| **Tính năng cần** | **ESP-NOW** | **ESP-MESH** |
| --- | --- | --- |
| Real-time audio full duplex | ✅ Rất tốt | ⚠️ Có thể nhưng latency cao hơn |
| Auto peer discovery | ⚠️ Cần custom code | ✅ Có sẵn |
| Auto mesh recovery khi mất node | ❌ Không có | ✅ Có sẵn |
| Gửi broadcast tới toàn bộ thiết bị | ✅ Có thể | ✅ Có thể |
| Giao tiếp nhiều hop (multi-hop) | ⚠️ Phải tự hiện thực | ✅ Có sẵn |
| OTA firmware update | ❌ Phải tự làm | ✅ Dễ |
| Tiêu hao năng lượng thấp | ✅ Rất tốt | ❌ Cao |
| Hỗ trợ giao tiếp đồng thời nhiều node | ⚠️ Hạn chế (20 peer) | ✅ Mở rộng tốt |

Vì esp now sử dụng năng lượng thấp, nghe đồn là vậy nên ta sẽ customize lại phần mềm.

There are 3 functions which we need to implement this.

* Auto peer discovery.
* Auto mesh recovery khi mất node.
* Giao tiếp nhiều hop (multi-hop).

Không làm code them khó đọc.

Viết dễ hiểu

Giải thích rõ rang.

Không

# **Auto peer discovery**

Tự động phát hiện các node khác trong mạng để xây dựng bảng peer ban đầu.

| **STT** | **Vấn đề** | **Mô tả ngắn** | **Giải pháp** |
| --- | --- | --- | --- |
| 1 | Khi nào nên thực hiện discovery? | Mở rộng peer list nhưng tránh broadcast quá thường xuyên gây nhiễu. | Chỉ discovery trong 30s đầu sau khi khởi động. |
| 2 | Xác thực node mới là gì? | Làm sao để phân biệt giữa node “hợp lệ” và không hợp lệ. | Xác thực sơ bộ bằng ID hoặc magic number trong gói broadcast. |
| 3 | Dùng broadcast hay ping tuần tự? | Broadcast nhanh hơn nhưng kém kiểm soát. | Dùng broadcast gói HELLO chứa ID. Các node khác phản hồi lại. |
| 4 | Giới hạn số lượng node lưu được? | Giới hạn bởi bộ nhớ, tránh tràn danh sách. | Dừng discovery khi đủ số lượng (ví dụ: 20). |
| 5 | Khi nào loại bỏ node cũ? | Khi làm mới danh sách. | Khi discovery mới diễn ra, reset danh sách. |
| 6 | Làm sao biết một peer còn online? | Cần heartbeat hoặc kiểm tra qua phản hồi (ACK). | Mỗi peer được gửi một ping nhẹ sau mỗi X phút hoặc chờ đến lần truyền tiếp theo kiểm tra ACK. |

# **Auto Mesh Recovery khi mất node**

Nếu node mất kết nối với peer chính, hệ thống phải tự động tìm tuyến thay thế.

| **STT** | **Vấn đề** | **Mô tả** | **Giải pháp** |
| --- | --- | --- | --- |
| 1 | Mất node là gì? | Không gửi được dữ liệu (timeout hoặc không ACK). | Thiết lập timeout nếu không có ACK sau X lần gửi. |
| 2 | Có cần heartbeat không? | Có giúp phát hiện node chết. | Có thể dùng gói ping định kỳ để xác nhận node còn sống (heartbeat). |
| 3 | Có thay đổi định tuyến không? | Có, khi mất kết nối với node cha. | Nếu mất node, chọn node có RSSI cao nhất còn online để chuyển tiếp. |
| 4 | Có lưu routing table không? | Cần, để biết chuyển dữ liệu qua node nào. | Dùng cấu trúc bảng định tuyến tĩnh cập nhật định kỳ. |
| 5 | Khi nào cập nhật routing table? | Khi mất node hoặc khi phát hiện RSSI tốt hơn. | Cập nhật bảng định tuyến khi không gửi được hoặc phát hiện route tốt hơn. |

