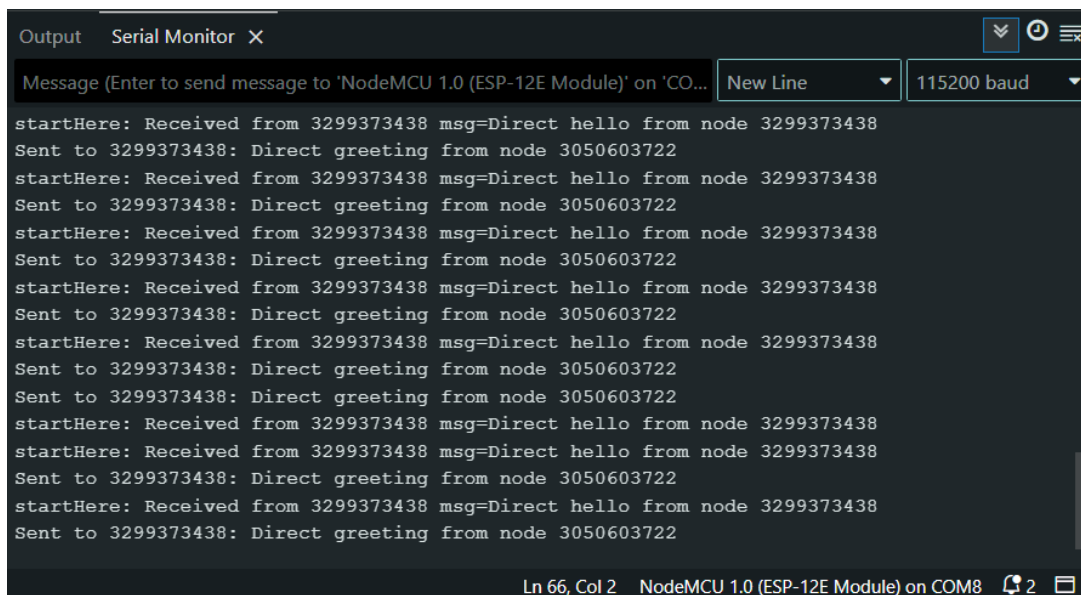


```

Output  Serial Monitor X
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'CO... New Line 115200 baud
startHere: Received from 3050603722 msg=Direct greeting from node 3050603722
Sent to 3050603722: Direct hello from node 3299373438
startHere: Received from 3050603722 msg=Direct greeting from node 3050603722
Sent to 3050603722: Direct hello from node 3299373438
startHere: Received from 3050603722 msg=Direct greeting from node 3050603722
Sent to 3050603722: Direct hello from node 3299373438
startHere: Received from 3050603722 msg=Direct greeting from node 3050603722
Sent to 3050603722: Direct hello from node 3299373438
startHere: Received from 3050603722 msg=Direct greeting from node 3050603722
Sent to 3050603722: Direct hello from node 3299373438
startHere: Received from 3050603722 msg=Direct greeting from node 3050603722
Sent to 3050603722: Direct hello from node 3299373438
startHere: Received from 3050603722 msg=Direct greeting from node 3050603722
Sent to 3050603722: Direct hello from node 3299373438
startHere: Received from 3050603722 msg=Direct greeting from node 3050603722
Sent to 3050603722: Direct hello from node 3299373438
Ln 16, Col 37 NodeMCU 1.0 (ESP-12E Module) on COM6 2

```

Figure 3: Serial Output for Direct Messaging from id: 3299373438



```

Output  Serial Monitor X
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'CO... New Line 115200 baud
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
startHere: Received from 3299373438 msg=Direct hello from node 3299373438
Sent to 3299373438: Direct greeting from node 3050603722
Ln 66, Col 2 NodeMCU 1.0 (ESP-12E Module) on COM8 2

```

Figure 4: Modified Code Example from id: 3050603722

## 3 Multi-Hop Messaging with painlessMesh ( $A \rightarrow B \rightarrow C$ )

### 3.1 Objective

Demonstrate that a message from Node A can be relayed to Node C via Node B, explain the procedure, and briefly describe how the code implements the behavior. This report summarizes the test steps, the role of each node, what the sketches do, and how to interpret the serial outputs (your screenshots).

