MeshCore: A Deterministic Flood-Routing Protocol for LoRa Mesh Networks

PHY-Layer Foundations, Routing Architecture, and Operational Guidance

April 2026

Version 2.0

DISTRIBUTION: Public — Unrestricted

Abstract

MeshCore is a lightweight, deterministic flood-routing mesh protocol engineered for LoRa chirp-spread-spectrum (CSS) radio networks operating on ISM-band frequencies. Unlike conventional mesh protocols that rely on routing tables, directed acyclic graphs (DAGs), or spanning-tree constructions, MeshCore employs a stateless per-hop forwarding model in which each relay node makes an independent retransmission decision based solely on the information carried within the received packet header. The protocol requires no persistent routing state, no network-wide topology synchronization, and no centralized coordination infrastructure. A defining characteristic of MeshCore's physical-layer integration is hop-level preamble regeneration: each relay node strips the received LoRa preamble and synthesizes a fresh preamble prior to retransmission, thereby resetting timing-drift accumulation and enabling independent receiver acquisition at every hop. This paper provides a comprehensive treatment of the MeshCore protocol stack, beginning with LoRa PHY-layer foundations—including chirp-spread-spectrum modulation theory, preamble acquisition mechanics, and error-coding schemes—and progressing through the packet format, routing engine architecture, collision-domain analysis, timing architecture, security model, hardware platform integration, and operational deployment guidance. The protocol is designed for off-grid, infrastructure-sparse, and emergency communication scenarios where RAM budgets may be as low as 32 KB and network topologies are inherently dynamic.

Table of Contents

List of Figures

List of Tables

Chapter 1 — Introduction

1.1 Motivation

The proliferation of low-power, long-range (LoRa) radio modules has created a substantial opportunity for decentralized mesh communication networks that operate without power, internet, cellular, Wi-Fi, or satellite infrastructure. LoRa's chirp-spread-spectrum (CSS) modulation enables link budgets exceeding 150 dB, yielding single-hop ranges of several kilometers in open terrain. However, realizing robust multi-hop mesh networking over LoRa radios presents unique challenges that existing protocol stacks have not fully addressed.

Meshtastic, the most widely adopted LoRa mesh firmware, operates on a single shared channel with a simple flooding mechanism. While effective for small networks (fewer than approximately 20 nodes), Meshtastic's approach suffers from several limitations at scale: single-channel contention leads to escalating collision rates as node count increases; the absence of true multi-hop route optimization causes redundant retransmissions; and the protocol's routing-table overhead, while modest, exceeds the RAM budget of the most constrained embedded targets. Reticulum, an alternative framework, offers more sophisticated networking capabilities but introduces complexity that may exceed the requirements of embedded, battery-powered deployments.

MeshCore was designed to address these limitations with the following core objectives:

- **Zero-infrastructure operation:** No gateways, servers, or internet connectivity required for basic mesh communication.
- **Deterministic behavior:** The relay decision at each hop is a pure function of the received packet header and local node configuration, producing predictable and analyzable network behavior.
- **Minimal RAM footprint:** The protocol's stateless forwarding model eliminates the need for routing tables, neighbor discovery databases, or topology maps, enabling operation on microcontrollers with as little as 32 KB of SRAM.
- **Hop-level independence:** Each relay hop performs independent preamble regeneration and receiver acquisition, isolating link-quality assessment to individual hops and preventing timing-drift accumulation across multi-hop paths.

1.2 Scope of This Paper

This white paper provides a comprehensive technical treatment of the MeshCore protocol from the physical layer through the application layer. The coverage includes:

- **PHY-layer foundations:** LoRa CSS modulation theory, preamble structure and acquisition, error coding, and PHY parameter selection rationale.
- **Packet format:** Byte-level frame structure, header encoding, path-section semantics, and payload-type taxonomy.
- **Routing engine:** Flood routing, direct routing, zero-hop mode, transport mode, and the per-hop relay decision algorithm.
- **Collision analysis:** Collision-domain modeling, timing architecture, random backoff strategy, and airtime budgeting.
- **Security:** Identity model, encryption, authentication, and threat analysis.
- **Hardware and operations:** Platform support, sensor integration, OTA updates, deployment planning, and regulatory compliance.

The following topics are explicitly excluded from scope: application-level APIs for companion applications, the MeshCore companion-app serial protocol (documented separately), and user-interface design considerations.

1.3 Terminology and Conventions

Term	Definition
Node	Any device running MeshCore firmware and participating in the mesh network.
Repeater	A node configured to relay packets for other nodes. May be *managed* (remotely configurable) or *unmanaged* (autonomous).
Room	A virtual group channel hosted by a Room Server node, enabling group messaging with access control.
Client	An end-user node that originates and terminates messages but does not relay traffic.
Companion Radio	A node operating in companion mode, acting as a radio modem for a connected phone or laptop via BLE or USB serial.
Flood Routing	The default routing mode in which every relay node rebroadcasts a received packet exactly once.

Term	Definition
Direct Routing	A unicast routing mode that uses the recorded path from a prior flood delivery to forward packets through specific relay nodes without flooding.
TTL	Time-to-live: the maximum number of relay hops permitted for a packet, decremented at each hop.
Path Encoding	A compact representation of the relay path embedded in the packet header for direct routing.
ADVERT	An advertisement packet broadcast by a node to announce its presence and capabilities.
CAD	Channel Activity Detection: a low-power LoRa receiver mode that detects preamble energy in approximately one symbol period.

Throughout this document, hexadecimal values are denoted with the prefix **0x** (e.g., 0x3C), binary values with **0b** (e.g., 0b00110000), and all signal-level measurements are expressed in dBm (power) or dB (ratio) unless otherwise noted.

Chapter 2 — System Architecture Overview

2.1 Layered Architecture

The MeshCore protocol is organized into four functional layers, each with clearly delineated responsibilities. The layered architecture facilitates modular implementation and enables PHY-layer substitution (e.g., replacing LoRa with a different packet radio) without affecting upper-layer logic.

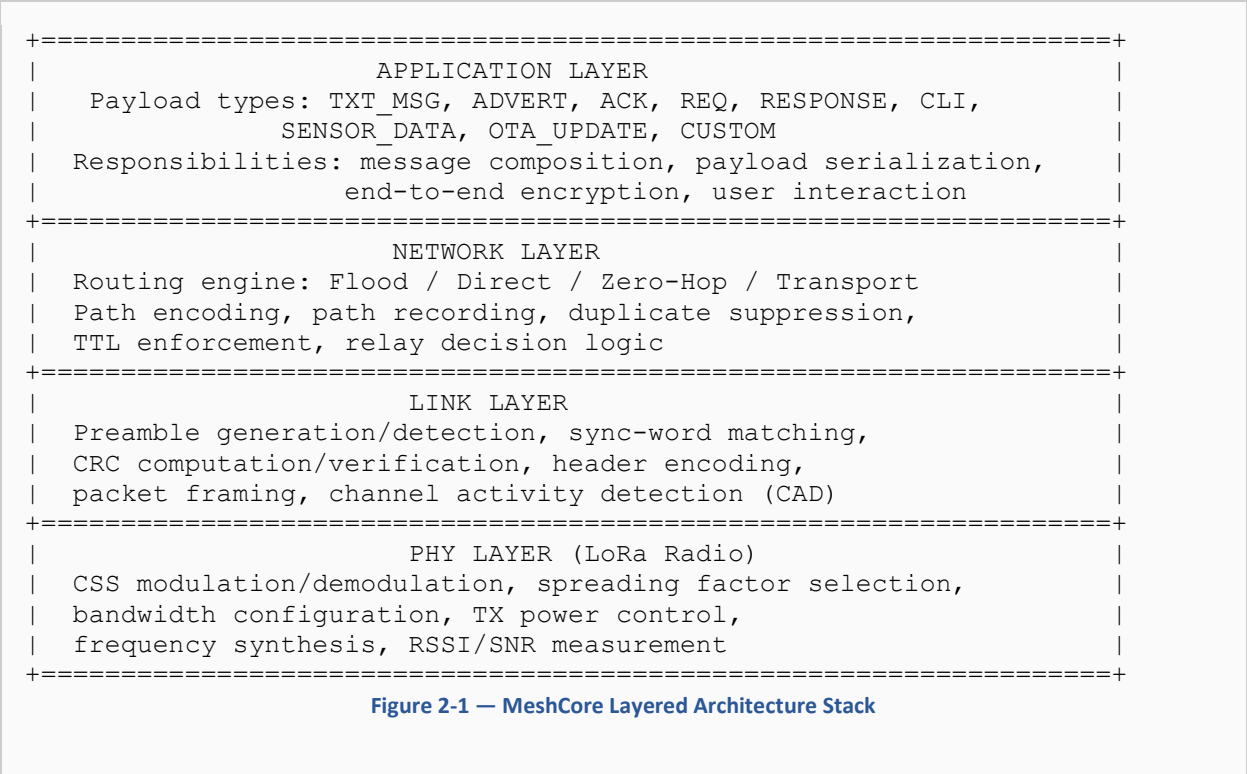


Figure 2-1 — MeshCore Layered Architecture Stack

2.2 Node Roles

MeshCore defines five distinct node roles, each with specific capabilities and behavioral constraints. Notably, Companion nodes do not relay packets, preventing suboptimal routing paths through devices intended only as user endpoints.