# **Multi-hop Communication**

| **STT** | **Vấn đề** | **Mô tả** | **Giải pháp** |
| --- | --- | --- | --- |
| 1 | Topology của mạng là gì? | Flat hoặc Opportunistic. | Dùng Opportunistic: node chọn peer có RSSI tốt để chuyển tiếp. |
| 2 | Có cần lưu định tuyến? | Có, để truyền dữ liệu đi nhiều hop. | Có routing table đơn giản với đích và next-hop. |
| 3 | Làm sao tránh vòng lặp hoặc truyền vô tận? | Dùng TTL hoặc hop count. | Gói tin chứa TTL giảm dần mỗi hop. Khi TTL = 0 thì dừng. |
| 4 | Có cần gắn địa chỉ người gửi và người nhận? | Bắt buộc để định tuyến chính xác. | Header gói có src\_id, dest\_id, ttl, packet\_id. |
| 5 | Nếu bị lặp gói thì sao? | Dùng ID gói để loại trừ duplicate. | Lưu danh sách packet\_id gần đây để tránh xử lý lại. |

| **Thông tin** | **Được tạo bởi** | **Được dùng bởi** | **Ghi chú** |
| --- | --- | --- | --- |
| peer\_table[] | Auto Peer Discovery | Cả 3 thuật toán | Lưu các peer đã khám phá |
| routing\_table[] | Multi-hop + Recovery | Multi-hop Communication, Recovery | next\_hop tới từng node |
| last\_seen[] hoặc heartbeat[] | Auto Peer Discovery | Mesh Recovery | Theo dõi peer còn sống |
| packet\_cache[] | Multi-hop Communication | Multi-hop Communication | Tránh lặp gói |
| fsm\_state | FSM của từng thuật toán | Tất cả | Dễ kết hợp hành vi |
| rssi\_table[] | Auto Peer Discovery | Multi-hop chọn tuyến | Ưu tiên đường truyền tốt |

**Controlled Flooding Algorithm with Packet Cache:**

* Kiểu thuật toán broadcast được dùng trong nhiều mesh protocol (BLE Mesh, Zigbee)
* Đặc biệt dùng trong mạng **không định tuyến cố định** (opportunistic forwarding)
* Tối giản nhưng hiệu quả trong mạng nhỏ

# Hiện thực cấu trúc dữ liệu và giải thuật

**🧩 Luồng hoạt động chính**

1. **DISCOVER:**
   * Gửi command, command có thể bất cứ thứ gì đã định nghĩa, CMD\_ADD\_PEER để yêu cầu thu nhận MAC
   * CMD\_UPDATE\_RSSI để thu thập rssi từ các node có trong peer\_table[]
   * Và còn nhiều CMD khác.
   * Thời gian hoạt động dựa trên scheduler.
2. **IDLE:**
   * Nếu có âm thanh từ mic → STATE\_SENDER
3. **SENDER:**
   * Gửi gói AUDIO\_PACKET đến các peer có RSSI tốt
   * Quay về STATE\_IDLE
4. **RECEIVER:**
   * Nếu nhận audio từ peer, thì lưu vào cache và phát audio hoặc TTL < 0 thì chuyển qua FORWARDER.
   * Nếu nhận CMD từ peer thì tùy vào cmd và sẽ theo luồng xử lí,.
5. **FORWARDER:**
   * TTL--; nếu TTL > 0, forward đến các peer khác, trừ src và last\_hop
6. **AUDIO\_PLAY:**
   * Phát âm thanh

| **STT** | **Module** | **Mục đích** | **Có thể test riêng?** |
| --- | --- | --- | --- |
| 1️⃣ | intercom\_fsm.c | FSM quản lý trạng thái hoạt động chính | ✅ Trung bình |
| 2️⃣ | peer\_manager.c | Quản lý danh sách các peer (MAC + RSSI) | ✅ Dễ |
| 3️⃣ | packet\_cache.c | Lọc gói tin đã nhận theo (src\_mac, seq\_num) | ✅ Dễ |
| 4️⃣ | audio\_processor.c | Gửi/nhận/phát audio | ✅ (test logic buffer, không cần mic) |
| 5️⃣ | espnow\_comm.c | Gửi/nhận gói ESP-NOW + phân tích gói tin | ❌ Khó test độc lập (phụ thuộc phần cứng) |
| 6️⃣ | command\_handler.c | Xử lý tất cả các CMD\_... đến | ✅ Dễ |
| 7️⃣ | scheduler.c | Định kỳ gọi FSM, xử lý timer logic | ✅ Dễ |
| 8️⃣ | config.h | Cấu hình các thông số như TTL, thời gian discover, kích thước bảng, độ dài audio... | ✅ |