### 3.2 Hardware & Network Setup

- Mesh name: CSE406
- Password: 12345678
- Port: 5555
- Node IDs:
  - Node A (sender) - 3299373438
  - Node B (middle / relay) - 4206966532
  - Node C (receiver) - 3050603722
- Physical arrangement for the demo: Node B placed between A and C (or use the forced-forward code so B relays even if A and C are in range).
- Serial monitors opened on all three boards at 115200 baud.

### 3.3 Procedure

1. Flash each board with its respective .ino:
  - A: NodeMCU-A-Main.ino (dynamic choice based on measured delay).
  - B: NodeMCU-B-Middle.ino (handles forwarding requests).
  - C: NodeMCU-C-Receiver.ino (receives messages and replies with ACK).
2. Open Serial monitors for A, B and C.
3. Start the meshes and wait for them to form connections (watch STARTUP, CONNECTION debug messages).
4. Trigger A's send task (automatic in the code). Observe the sequence of messages on Serial for all three nodes.

```
Output Serial Monitor X
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM6')
New Line 115200 baud

CONNECTION: stationScan(): CSE406
CONNECTION: scanComplete(): Scan finished
CONNECTION: scanComplete(): -- > Cleared old APs.
CONNECTION: scanComplete(): num = 15
CONNECTION: found : CSE406, -27dBm
CONNECTION: found : CSE406, -26dBm
CONNECTION: Found 2 nodes
CONNECTION: connectToAP(): No unknown nodes found scan rate set to normal
[A] Requested delay measurement to 3050603722
COMMUNICATION: routePackage(): Recvrd from 3050603722: ("type":3,"dest":3299373438,"from":3050603722,"msg":{"type":2,"t0":141170253,"t1":141175243,"t2":141177486})
[A] Delay result node=3050603722 delay=5617 us
COMMUNICATION: sendSingle(): dest=3050603722 msg=Hello from A (policy)
[A] Direct send to 3050603722 (delay=5617): sendSingle returned true
COMMUNICATION: routePackage(): Recvrd from 3050603722: ("type":9,"dest":3299373438,"from":3050603722,"msg":"ACK from C to 3299373438 : got your message")
[A] Received from 3050603722 : ACK from C to 3299373438 : got your message
COMMUNICATION: routePackage(): Recvrd from 4206966532: ("nodeId":4206966532,"type":6,"dest":3299373438,"from":4206966532)
[A] Requested delay measurement to 3050603722
COMMUNICATION: routePackage(): Recvrd from 3050603722: ("type":3,"dest":3299373438,"from":3050603722,"msg":{"type":2,"t0":146170354,"t1":146174797,"t2":146177055})
[A] Delay result node=3050603722 delay=4709 us
COMMUNICATION: sendSingle(): dest=3050603722 msg=Hello from A (policy)
[A] Direct send to 3050603722 (delay=4709): sendSingle returned true
COMMUNICATION: routePackage(): Recvrd from 3050603722: ("type":9,"dest":3299373438,"from":3050603722,"msg":"ACK from C to 3299373438 : got your message")
[A] Received from 3050603722 : ACK from C to 3299373438 : got your message

Ln 84, Col 8 NodeMCU 1.0 (ESP-12E Module) on COM6
```