Table 2-1 — MeshCore Node Roles and Capabilities

Role	Relays Packets	Originates Messages	Hosts Rooms	Typical Hardware
Client Node	No	Yes	No	Heltec V3, T-Beam, handheld devices
Repeater (Managed)	Yes	No (ADVERT only)	No	Heltec V3, RAK4631 with solar
Repeater (Unmanaged)	Yes	No (ADVERT only)	No	Any supported device, autonomous
Room Server	Yes	Yes (room management)	Yes	ESP32 or nRF52840 with stable power
Companion Radio	No	Yes (via host app)	No	T-Beam + phone via BLE/USB

2.3 Channel Architecture

MeshCore operates on a single shared radio channel in half-duplex mode. All nodes within radio range share the same frequency, spreading factor, and bandwidth configuration. This single-channel architecture simplifies implementation and eliminates the need for frequency-hopping coordination, but it requires careful attention to collision management (Chapter 7).

Duty-cycle constraints are governed by regional regulatory frameworks:

- **European Union (863–870 MHz):** ETSI EN 300 220 imposes a 1% duty-cycle limit per sub-band, equating to a maximum of 36 seconds of transmission time per hour per sub-band.
- **United States (902–928 MHz):** FCC Part 15.247 permits frequency-hopping spread spectrum (FHSS) with dwell-time limits of 400 ms per hop for systems using at least 50 hopping channels, or digital modulation with a maximum conducted output power of +30 dBm (1 W).
- **Australia (915–928 MHz):** ACMA regulations permit 1 W EIRP with no explicit duty-cycle limit but with spectral density constraints.

Chapter 3 — LoRa PHY-Layer Foundations

This chapter provides a thorough treatment of the LoRa physical layer as it pertains to MeshCore operation. Understanding these PHY-layer mechanics is essential for analyzing mesh performance, diagnosing link failures, and optimizing deployment configurations.

3.1 Frequency

In the United States the FCC regulates the ISM (Industrial, Scientific, and Medical) bands under 47CFR Part 18 and devices operating in these bands are unlicensed. This is the governing section that defines ISM equipment, allowed frequencies, tolerances, and operating conditions. Any of the bands (6.78MHz, 13.56MHz, 27.12MHz, 40.68MHz, 915Mhz, etc.) may be utilized, but in the United States MeshCore is typically on the 915MHz band (902MHz – 928Hz). The US standard is 910.525MHz, but 927.875 has also been used. MeshCore users outside the United States primarily operate in the 868 MHz EU ISM band, the 915–928 MHz AU/NZ/Canadian band, and in some countries the 433 MHz ISM band.

3.1 Chirp Spread Spectrum Modulation

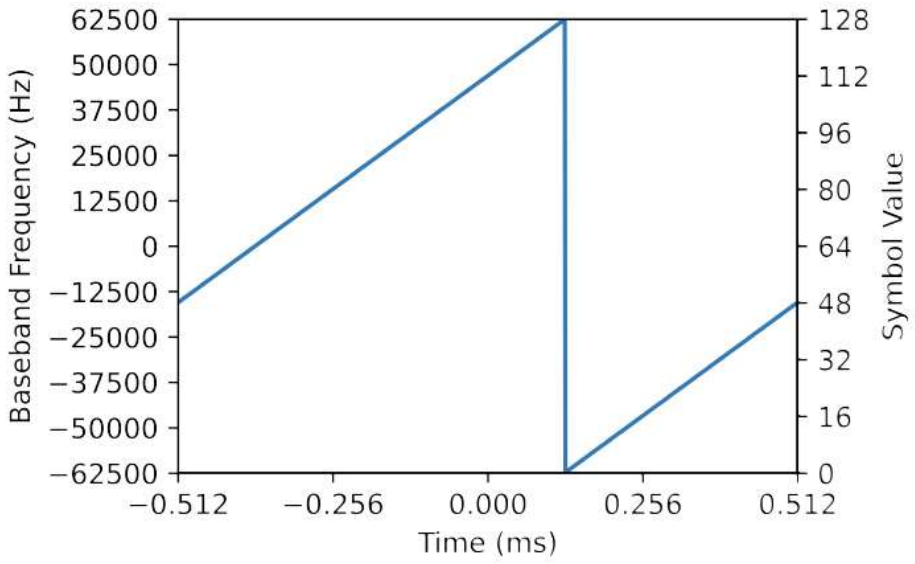
LoRa modulation is based on chirp spread spectrum (CSS), a technique in which data is encoded as frequency-varying chirp signals that sweep across the allocated bandwidth. The fundamental waveform is the *base chirp* (also termed the *up-chirp*), a sinusoidal signal whose instantaneous frequency increases linearly from the lower band edge ($f_0 - BW/2$) to the upper band edge ($f_0 + BW/2$) over one symbol period.

Data encoding is achieved through cyclic time-shifts of the base chirp. Each symbol carries SF bits of information, where SF is the *spreading factor*. The number of possible cyclic shifts (and thus the symbol alphabet size) is $M = 2^{SF}$. A data value *k* (where $0 \le k < 2^{SF}$) shifts the base chirp by k chips, causing the frequency sweep to begin at an offset position and wrap around at the band edge.

The symbol duration is determined by:

$$T_{sym} = 2^{SF} / BW$$

where BW is the chirp bandwidth in Hz. Each symbol contains 2^{SF} chips, and the chip rate equals the bandwidth. The spreading factor thus controls the trade-off between data rate, receiver sensitivity, and time-on-air.



An up-chirp encoding symbol value 48 with SF7 and $BW = 125$ kHz.

The *processing gain* of CSS modulation is approximately $10 \cdot \log_{10}(2^{SF})$ dB, which allows LoRa signals to be demodulated well below the noise floor. At SF12, the processing gain is approximately 36 dB, enabling reception at signal-to-noise ratios as low as −20 dB.

Table 3-1 — LoRa Spreading Factor Parameters (BW = 125 kHz and 250 kHz)

SF	Chips/Symbol	T_{sym} @ 125 kHz	Bit Rate @ 125 kHz (bps)	T_{sym} @ 250 kHz	Bit Rate @ 250 kHz (bps)	Sensitivity (dBm)	Processing Gain (dB)
7	128	1.024 ms	5,469	0.512 ms	10,938	−123	21.1
8	256	2.048 ms	3,125	1.024 ms	6,250	−126	24.1
9	512	4.096 ms	1,758	2.048 ms	3,516	−129	27.1
10	1,024	8.192 ms	977	4.096 ms	1,953	−132	30.1
11	2,048	16.384 ms	537	8.192 ms	1,074	−134.5	33.1
12	4,096	32.768 ms	293	16.384 ms	586	−137	36.1

Bit rates shown assume coding rate CR 4/5. Sensitivity values are typical for Semtech SX1262 transceivers at BW = 125 kHz.

3.2 Preamble Structure and Acquisition

Every LoRa packet transmission begins with a preamble sequence that enables the receiver to detect the presence of a signal, synchronize its timing, and prepare for data demodulation. The LoRa preamble comprises three distinct sections:

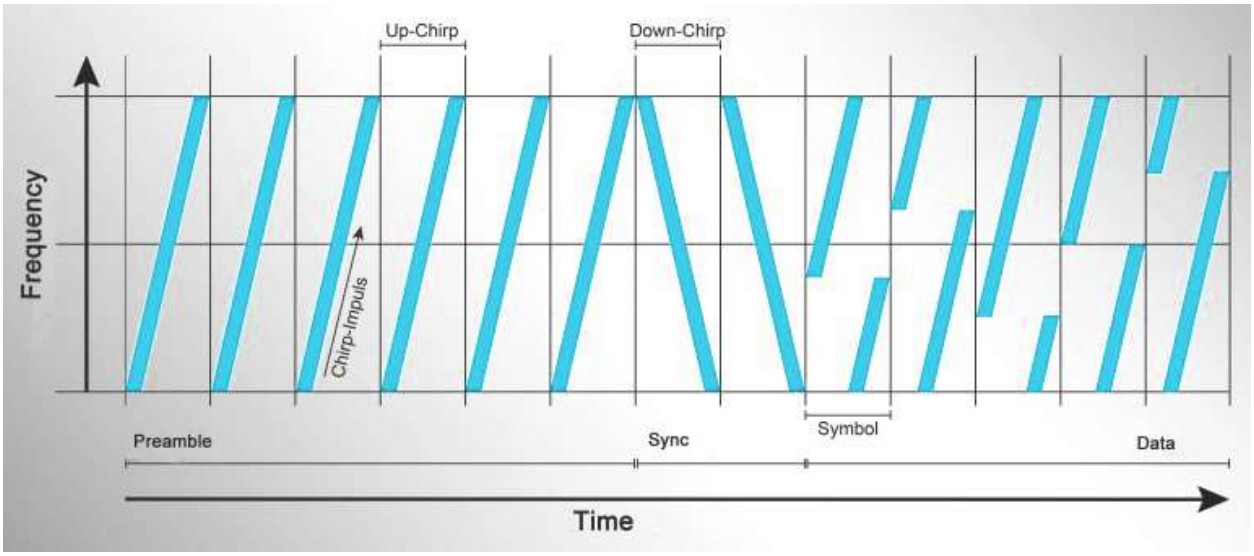
1. **Preamble up-chirps:** A configurable number ($N_{preamble}$, typically 8) of unmodulated base up-chirps. These repetitive chirps provide the correlation target for initial signal detection.

