How MeshCore Works

The protocol, routing, encryption, and firmware explained.

- MeshCore Protocol Overview
- MeshCore Firmware Types

MeshCore Protocol Overview

MeshCore is a LoRa mesh networking platform created by Scott (Ripple Radios / ripplebiz), its lead firmware engineer; Liam Cottle builds the app and web tooling. This overview draws on the official MeshCore documentation, FAQ, and source code.

What Makes MeshCore Different

Many LoRa mesh platforms (for example Meshtastic) rely on **managed/broadcast flooding**, where nodes re-broadcast messages subject to hop limits and duplicate suppression rather than naive "every node repeats everything" flooding. MeshCore instead uses a **flood-first, direct-route-after** approach:

1. The first message to any destination is flood-routed (all repeaters in range re-broadcast)
2. The destination returns the path it received the flood through (`PAYLOAD_TYPE_PATH` packet)
3. All subsequent messages to that destination use **direct routing**, embedded with the specific repeater path - only those repeaters forward it

For repeated unicast traffic, this reduces channel utilization once paths are established; group and broadcast traffic still floods, and churn (such as moving nodes or stale paths) triggers re-floods.

Encryption

- **Symmetric encryption: AES-128** (ECB mode)
- **Message authentication: HMAC-SHA256** (2-byte truncated, encrypt-then-MAC)
- **Key exchange: ECDH via X25519** (Ed25519 identity keys transposed)
- **Identity signing: Ed25519** - advertisements signed to prevent spoofing

Firmware Types

- **Companion** - your personal node; connects to the MeshCore app via BLE or USB serial
- **Repeater** - infrastructure node; forwards messages, no user interface
- **Room Server** - stores and delivers missed messages; hosts community "rooms"
- **Sensor** - telemetry node for sensor data

Frequency

- **USA/Canada:** 910.525 MHz, SF7, BW 62.5 kHz, CR5 (a community-chosen "narrow" preset within the unlicensed 902-928 MHz band, as of October 2025 - not an FCC-assigned channel; other settings within the band are also permissible under FCC Part 15.247)
- **EU/UK:** 868 MHz band
- **Australia/NZ:** 915 MHz band

Key Capabilities

- Room Servers provide message store-and-forward (last messages in RAM (capacity firmware-dependent) per client)
- Path hash modes allow tuning for mobile vs. fixed repeater networks
- Regional scoping via ISO country codes prevents cross-region interference
- 50+ supported devices across the RAK WisBlock, Heltec, LilyGo, and Seeed ecosystems

App and Tools

- Android/iOS app by Liam Cottle: Google Play (`com.liamcottle.meshcore.android`) / App Store (`id6742354151`)
- Web app: **app.meshcore.nz**
- Flasher: **flasher.meshcore.io**
- Config tool: **config.meshcore.io**
- Python CLI: github.com/meshcore-dev/meshcore-cli (the `fdlamotte/meshcore-cli` repo now redirects here)

Sources: the official MeshCore repository (meshcore-dev/MeshCore), its FAQ and source code, docs.meshcore.io, and meshcore.co.uk. Region presets and version-dependent details change over time; check the upstream docs for the current values.

MeshCore Firmware Types

MeshCore has three main deployable firmware roles — Companion, Repeater, and Room Server. (A compile-time 'Simple Sensor' build also exists as an experimental/example application, and 'sensor' appears as a contact type, but it is not one of the three primary deployable roles.) All three are available through the official flasher at **flasher.meshcore.io**. The information on this page is verified from the official MeshCore FAQ and GitHub repository.

The Three Firmware Roles

1. Companion Firmware

The firmware for your personal handheld or desk node - the device you use to send and receive messages. Companion firmware connects to the MeshCore app on your phone (Android, iOS, or web).

Connection modes (the same Companion firmware family, built for how the app connects):

- **BLE Companion** - connects to the app via Bluetooth Low Energy. Best for mobile use.
- **USB Serial Companion** - connects via USB cable. Good for desk use or CLI access.
- **WiFi Companion** - a WiFi connection mode (not a separate firmware type) for fixed home setups on boards that support it.

Companion firmware is installed on: RAK4631, Heltec V3, T-Deck, T-Deck Plus, T-Echo, Station G2, and other supported devices.

2. Repeater Firmware

The firmware for infrastructure nodes that extend network coverage. A repeater runs unattended - it receives messages and re-transmits them to extend the mesh, but has no direct user interface.

- Configured via CLI (serial console or BLE) or the config tool at **config.meshcore.io**
- Supports power-saving mode (`powersaving on`) for solar/battery deployments. When enabled, the device sleeps between radio transmissions.
- A Room Server can also repeat via the `set repeat on` command, so one device can do both — but the FAQ does not recommend it: a room server with repeat enabled lacks the full repeater and remote-administration features. For the best experience, run a repeater

and a room server on separate devices.

3. Room Server Firmware

The firmware that turns a device into a message store-and-forward server. A room server:

- Stores messages per client (capacity depends on firmware version)
- Delivers missed messages when a client reconnects
- Hosts a "room" that community members join
- Runs as LoRa firmware on a supported embedded radio device. (A Raspberry Pi can only act as a room server if it has an attached LoRa radio/companion node — a bare Pi has no radio and cannot join the mesh on its own.)

A Note on "Sensor"

"Sensor" is not a fourth co-equal deployable firmware role. Environmental-sensor support in MeshCore is a compile-time build feature (enabled with build flags), and a "Simple Sensor" example build exists in the source tree as an experimental/example application. Separately, "sensor" appears as a contact-type enum value (sensor=4) used to classify a node in contact lists. Neither makes Sensor a first-class deployable role alongside Companion, Repeater, and Room Server. See the IoT & Sensors book for how MeshCore's compile-time sensor support actually works.

Developer / Example Firmware

The MeshCore source repository also contains example firmware used for development and specialized applications:

- `simple_sensor` - an example "Simple Sensor" telemetry build (compile-time, not a primary deployable role)
- `simple_secure_chat` - standalone terminal chat (not a general-use firmware)
- `kiss_modem` - KISS protocol modem interface for integration with other software

These example firmwares are for developers and are not distributed as primary deployable roles through the standard flasher.

Source: Official MeshCore FAQ (github.com/meshcore-dev/MeshCore/blob/main/docs/faq.md), build.sh, and flasher.meshcore.io. Verified 2026-05-03.