Figure 5: Node A (Main) Serial Log

```
Output Serial Monitor X
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM10')
New Line 115200 baud

COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":134488396,"t1":134713915,"t2":134716093})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":134488396,"t1":134713915,"t2":134716093})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("nodeId":3299373438,"subs":[{"nodeId":3050603722}], "type":6,"dest":4206966532,"from":3299373438)
CONNECTION: stationScan(): CSE406
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":0})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":134459197,"t1":134517238,"t2":134519445})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":134488396,"t1":134713915,"t2":134716093})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":134925133,"t1":134978994,"t2":134981164})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":135215263,"t1":135463548,"t2":135465675})
CONNECTION: scanComplete(): Scan finished
CONNECTION: scanComplete(): -- > Cleared old APs.
CONNECTION: scanComplete(): num = 11
CONNECTION: found : CSE406, -19dBm
CONNECTION: found : CSE406, -49dBm
CONNECTION: Found 2 nodes
CONNECTION: connectToAP(): Already connected, and no unknown nodes found: scan rate set to slow
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":135700227,"t1":135680705,"t2":135682836})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":135892956,"t1":135897269,"t2":135899407})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("nodeId":3299373438,"subs":[{"nodeId":3050603722}], "type":5,"dest":4206966532,"from":3299373438)
CONNECTION: stationScan(): CSE406
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":0})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":135981220,"t1":135985860,"t2":135988141})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":4,"dest":4206966532,"from":3299373438,"msg":{"type":2,"t0":135950967,"t1":135968957,"t2":135971045})
CONNECTION: scanComplete(): Scan finished
CONNECTION: scanComplete(): -- > Cleared old APs.
CONNECTION: scanComplete(): num = 10
CONNECTION: found : CSE406, -51dBm
CONNECTION: found : CSE406, -18dBm
CONNECTION: Found 2 nodes
CONNECTION: connectToAP(): Already connected, and no unknown nodes found: scan rate set to slow
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("nodeId":3299373438,"subs":[{"nodeId":3050603722}], "type":6,"dest":4206966532,"from":3299373438)

Ln 92, Col 1 NodeMCU 1.0 (ESP-12E Module) on COM10
```

Figure 6: Node B (Middle/Relay) Serial Log

```
Output Serial Monitor X
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM8')
New Line 115200 baud

CONNECTION: scanComplete(): num = 13
CONNECTION: found : CSE406, -52dBm
CONNECTION: found : CSE406, -26dBm
CONNECTION: Found 2 nodes
CONNECTION: connectToAP(): Already connected, and no unknown nodes found: scan rate set to slow
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":3,"dest":3050603722,"from":3299373438,"msg":{"type":1,"t0":161170237})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":9,"dest":3050603722,"from":3299373438,"msg":"Hello from A (policy)")
[C] Received from 3299373438 : Hello from A (policy)
COMMUNICATION: sendSingle(): dest=3299373438 msg=ACK from C to 3299373438 : got your message
[C] Sent ACK to 3299373438: sendSingle returned true
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":3,"dest":3050603722,"from":3299373438,"msg":{"type":1,"t0":166170290})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":9,"dest":3050603722,"from":3299373438,"msg":"Hello from A (policy)")
[C] Received from 3299373438 : Hello from A (policy)
COMMUNICATION: sendSingle(): dest=3299373438 msg=ACK from C to 3299373438 : got your message
[C] Sent ACK to 3299373438: sendSingle returned true
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":3,"dest":3050603722,"from":3299373438,"msg":{"type":1,"t0":171170321})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":9,"dest":3050603722,"from":3299373438,"msg":"Hello from A (policy)")
[C] Received from 3299373438 : Hello from A (policy)
COMMUNICATION: sendSingle(): dest=3299373438 msg=ACK from C to 3299373438 : got your message
[C] Sent ACK to 3299373438: sendSingle returned true
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":3,"dest":3050603722,"from":3299373438,"msg":{"type":1,"t0":176170306})
COMMUNICATION: routePackage(): Recvrd from 3299373438: ("type":9,"dest":3050603722,"from":3299373438,"msg":"Hello from A (policy)")
[C] Received from 3299373438 : Hello from A (policy)
COMMUNICATION: sendSingle(): dest=3299373438 msg=ACK from C to 3299373438 : got your message
[C] Sent ACK to 3299373438: sendSingle returned true

Ln 48, Col 1 NodeMCU 1.0 (ESP-12E Module) on COM8
```

Figure 7: Node C (Receiver) Serial Log

## 3.4 Brief code explanation

### 3.4.1 Common library & debug

All sketches use painlessMesh with:



```
mesh.setDebugMsgTypes(ERROR | STARTUP | CONNECTION | COMMUNICATION);
mesh.init(MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT);
mesh.update(); // in loop
```

These produce useful debug lines showing connections and internal forwarding.