   Sync-word chirps: Two chirps modulated with the sync-word value. The sync word (e.g., 0x34 for LoRaWAN, 0x12 or 0x14 for private networks) enables the receiver to distinguish between networks operating on the same frequency. MeshCore uses the default private LoRa sync word (0x12) rather than the public LoRaWAN sync word (0x34, TTN, Helium, etc.). This ensures that MeshCore traffic is ignored by LoRaWAN infrastructure and vice versa, preventing cross-protocol interference on shared ISM bands. However, private LoRaWAN deployments and non-LoRaWAN point-to-point LoRa networks using the default private sync-word will not be excluded and are accepted.

   The private sync word is the power-on default for Semtech SX127x and SX126x transceivers, requiring no additional configuration.

   It should be noted that Meshtastic also uses a private sync word, but it is not the default. It is a custom sync word, so Meshtastic messages that are sent on the same channel as MeshCore are excluded and not received by the MeshCore radio receiver.

   Start-of-Frame Delimiter (SFD): 2.25 down-chirps (conjugate of the base chirp, sweeping from high to low frequency). The SFD provides a sharp correlation boundary that marks the transition from preamble to data.




```
LoRa Packet Preamble Structure
=============================================================================================
|<--- N_preamble up-chirps --->|<-Sync->|<--SFD-->|<-Header->|<---Payload--->|<CRC> |
|                              |  Word  |         |          |               |      |
+------+------+------+----+----+--+--+---+---+--+-+----------+---------------+------+
| Up 1 | Up 2 | Up 3 |....| Up8|S1|S2|Dn1|Dn2|.25 |  Header  |   Payload     | CRC  |
+------+------+------+----+----+--+--+---+---+--+-+----------+---------------+------+
|                              |        |         |                                  |
|   8 unmodulated up-chirps    | 2 sync | 2.25 dn |         Data Section             |
|   (correlation target)       | chirps | chirps  |                                  |
|                              |(network|(frame   |                                  |
|                              | ID)    | delim)  |                                  |
=============================================================================================
      Direction of time --->
```

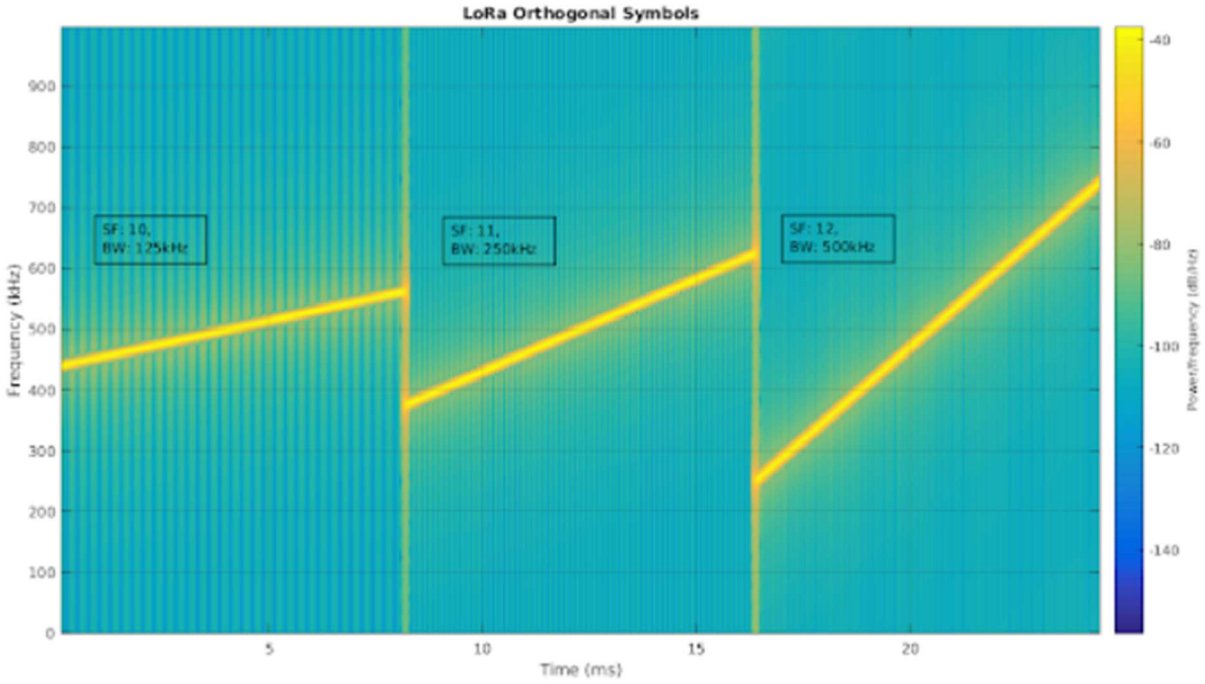
Figure 3-1 — LoRa Preamble Structure

The receiver's preamble acquisition process follows a defined sequence: the receiver correlates the incoming signal against a locally generated base-chirp replica, seeking an energy peak above the detection threshold. Upon detecting sufficient correlation energy across multiple consecutive chirps, the receiver locks its symbol timing, reads the sync-word chirps to verify network membership, confirms the SFD transition, and then proceeds to header and payload demodulation.

Table 3-2 — Preamble Detection Thresholds by Spreading Factor

SF	Min. Detectable SNR (dB)	Detection Latency (1 symbol)	CAD Detection Time	Min. Preamble Chirps for Lock
7	−7.5	1.024 ms	~1.1 ms	4
8	−10.0	2.048 ms	~2.1 ms	4

SF	Min. Detectable SNR (dB)	Detection Latency (1 symbol)	CAD Detection Time	Min. Preamble Chirps for Lock
9	−12.5	4.096 ms	~4.2 ms	5
10	−15.0	8.192 ms	~8.4 ms	5
11	−17.5	16.384 ms	~16.8 ms	6
12	−20.0	32.768 ms	~33.6 ms	6



3.3 Hop-Level Preamble Regeneration in MeshCore

A defining characteristic of MeshCore's physical-layer integration is *hop-level preamble regeneration*. When a relay node receives a packet, it does not forward the received preamble bit-for-bit. Instead, the relay node performs the following sequence:

2. Receives and demodulates the complete packet (preamble through CRC).
3. Strips the received preamble entirely.
4. Processes the packet at the network layer (relay decision, TTL decrement, path update).
5. Synthesizes a *fresh* preamble using the local oscillator and timing reference.
6. Transmits the new preamble followed by the (possibly modified) packet data.

This behavior has three critical consequences for mesh network performance:

- **Timing-drift reset:** Clock drift accumulated during the previous hop's transmission and propagation is not carried forward. Each hop begins with a freshly synthesized preamble locked to the relay node's crystal oscillator, preventing cumulative timing errors across long multi-hop paths.
- **Independent receiver acquisition:** The receiving node at each hop performs a full, independent acquisition sequence. Link quality (RSSI, SNR) is measured per-hop, not end-to-end, enabling accurate per-link quality assessment.
- **Sync-word enforcement:** Each retransmission includes a freshly generated sync word, ensuring that network-membership filtering is enforced at every hop.

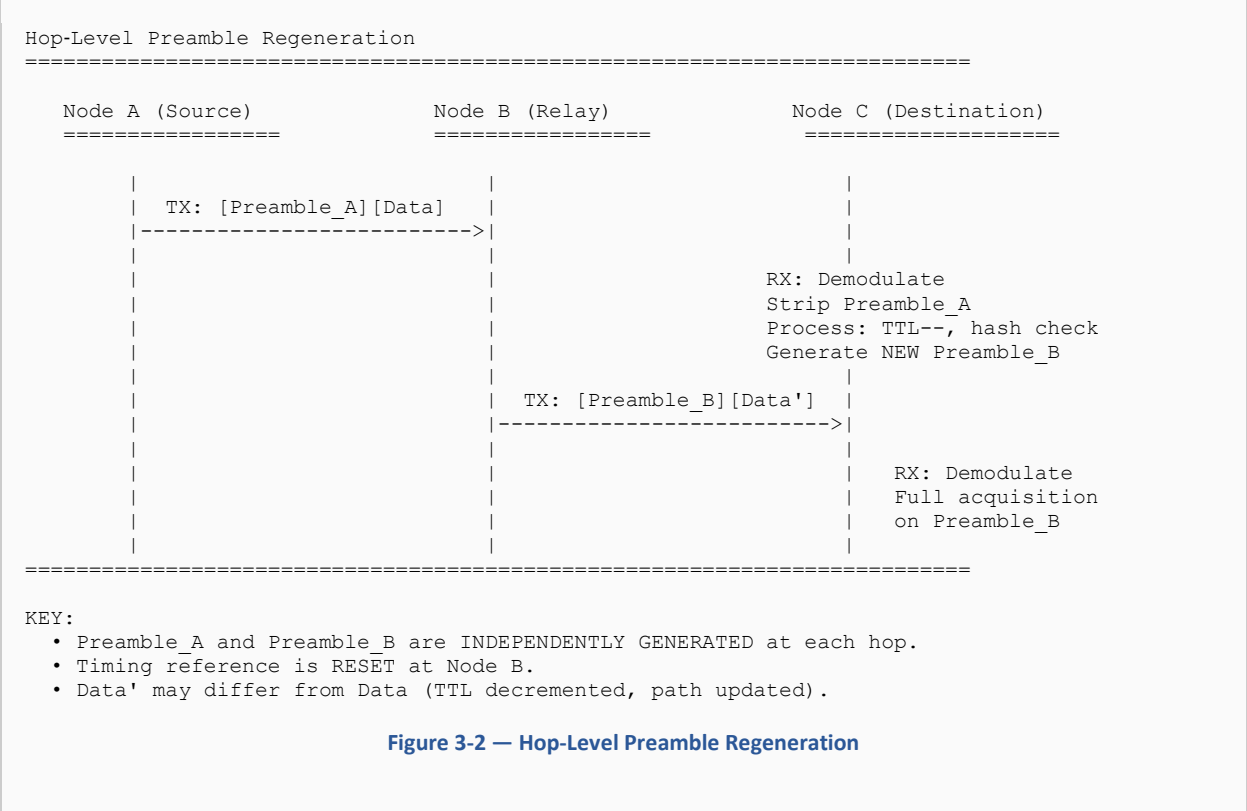


Figure 3-2 — Hop-Level Preamble Regeneration

3.4 Receiver Acquisition Flow

The MeshCore receiver operates as a state machine with the following states, cycling continuously when the node is active:

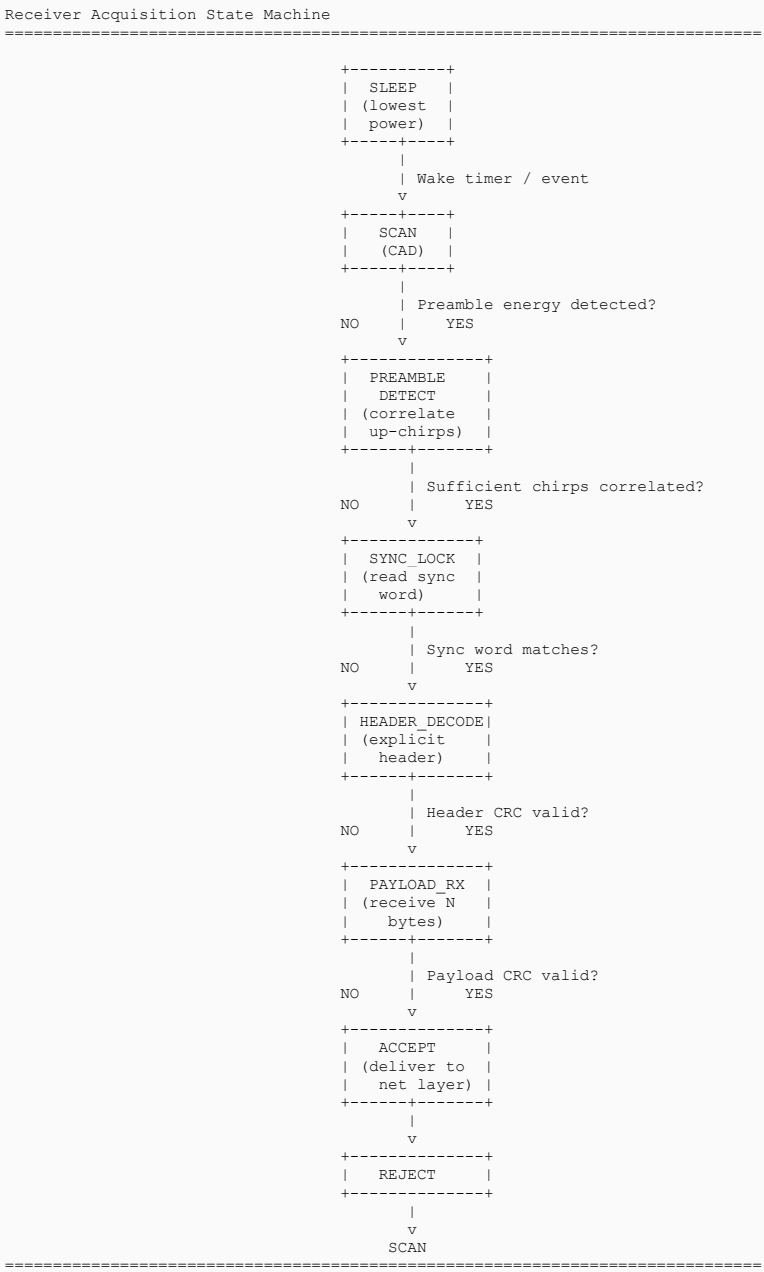


Figure 3-3 — Receiver Acquisition State Machine

Channel Activity Detection (CAD): CAD mode is a low-power listening state in which the receiver samples incoming energy for approximately one symbol period and performs a preamble-correlation check. If preamble energy is detected, the receiver transitions to full reception mode. If not, the receiver may return to sleep. CAD detection latency is approximately one symbol period (e.g., ~1 ms at SF7/125 kHz, ~33 ms at SF12/125 kHz). This mode enables duty-cycled operation for battery-powered nodes.

3.5 Error Coding and CRC

LoRa employs forward error correction (FEC) to improve resilience against bit errors introduced by noise, interference, and multipath fading. The coding rate (CR) parameter specifies the ratio of data bits to total transmitted bits:

Coding Rate	Notation	Data Bits per Block	Parity Bits per Block	Overhead
4/5	CR1	4	1	25%
4/6	CR2	4	2	50%
4/7	CR3	4	3	75%
4/8	CR4	4	4	100%

The LoRa explicit header is always transmitted at CR 4/8 regardless of the payload coding rate, ensuring maximum robustness for the critical header information (payload length, coding rate, CRC presence).

Payload integrity is verified using a CRC-16 (CCITT polynomial) appended to the payload. Packets failing the CRC check are silently discarded. The combination of FEC and CRC provides a two-tier error-handling strategy: FEC corrects minor bit errors in-flight, while CRC detects residual errors that exceed the FEC correction capability.

Higher coding rates increase airtime proportionally but improve reliability on marginal links. MeshCore's dynamic CR feature (under development) will allow the firmware to automatically select a higher coding rate for hops with poor SNR and a lower coding rate for strong links, optimizing the airtime-vs-reliability trade-off on a per-hop basis.

3.6 LoRa PHY Parameter Selection for MeshCore

MeshCore defines several PHY configuration profiles to accommodate different deployment scenarios. The default profile balances range, airtime, and collision probability for general-purpose mesh networking.

Table 3-3 — MeshCore PHY Configuration Profiles

Profile	SF	BW (kHz)	CR	Preamble	Bitrate (bps)	Sensitivity (dBm)	Max Payload (B)	Typical Range
Long Range	12	125	4/8	8	146	−137	184	15–25 km (LOS)
Standard	10	125	4/6	8	651	−132	184	5–12 km
Medium Range	9	250	4/5	8	3,516	−126	184	3–7 km
Fast / Short	7	250	4/5	8	10,938	−120	184	1–3 km
Turbo	7	500	4/5	8	21,875	−117	184	0.5–1.5 km

Trade-off analysis: Increasing the spreading factor by one step approximately doubles the time-on-air while improving sensitivity by approximately 2.5 dB. For multi-hop mesh networks, the relationship between airtime and collision probability is particularly important: longer packets occupy the channel for longer periods, increasing the probability of collision with packets from other nodes in the same collision domain. The Standard profile (SF10, BW 125, CR 4/6) is recommended as the default for most MeshCore deployments, as it provides a practical balance between 5–12 km single-hop range and manageable airtime for multi-hop forwarding.

Chapter 4 — Packet Format and Encoding

4.1 Frame Structure

The MeshCore packet format is designed for minimal overhead while providing sufficient structure for multi-hop routing, payload-type discrimination, and integrity verification. The general frame layout is as follows:

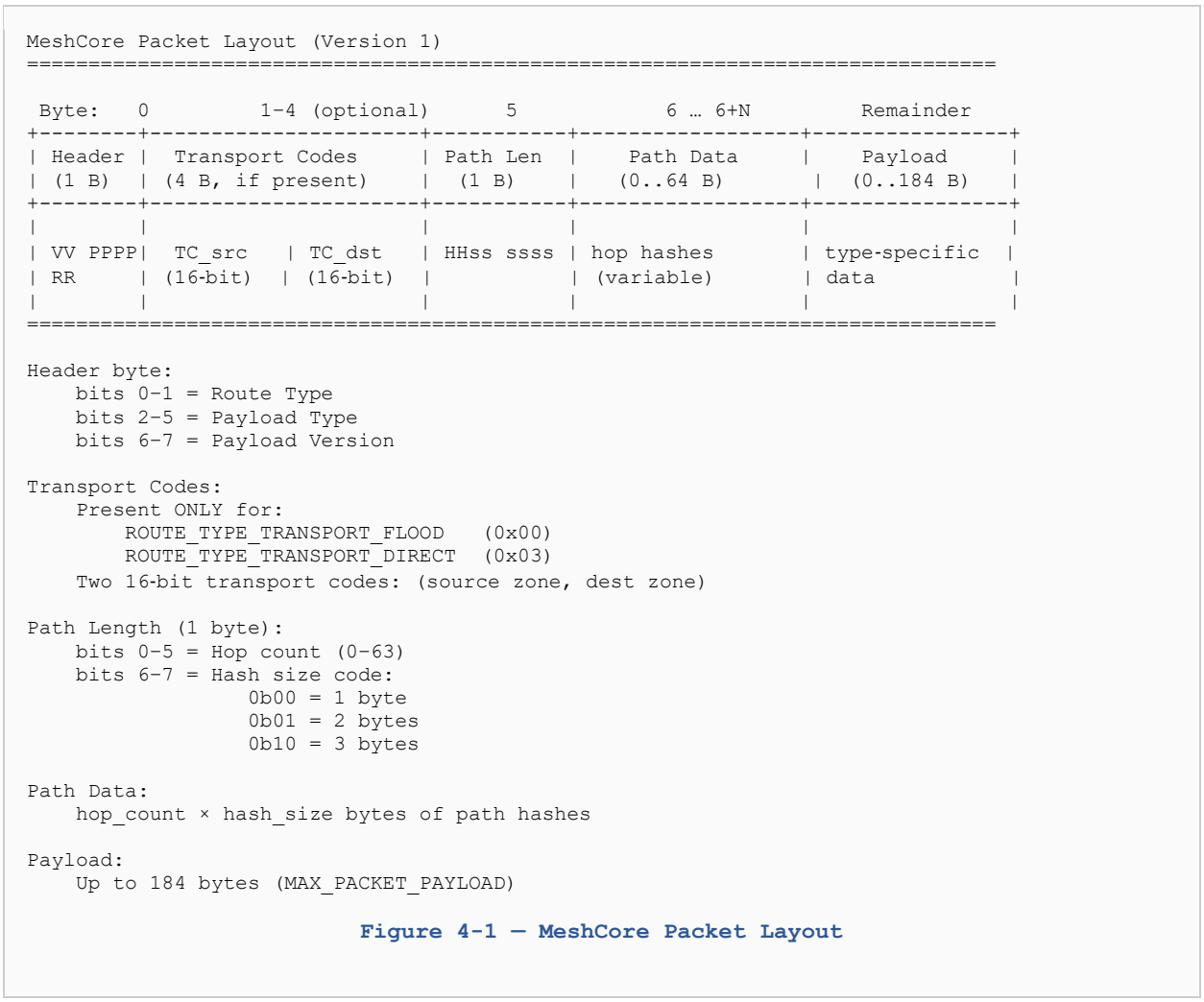


Figure 4-1 — MeshCore Packet Layout

4.2 Header Byte Encoding

The header byte is a single-byte bit field that encodes the routing mode, payload type, and protocol version in a compact format:

Table 4-1 — Header Byte Bit-Field Encoding

Bits	Mask	Field	Description
0–1	0x03	Route Type	Determines how the packet is forwarded: Flood, Direct, Transport, etc.
2–5	0x3C	Payload Type	Identifies the payload format: REQ, RESPONSE, TXT_MSG, ACK, ADVERT, etc.
6–7	0xC0	Payload Version	Versioning field for forward compatibility of payload formats.

Route Types:

Value	Name	Description
0x00	ROUTE_TYPE_TRANSPORT_FLOOD	Flood routing with transport codes for zone-based filtering.
0x01	ROUTE_TYPE_FLOOD	Standard flood routing; every relay rebroadcasts once.
0x02	ROUTE_TYPE_DIRECT	Unicast direct routing using a recorded path from a prior flood delivery.
0x03	ROUTE_TYPE_TRANSPORT_DIRECT	Direct routing with transport codes.

4.3 Path Section

The path section consists of a *path_length* byte followed by a variable-length sequence of path hashes. The path_length byte encodes both the hop count and the hash size in a packed format:

- **Bits 0–5 (hop count):** The number of path hashes present in the path data (range 0–63).
- **Bits 6–7 (hash size code):** Encoded as hash_size − 1. Values: 0b00 = 1-byte hashes, 0b01 = 2-byte hashes, 0b10 = 3-byte hashes, 0b11 = reserved.

The total path data length in bytes is hop_count × hash_size. For example, a path_length value of 0x45 indicates 5 hops using 2-byte hashes (10 bytes of path data).

In **flood routing mode**, the path section records the sequence of relay nodes through which the packet has traveled. Each relay appends its own hash to the path before retransmission. In **direct routing mode**, the path section contains the pre-computed route: relay nodes forward the packet only if their own hash appears at the expected position in the path sequence.

4.4 Payload Types

Table 4-2 — MeshCore Payload Types

Value	Name	Description
0x00	PAYLOAD_TYPE_REQ	Request packet containing destination/source hashes and MAC. Initiates a transport-mode handshake.
0x01	PAYLOAD_TYPE_RESPONSE	Response to a REQ or ANON_REQ. Carries the requested data or acknowledgment.
0x02	PAYLOAD_TYPE_TXT_MSG	Plain-text or encrypted text message payload.

Value	Name	Description
0x03	PAYLOAD_TYPE_ACK	Acknowledgment of a previously received packet, identified by hash.
0x04	PAYLOAD_TYPE_ADVERT	Node advertisement: broadcasts identity, capabilities, and presence.
0x05	PAYLOAD_TYPE_GRP_TXT	Group text message destined for a Room Server channel.
0x06	PAYLOAD_TYPE_CLI	Command-line interface command for remote node management.
0x07	PAYLOAD_TYPE_STATS	Node statistics and telemetry data (RSSI, SNR, battery, packet counts).
0x08	PAYLOAD_TYPE_SENSOR_DATA	Sensor readings (temperature, humidity, soil moisture, etc.).
0x09	PAYLOAD_TYPE_OTA_UPDATE	Over-the-air firmware update chunk with sequence number and CRC.
0x0A	PAYLOAD_TYPE_ANON_REQ	Anonymous request without source identification.
0x0B–0x0F	Reserved / CUSTOM	Reserved for future protocol extensions and user-defined payloads.

4.5 Message Integrity

MeshCore employs a truncated hash for two purposes: *duplicate detection* and *integrity verification*. Each packet's payload is hashed, and a 4-byte truncated hash is derived. This hash serves a dual role:

- **Duplicate suppression:** Each relay node maintains a ring buffer of recently observed packet hashes (the *hash cache*). When a packet arrives, its hash is compared against the cache. If a match is found, the packet is a duplicate and is silently discarded. The cache is bounded in size (typically 64–128 entries) and entries expire based on TTL.
- **Integrity verification:** The hash provides a lightweight check against payload corruption that may not be caught by the LoRa CRC (e.g., if corruption occurs between CRC verification and network-layer processing).

For authenticated and encrypted payloads, a full Message Integrity Code (MIC) derived from the encryption key replaces the simple hash, providing cryptographic assurance of both integrity and authenticity.

Chapter 5 — Routing Engine Architecture

5.1 Design Philosophy

The MeshCore routing engine is founded on a principle of *stateless per-hop forwarding*. Unlike protocols that maintain routing tables (e.g., AODV, OLSR), spanning trees (e.g., IEEE 802.1D), or directed acyclic graphs (e.g., RPL), MeshCore requires no persistent routing state beyond a small, bounded hash cache for duplicate suppression. Each relay node's forwarding decision is a pure function of:

- The received packet's header fields (route type, TTL, payload type).
- The packet's hash (for duplicate detection).
- The local node's role and configuration (repeater vs. client, RSSI/SNR thresholds).
- The received signal quality (RSSI, SNR) of the current hop.

This design philosophy yields several advantages for resource-constrained embedded systems: RAM usage is bounded and predictable (no dynamically sized routing tables); the protocol converges instantly (no convergence delay after topology changes); and node behavior is deterministic and analyzable.

5.2 Flood Routing

Flood routing is MeshCore's default and most robust routing mode. When a packet is transmitted in flood mode, every relay node that receives it will rebroadcast it exactly once (subject to TTL and duplicate-suppression constraints). The flood propagation pattern expands concentrically from the source through successive relay tiers:

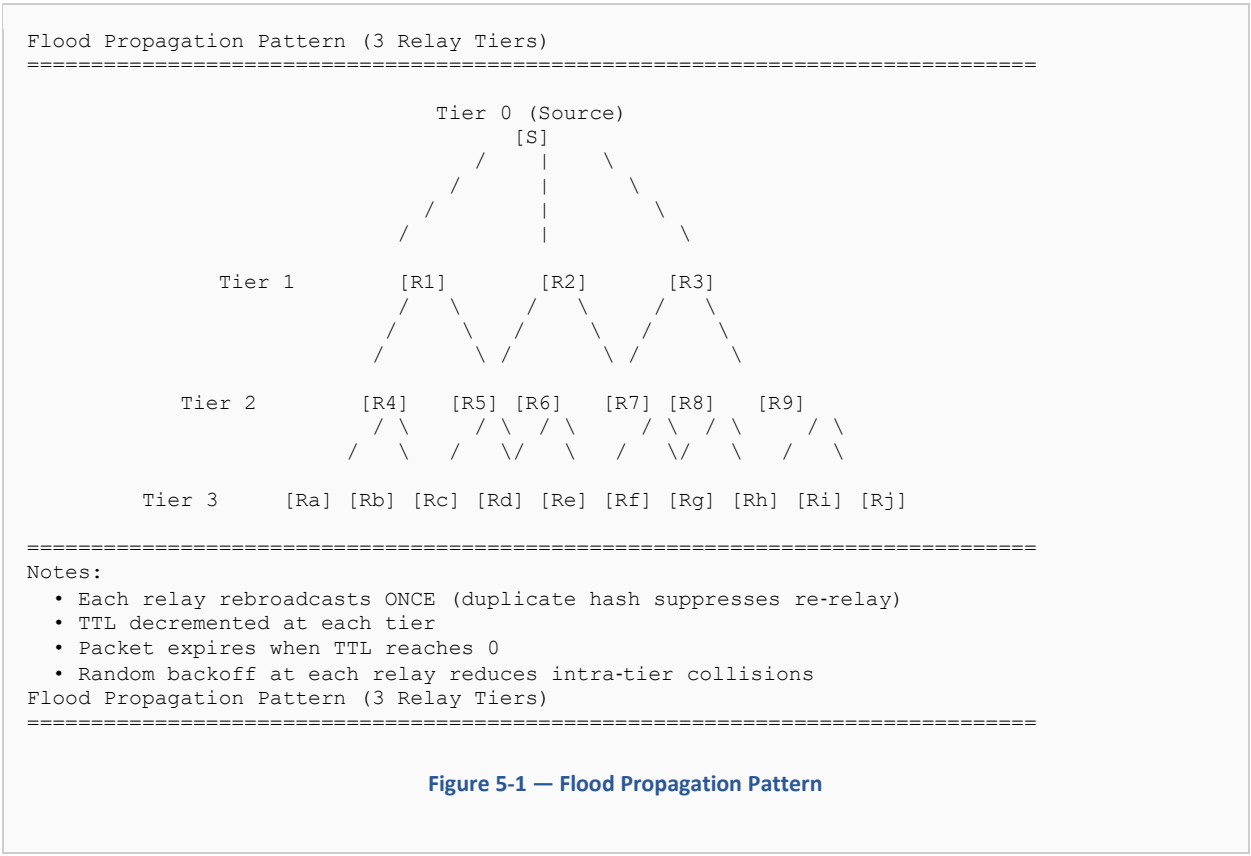


Figure 5-1 — Flood Propagation Pattern

Duplicate suppression is enforced via the hash cache: a ring buffer of recent packet hashes maintained at each relay node. When a packet arrives, the relay computes the packet's hash and checks it against the cache. If a match is found, the packet has already been relayed and is discarded. If no match is found, the hash is added to the cache and the packet is forwarded (subject to TTL and signal-quality checks). The hash cache is typically sized at 64–128 entries, bounded by available RAM.

5.3 Direct Routing

Direct routing is a unicast mode that leverages *a previously recorded path* to forward packets through a specific sequence of relay nodes rather than flooding the entire network. The path is encoded in the packet's path section as a sequence of relay node hashes. Each relay node examines the path data and forwards the packet only if its own hash appears at the expected position in the path sequence.

Direct routing is initiated when the originating node possesses a recorded path for the destination, obtained from a prior flood delivery. If the direct-routed packet fails to reach the destination (e.g., because an intermediate relay is offline), the protocol falls back to flood routing for retransmission.

Advantages of direct routing over flood routing include reduced airtime consumption (fewer nodes retransmit), lower collision probability, and reduced battery consumption across the network. The trade-off is that direct routes may become stale as topology changes.

5.4 Zero-Hop Mode

Zero-hop mode is a local broadcast mode in which the packet is transmitted once and is not relayed by any node. Zero-hop transmission is used for:

- **ADVERT packets:** Node advertisements are broadcast in zero-hop mode to announce presence to immediate neighbors without flooding the network.
- **Neighbor discovery:** Nodes use zero-hop broadcasts to identify other nodes within direct radio range.
- **Local commands:** CLI commands directed at nodes within direct range may use zero-hop mode.

5.5 Transport Mode

Transport mode provides end-to-end acknowledged delivery over the mesh network. It is used when reliable delivery is required (e.g., for OTA firmware updates, sensor data uploads, or critical messages). The transport-mode handshake follows this sequence:

7. **REQ (Request):** The sender transmits a REQ packet containing the destination hash, source hash, and a MAC derived from a shared secret.
8. **RESPONSE:** The destination validates the MAC and replies with a RESPONSE packet containing the requested data or confirmation.
9. **ACK:** The sender acknowledges receipt of the RESPONSE.

If the sender does not receive a RESPONSE within the configured timeout, it retransmits the REQ with exponential backoff: the first retry occurs after T_{base}, the second after $2 \cdot T_{base}$, the third after $4 \cdot T_{base}$, up to a maximum retry count (typically 3–5).

5.6 Relay Decision Logic

The following pseudocode describes the per-hop relay decision executed by each relay node upon receiving a packet:

```
Algorithm 5-1: Per-Hop Relay Decision
--------------------------------------------------------------------------------

FUNCTION relay_decision(packet, local_config):

    // Step 1: Check node role
    IF local_config.role == COMPANION THEN
        RETURN DROP        // Companion nodes never relay

    // Step 2: Check TTL
    IF packet.ttl <= 0 THEN
        RETURN DROP        // TTL expired

    // Step 3: Check duplicate hash
    hash = compute_hash(packet.payload)

    IF hash IN local_config.hash_cache THEN
        RETURN DROP        // Duplicate packet
    ELSE
        local_config.hash_cache.insert(hash)

    // Step 4: Check routing type
    IF packet.route_type == ZERO_HOP THEN
        RETURN DROP        // Zero-hop packets are not relayed

    IF packet.route_type == DIRECT THEN
        IF local_config.node_hash NOT IN packet.path[expected_position] THEN
            RETURN DROP    // This node is not on the direct path

    // Step 5: Apply RSSI/SNR filter
    IF packet.rssi < local_config.rssi_floor THEN
        RETURN DROP        // Signal too weak

    IF packet.snr < local_config.snr_floor THEN
        RETURN DROP        // SNR below threshold

    // Step 6: Apply random backoff
    backoff = RANDOM(0, local_config.t_backoff_max)
    WAIT(backoff)

    // Step 7: Re-check hash cache after backoff
    // (another relay may have already forwarded)
    IF hash IN local_config.hash_cache_updated THEN
        RETURN DROP

    // Step 8: Retransmit with fresh preamble
    packet.ttl = packet.ttl - 1
    packet.path.append(local_config.node_hash)

    GENERATE_FRESH_PREAMBLE()
    TRANSMIT(packet)

    RETURN RELAYED
```

Chapter 6 — Collision Domains and Timing Architecture

6.1 Collision Domain Definition

In a LoRa mesh network, a *collision domain* is defined as the set of all nodes that can receive each other's transmissions on the same frequency, spreading factor, and bandwidth. When two or more nodes within the same collision domain transmit simultaneously, a *collision* occurs: the overlapping signals destructively interfere, and neither packet is successfully demodulated (except in cases where the *capture effect* applies—see below).

The LoRa capture effect permits successful demodulation of the stronger signal when two co-channel signals are received with a power differential exceeding approximately 6 dB. However, this mitigation is unreliable in dense networks where multiple transmitters may have similar path losses to the receiver.

6.2 Collision Probability Model

For a network of N nodes, each transmitting with a per-interval duty cycle *d*, the probability that a given transmission collides with at least one other transmission is approximately:

$$P_{collision} \approx 1 - (1 - d)^{(N-1)}$$

This model assumes transmissions are independently and uniformly distributed in time (Poisson approximation).

Table 6-1 — Collision Probability vs. Node Count (per transmission)

N (Nodes)	d = 0.1%	d = 0.5%	d = 1.0%	d = 2.0%	d = 5.0%
5	0.4%	2.0%	3.9%	7.8%	18.5%
10	0.9%	4.4%	8.6%	16.6%	37.0%
20	1.9%	9.1%	17.4%	31.6%	60.3%
50	4.8%	21.8%	38.9%	62.5%	92.3%
100	9.5%	39.4%	63.4%	86.7%	99.4%

These values underscore the importance of minimizing duty cycle per node and implementing collision-avoidance strategies (random backoff) in dense deployments.

