

Advanced MeshCore Topics

- [MeshCore Path Discovery Deep Dive](#)
- [MeshCore Network Troubleshooting Reference](#)

MeshCore Path Discovery

Deep Dive

A detailed look at MeshCore's flood-first / direct-route-after path learning, derived from `docs/packet_format.md` and the firmware source. MeshCore does not use a dedicated route-discovery protocol (there is no AODV, Route Request, or Route Error); paths are learned as a byproduct of the initial flood.

The Core Mechanism

MeshCore's routing uses two phases that together reduce the channel waste of pure flooding. Note that only the initial flood has path redundancy: once a direct path is learned, subsequent messages follow that single path with no alternate-route redundancy.

Phase 1: Flood with Path Recording

As a `ROUTE_TYPE_FLOOD` packet (route type 0x01) traverses repeaters, each forwarding repeater appends its path-hash to the packet's `path` field, so the destination receives the full ordered path carried inside the packet itself. The path is tracked in the flooded packet, not recorded separately at the destination.

Phase 2: Path Return via `PAYLOAD_TYPE_PATH`

The destination sends a `PAYLOAD_TYPE_PATH` response back to the original sender. This packet contains the ordered list of repeaters from the flood path. The path packet itself uses direct routing if a reverse path is known, otherwise floods back.

Phase 3: Stored Direct Routes

The sender stores the learned path and uses it for all subsequent messages to that destination using `ROUTE_TYPE_DIRECT`. The path remains valid until a direct-routed message fails (no

acknowledgement within timeout), at which point the sender falls back to flooding and re-learns the path. Because a direct-routed message follows a single learned path with no redundancy, if one repeater on that path fails, messages are silently lost until the timeout triggers a re-flood — only the initial flood has path redundancy.

Path Hash Modes

MeshCore supports configurable path hash modes via the `set path.hash.mode {0|1|2}` CLI command, which selects the advert path-hash *size* (0 = 1 byte, 1 = 2 byte, 2 = 3 byte). The **default is 0** (1-byte). This setting affects only the size of the path-hash a repeater uses in its *own* advert broadcasts; it does **not** change how path uniqueness is determined, how the routing table is managed, or what packet ID/hash size this repeater forwards. The feature requires firmware $\geq$ 1.14.

Flood Limits

To prevent unbounded flooding, MeshCore allows configuration of `flood.max` (the maximum hop count a repeater will forward a flood packet to, related to the 64-hop firmware ceiling) and `flood.advert.interval` (the flood-advert cadence in hours). These limit worst-case channel usage.

Regional Scoping

Transport route types (`ROUTE_TYPE_TRANSPORT_FLOOD` and `ROUTE_TYPE_TRANSPORT_DIRECT`) include a 2-byte transport code that allows multi-region networks to scope traffic appropriately. This transport code is calculated from the region scope — a user-defined named scope (e.g. `#Europe`, `#USA`) hashed into the 2-byte value — not an ISO country code.

Sources: route types, the path field, transport codes, and MAX_PATH_SIZE (64) are per `docs/packet_format.md`. CLI parameters (`path.hash.mode`, `flood.*`, region scopes) are per the MeshCore CLI docs. Verified 2026-05-03.

MeshCore Network

Troubleshooting Reference

A systematic approach to troubleshooting MeshCore network issues saves time and frustration. This reference covers the most common problems and their diagnostic approaches.

Diagnostic Framework

When a problem is reported, ask these questions in order:

1. Is the problem isolated to one node pair, or affecting all nodes?
2. Is the problem one-directional or bidirectional?
3. Did it work before? What changed?
4. Is the issue at the RF layer (no signal) or the protocol layer (signal present, no delivery)?

Problem: Node Not Seen by Others

Symptoms: A node is powered on but doesn't appear in other nodes' neighbor lists or can't communicate with anyone.

Diagnostics:

- Check: Is the node's LoRa radio reporting healthy state? (Serial CLI: `stats-core` for battery/uptime/queue length and `stats-radio` for noise floor, last RSSI/SNR, airtime and receive errors)
- Check: Is the frequency/preset matching other nodes? (Verify with `get radio`, which shows freq/bw/sf/cr, or `get freq`)
- Check: Is the antenna connected? (Operate with an antenna or dummy load attached; transmitting into an open or badly mismatched port can stress the power amplifier on some boards)
- Check: Is the channel key matching? (Different key = nodes don't recognize each other's packets)
- Check: Is there a hardware failure? (Try a different known-good node at the same location)

Problem: Messages Not Delivered

Symptoms: Nodes can see each other but messages don't arrive, or arrive with high latency.

- **Low SNR (link near the demodulation floor):** LoRa's minimum decodable SNR depends on the configured spreading factor — roughly -7.5 dB at SF7 down to about -20 dB at SF12 (per Semtech SX126x datasheets). If the measured SNR is near or below the floor for your SF, the link is marginal: raise the spreading factor, improve antennas, or add a repeater. A single flat threshold (such as "-15 dB") is misleading because it is undecodable at low SF yet still usable at SF12.
- **High channel utilization:** Too many nodes flooding the network. Check for misconfigured flooding, reduce hop limit.
- **Path discovery failure:** path discovery flood is flooding but path response is not returning. One-way link - the return path may be blocked by terrain or a directional antenna pointing wrong way.
- **Room server not receiving messages:** Check that room server's attached LoRa node is on the correct channel and hearing the network.

Problem: Room Server Clients Not Syncing

Symptoms: App connects to room server but doesn't show recent messages or shows empty history.

- Check that the room server is actually hearing the mesh: connect to its serial console and run `stats-radio` (noise floor, last RSSI/SNR, receive errors) and `stats-packets` (received/sent packet counters).
- Check the access control list with `get acl` to confirm the connecting client is permitted.
- Check key: Room key (and admin password) must match between app configuration and server configuration. The default admin password is `password` and should be changed.
- Note: a MeshCore room server is LoRa firmware running on an embedded radio device — it is not a Linux daemon listening on a TCP port. There is no firewall/IP port to open for the room server itself. (For reference, the `meshcore-cli` host tool's TCP companion connection defaults to port 5000, not a room-server port.)

Problem: Repeater Goes Offline Periodically

Symptoms: A backbone repeater disappears from the network at irregular intervals, then reappears.

- **Power issue:** Insufficient solar charging, battery capacity degradation. Check voltage logs.
- **Firmware watchdog:** nRF52840 watchdog timer should prevent lockups but may be triggering reboots on firmware bugs. Check for panic/reset logs in serial output.
- **Thermal issue:** Summer heat causing processor throttling or thermal shutdown. Check enclosure temperature.
- **Stability over long uptime:** If a periodic reboot reliably resolves the symptom, update to the latest firmware first. A scheduled reboot can serve as a generic mitigation while you isolate the cause; if you can reproduce a specific fault, file it against the firmware repo with the version and serial logs.

Useful CLI Diagnostic Sequence

```
# Connect to node serial console (device path varies:
# /dev/ttyUSB0 or /dev/ttyACM0 on Linux, COMx on Windows; 115200 baud is correct)
screen /dev/ttyUSB0 115200

# Health and radio state:
stats-core      # Battery, uptime, queue length, debug flags
stats-radio     # Noise floor, last RSSI/SNR, airtime, receive errors
get radio       # Current freq/bw/sf/cr

# Heard nodes and packet counters:
neighbors       # List recently-heard neighbors (encoded as pubkey-prefix:timestamp:snr*4)
stats-packets   # Packet counters (Received, Sent)

# Firmware version:
ver             # Firmware version string

# Room server access control:
get acl         # View the room server's access control list
```