### 3.4.2 Node A

#### NodeMCU-A-Main.ino (adaptive routing)

- Before sending, A calls `mesh.startDelayMeas(C_ID)` to measure network trip delay to C.
- Callback `onNodeDelayReceived` compares measured delay (microseconds) to a threshold (`DELAY_THRESHOLD_US`, e.g., 70 ms).
  - If `delay < threshold` → `mesh.sendSingle(C_ID, payload)` (direct send).
  - Else → `mesh.sendSingle(B_ID, "FORWARD_TO:...")` (ask B to forward).
- Purpose: show adaptive decision-making based on link quality.

Key functions used by A:

- `mesh.startDelayMeas(nodeId)` — initiates delay measurement
- `mesh.sendSingle(nodeId, payload)` — send to a specific node (mesh routes as needed)
- `mesh.onNodeDelayReceived(callback)` — receives delay result

### 3.4.3 Node B — NodeMCU-B-Middle.ino (middle / relay)

- `onReceive` inspects incoming strings. If a message starts with `FORWARD_TO:` it parses the destination and payload:
  - Format parsed: `FORWARD_TO:;dest_id;:;payload;`
  - Then calls `mesh.sendSingle(dest_id, payload)` to forward to C.
- Also logs topology and starts delay measurements for known nodes on connection changes to monitor link quality.
- Role summary: application-level relay — B explicitly performs the forward when asked; also participates in native `painlessMesh` routing (the library may forward packets itself if configured).

Important functions in B:

- `mesh.getNodeList()` — enumerate known nodes (used to print neighbors)
- `mesh.subConnectionJson(true)` — prints topology info (useful debug)
- `mesh.onReceive(&receivedCallback)` — handle incoming messages

### 3.4.4 Node C — NodeMCU-C-Receiver.ino (final receiver)

- Prints every received message.
- Sends an ACK back to the original sender using `mesh.sendSingle(from, ack)`. The ACK will be routed back via the mesh (through B if needed).
- Purpose: confirm successful delivery and allow A to detect replies.

## 3.5 How message flow looks

A --> B: Send "FORWARD\_TO:<C\_ID>:<payload>"</payload>

B: Parse and forward payload to C

B --> C: Send payload

C: Receive payload and send ACK back to A (routed via mesh, potentially through B)

C --> B: Send ACK

B --> A: Forward ACK

A: Receive ACK (confirmation of delivery)

## 4 Mesh vs. Star Topology: Advantages and Applications

### 4.1 Advantages of Mesh over Star Topology

Mesh topology offers several benefits compared to the traditional star topology:

- **Resilience and Fault Tolerance:** In a mesh network, nodes are interconnected, allowing data to take alternative routes if one link or node fails. This reduces the risk of total network failure, which is a common issue in star topology where the central hub is a single point of failure.
- **Scalability:** Adding new nodes to a mesh network does not significantly disrupt existing connections. In contrast, star topology often requires upgrades to the central hub for expansion.
- **Improved Coverage:** Mesh networks can cover larger areas by allowing nodes to relay messages, making them suitable for environments where direct connections to a central hub are not feasible.
- **Load Distribution:** Traffic can be dynamically routed through multiple paths, reducing congestion and improving overall network performance.

### 4.2 Potential Applications

Due to these advantages, mesh topology is widely used in modern applications:

- **Smart Homes:** Devices such as smart lights, thermostats, and security cameras can form a mesh network, ensuring reliable communication even if some devices are offline.

- **Sensor Networks:** Environmental monitoring, agricultural sensors, and industrial IoT systems often rely on mesh topology to ensure continuous data collection from remote or hard-to-reach sensors.
- **Disaster Recovery Networks:** In emergency situations where infrastructure is damaged, mesh networks can quickly establish communication links without relying on a central hub.
- **Military and Field Operations:** Mesh topology enables secure and robust communication in dynamic and hostile environments.