6.3 Timing Architecture

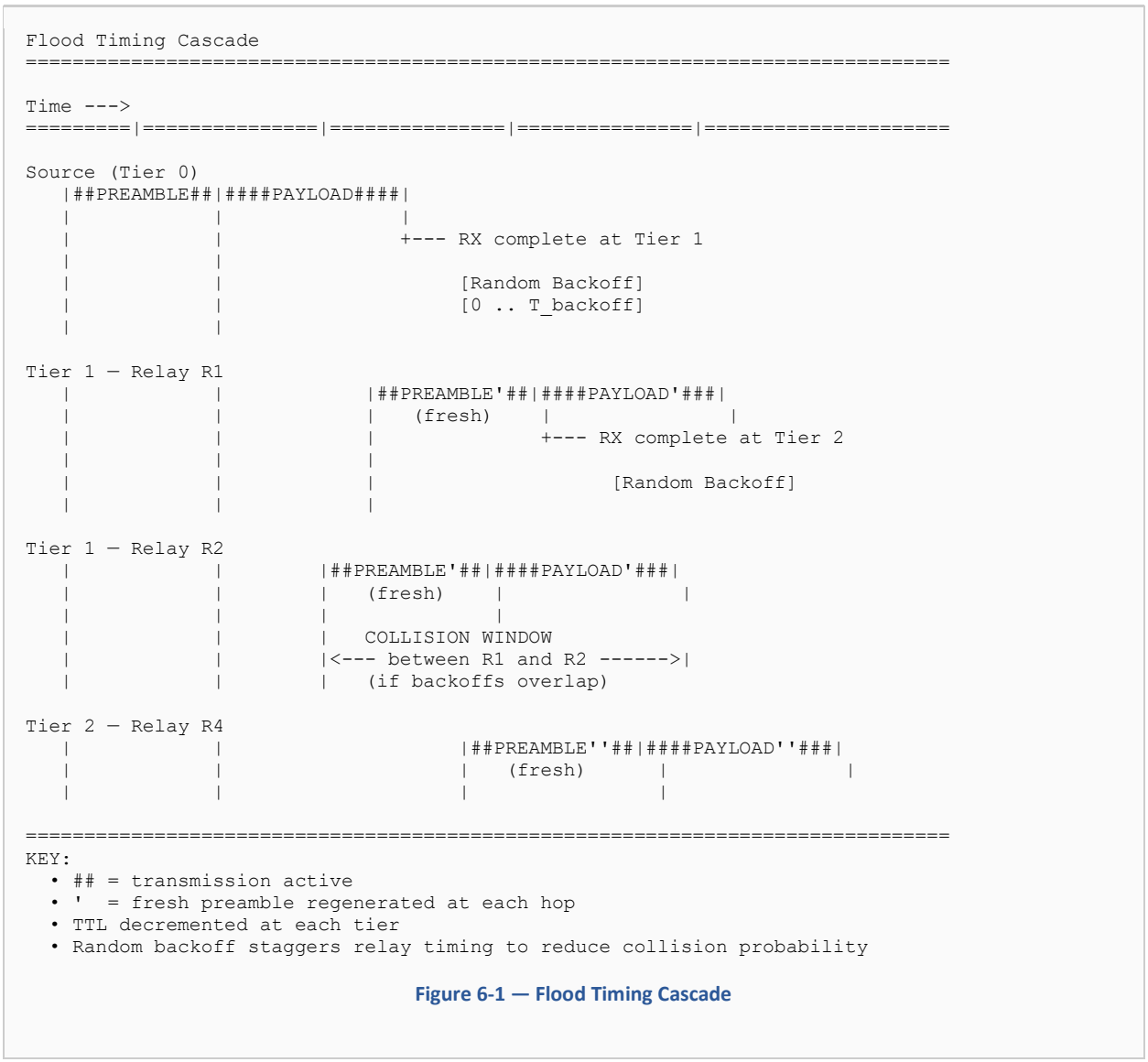


Figure 6-1 — Flood Timing Cascade

6.4 Random Backoff Strategy

To reduce the probability of collision among equidistant relay nodes (which receive the same packet at approximately the same time), each relay node inserts a random delay before retransmission. The backoff interval is drawn from a uniform distribution:

$$T_{backoff} \sim \text{Uniform}(0, T_{backoff\_max})$$

The value of $T_{backoff\_max}$ is typically set as a function of the time-on-air for the current packet and SF configuration, ensuring that the backoff window is long enough to spread relay transmissions across time but short enough to avoid excessive latency.

6.5 Airtime Budgeting

Total airtime per packet comprises the preamble duration, header duration, payload duration, and CRC duration. The preamble duration is:

$$T_{preamble} = (N_{preamble} + 4.25) \times T_{sym}$$

where the 4.25 accounts for the 2 sync-word chirps and 2.25 SFD down-chirps.

Table 6-2 — Airtime vs. Payload Size (CR 4/5, 8 preamble symbols)

SF / BW	10 B Payload	50 B Payload	100 B Payload	184 B Payload
SF7 / 125 kHz	36 ms	72 ms	118 ms	195 ms
SF8 / 125 kHz	62 ms	123 ms	206 ms	349 ms
SF9 / 125 kHz	114 ms	226 ms	390 ms	657 ms
SF10 / 125 kHz	206 ms	411 ms	739 ms	1,270 ms
SF11 / 125 kHz	413 ms	823 ms	1,478 ms	2,539 ms
SF12 / 125 kHz	825 ms	1,646 ms	2,957 ms	5,079 ms
SF7 / 250 kHz	18 ms	36 ms	59 ms	98 ms
SF10 / 250 kHz	103 ms	206 ms	370 ms	635 ms

6.6 Collision Domain Diagrams

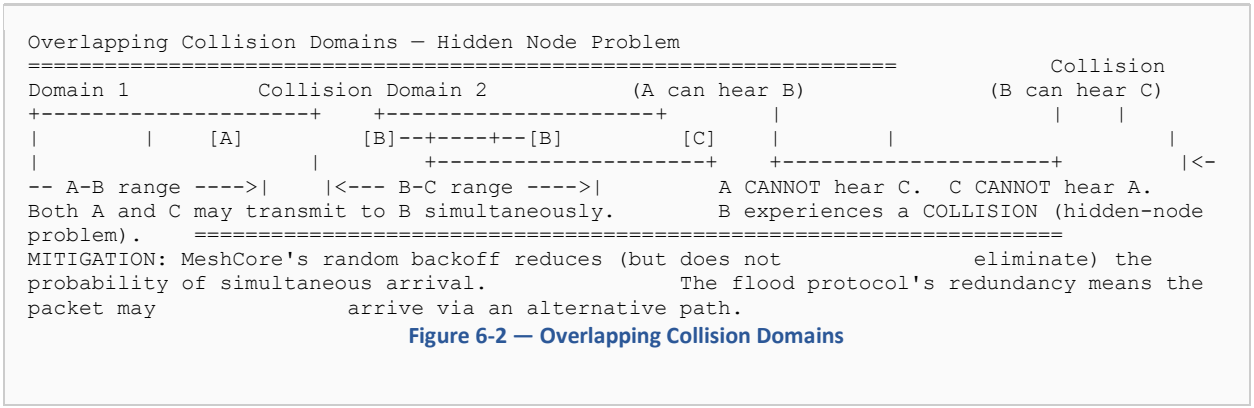


Figure 6-2 — Overlapping Collision Domains

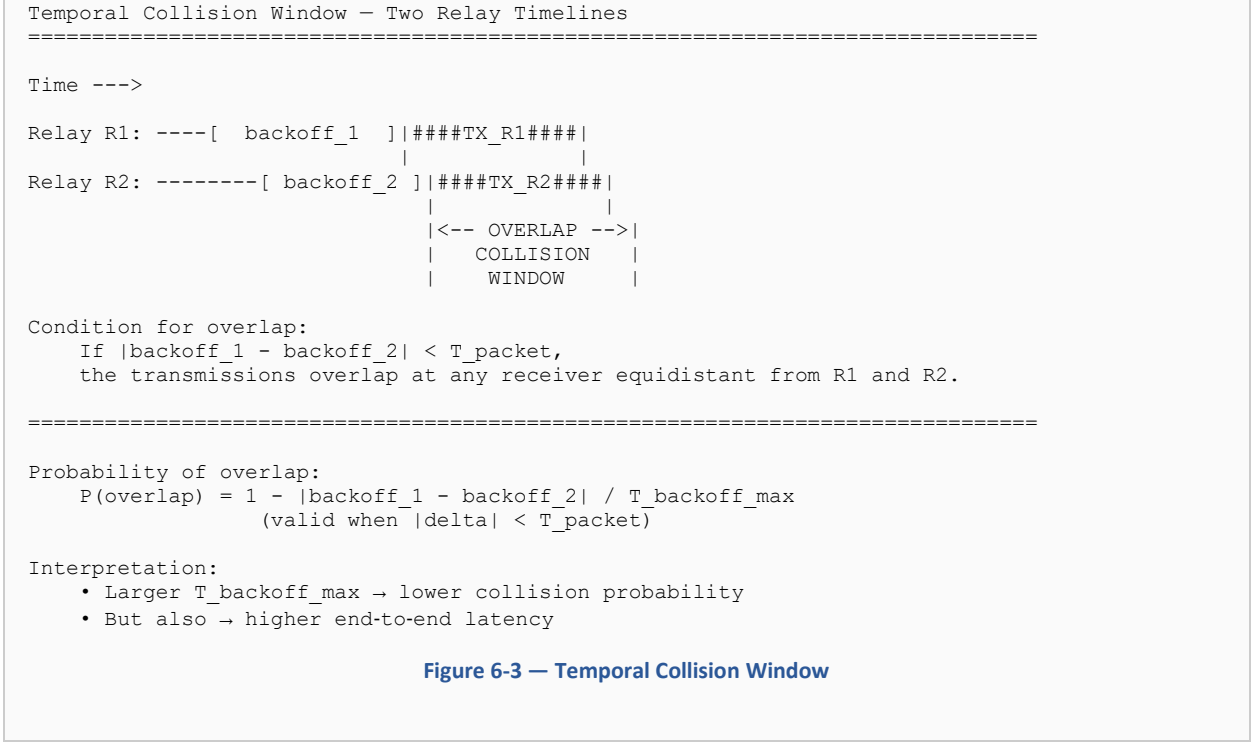


Figure 6-3 — Temporal Collision Window

Chapter 7 — Security and Identity Model

7.1 Node Identity

Each MeshCore node generates an Ed25519 keypair at first boot. The *public key* serves as the node's cryptographic identity, and a truncated hash of the public key (6 bytes) serves as the node's network address. This approach provides the following properties:

- **Self-sovereign identity:** No central authority is required to assign or validate node addresses. Each node independently generates its own identity.
- **Collision resistance:** The 6-byte (48-bit) address space provides approximately 281 trillion unique addresses, making accidental collisions statistically negligible for practical network sizes.
- **Cryptographic binding:** The address is derived from the public key, preventing address spoofing without possession of the corresponding private key.

7.2 Encryption

MeshCore supports end-to-end encryption for direct messages and group encryption for room-based communication:

- **Direct messages:** The sender and recipient perform an X25519 Elliptic Curve Diffie-Hellman (ECDH) key exchange using their Ed25519 keys (converted to X25519 format). The resulting shared secret is used to derive an AES-256-CTR or ChaCha20 session key. All direct-message payloads are encrypted with this key, ensuring that intermediate relay nodes cannot read the message content.
- **Room messages:** Room Servers distribute a group symmetric key to authorized room members. Group text messages are encrypted with this shared key, providing confidentiality from non-members while allowing efficient multicast.
- **Flood messages:** General-purpose flood broadcasts (e.g., ADVERT) may be transmitted in cleartext, as their content is intended for all network participants.

7.3 Authentication

Message authentication ensures that packets originate from the claimed source and have not been tampered with in transit:

- **ADVERT packets:** Signed using the sender's Ed25519 private key. Receivers verify the signature against the sender's known public key to confirm authenticity.

- **REQ/RESPONSE packets:** Authenticated using a truncated HMAC derived from the shared ECDH secret between sender and recipient. The 4-byte MAC provides lightweight authentication suitable for constrained payloads.
- **Transport-mode packets:** The transport-code fields and MAC collectively authenticate the transport session, preventing unauthorized injection of transport-mode traffic.

7.4 Threat Model

MeshCore's security model addresses the following threat categories:

Threat	Attack Vector	Mitigation
Eavesdropping	Passive listening on the shared radio channel.	End-to-end encryption (X25519 + AES-256-CTR / ChaCha20). Relay nodes cannot decrypt direct-message payloads.
Replay Attacks	Re-injecting previously captured packets.	Hash cache detects duplicate packets. Nonce-based encryption prevents payload replay.
Sybil Attacks	An adversary creates multiple fake identities to influence routing or flooding.	Key pinning in Rooms: Room Servers maintain an authorized-member list based on Ed25519 public keys. Unknown keys are rejected.
Jamming	Continuous or burst interference to deny service.	Inherent LoRa CSS resistance to narrowband interference. CAD enables rapid channel-assessment to avoid transmitting during jamming. Geographic diversity of relay paths provides alternative routes.
Packet Injection	Forging packets with spoofed source addresses.	MAC/signature verification at the destination. Spoofed packets fail authentication and are discarded.
Path Manipulation	Altering path hashes in transit to redirect direct-routed packets.	Path integrity is implicitly verified: if a relay node's hash is not at the expected position, the packet is dropped. Encrypted payloads prevent content manipulation.

Chapter 8 — Hardware Platforms and Integration

8.1 Supported Platforms

MeshCore firmware supports a range of LoRa-capable embedded platforms based on ESP32, nRF52840, and STM32 microcontrollers. The following table summarizes the primary supported hardware:

Table 8-1 — Supported Hardware Platforms

Platform	MCU	Radio	Flash	RAM	Key I/O
Heltec V3	ESP32-S3	SX1262	8 MB	512 KB	OLED, USB-C, Wi-Fi, BLE
LILYGO T-Beam v1.2	ESP32	SX1262 / SX1276	4 MB	520 KB	GPS, 18650 battery, BLE
LILYGO T-Echo	nRF52840	SX1262	1 MB	256 KB	E-ink display, GPS, BLE
RAK4631 (WisBlock)	nRF52840	SX1262	1 MB	256 KB	Modular WisBlock I/O, BLE
RAK11200 (WisBlock)	ESP32	SX1262 (via RAK13300)	4 MB	520 KB	Wi-Fi, BLE, WisBlock I/O
Station G2	ESP32-S3	SX1262	16 MB	512 KB	Ethernet, USB, high-power TX

The RAK4631 (nRF52840-based) is particularly well-suited for solar-powered repeater deployments due to the nRF52840's ultra-low-power sleep modes (typical sleep current ~2 µA) and the WisBlock ecosystem's modular solar-charging and sensor boards.

8.2 Sensor Integration

MeshCore supports sensor data collection and transmission through the PAYLOAD_TYPE_SENSOR_DATA payload type. The RAK WisBlock ecosystem provides a standardized sensor interface with the following commonly deployed modules:

- **RAK12035:** Capacitive soil moisture sensor (volumetric water content, temperature).
- **RAK1901:** Temperature and humidity sensor (SHTC3, ±0.2°C accuracy).
- **RAK1902:** Barometric pressure sensor (LPS22HB, ±0.1 hPa).
- **RAK1904:** 3-axis accelerometer (LIS3DH) for motion detection and orientation.
- **RAK12500:** GNSS module (u-blox ZOE-M8Q) for position reporting.

Sensor rail sequencing on RAK Board Support Packages (BSP) follows a defined power-up order: the sensor power rail is enabled, a stabilization delay is observed (typically 100–500 ms depending on the sensor), the I2C bus is initialized, and sensor readings are acquired. This sequencing prevents spurious readings from partially powered sensors.

8.3 Companion Radio Architecture

In companion mode, a MeshCore node operates as a radio modem for a connected host device (smartphone, laptop, or tablet). The host application communicates with the companion radio over BLE (Bluetooth Low Energy) or USB serial using a defined binary protocol. Key characteristics of companion mode include:

- **No relay function:** Companion nodes do not relay packets, preventing suboptimal routing paths through user endpoint devices.
- **Transparent bridging:** The companion radio handles all mesh-layer operations (packet framing, routing decisions, encryption) and presents a simple message-oriented interface to the host application.
- **Multiple application support:** The companion protocol supports concurrent connections from multiple applications on the host device.

8.4 OTA Update Mechanism

MeshCore supports over-the-air (OTA) firmware updates for remote nodes using the transport-mode delivery mechanism. The OTA process operates as follows:

10. **Image preparation:** The firmware binary is divided into fixed-size chunks (typically 128–256 bytes each) and a manifest is generated containing the total image size, chunk count, and a CRC-32 checksum of the complete image.
11. **Manifest transmission:** The update initiator sends the manifest to the target node via transport mode (REQ/RESPONSE). The target validates the manifest and acknowledges readiness.
12. **Chunk transfer:** Each chunk is transmitted sequentially via transport mode with per-chunk sequence numbering and CRC verification. Failed chunks are retransmitted using the transport-mode exponential backoff mechanism.

13. **Image verification:** Upon receiving all chunks, the target node reassembles the firmware image and validates the overall CRC-32 against the manifest. If verification succeeds, the node reboots into the new firmware.

Chapter 9 — Operational Considerations

9.1 Deployment Planning

Effective MeshCore deployment requires balancing coverage, reliability, and collision avoidance. The following guidelines inform deployment planning:

- **Node placement:** Repeater nodes should be placed at elevated positions with clear line-of-sight to neighboring nodes. Hilltops, building rooftops, and towers are preferred locations. Each repeater should ideally have line-of-sight to at least two other repeaters to provide path redundancy.
- **Relay density vs. collision trade-off:** Adding more repeaters improves coverage and path redundancy but increases the number of nodes in each collision domain, raising collision probability. The recommended relay density is 3–8 repeaters per collision domain for networks using SF10 or higher.
- **Antenna selection:** Omnidirectional antennas (e.g., ground-plane verticals, J-pole antennas) are recommended for repeater nodes to provide 360-degree coverage. Directional antennas (e.g., Yagi) may be used for point-to-point links between specific repeater pairs where focused coverage is desired.
- **Power supply:** Solar-powered repeater installations should be dimensioned for worst-case seasonal insolation. A 6W solar panel with a 6000 mAh LiPo battery is a reasonable minimum for a RAK4631-based repeater in temperate climates.

9.2 Monitoring and Telemetry

MeshCore nodes periodically generate telemetry data (PAYLOAD_TYPE_STATS) that enables remote monitoring of network health. Key telemetry metrics include:

- **RSSI/SNR statistics:** Per-neighbor signal quality measurements, including min, max, and moving average.
- **Packet counts:** Total packets received, relayed, originated, and dropped (with drop reason: duplicate, TTL expired, CRC failure, RSSI below floor).
- **Duty cycle utilization:** Percentage of time spent transmitting within the regulatory measurement interval.
- **Battery voltage:** Current battery voltage and estimated remaining capacity.
- **Uptime:** Time since last reboot, useful for detecting instability or watchdog resets.

9.3 Regulatory Compliance

MeshCore deployments must comply with the ISM-band regulations applicable to the operating region:

Region	Frequency Band	Regulation	Key Constraints
United States	902–928 MHz	FCC Part 15.247	Max +30 dBm conducted, FHSS or digital modulation, no duty-cycle limit per se.
European Union	863–870 MHz	ETSI EN 300 220	1% duty cycle (36 s/hr per sub-band), max +14 dBm ERP.
Australia/NZ	915–928 MHz	ACMA / RSM	Max +30 dBm EIRP, spectral density limits.
Japan	920–928 MHz	ARIB STD-T108	Max +13 dBm, LBT (Listen Before Talk) required.
India	865–867 MHz	WPC / DoT	Max +30 dBm EIRP, 1 W max TX power.

Operators are responsible for configuring the appropriate frequency, TX power, and duty-cycle parameters for their regulatory domain. MeshCore firmware includes region-specific defaults that enforce regulatory limits when properly configured.

9.4 Troubleshooting

Common failure modes encountered in MeshCore deployments and their diagnostic procedures:

Failure Mode	Symptoms	Diagnostic Procedure	Resolution
Hidden-node collisions	Intermittent packet loss at nodes served by multiple equidistant repeaters.	Monitor packet-drop counters at the affected node. Check for CRC failures correlating with other repeaters' TX activity.	Adjust relay placement to reduce equidistant geometries. Increase T_backoff_max.
Route flapping	Alternating between direct and flood routes for the same destination. Increased latency variability.	Examine hint-update logs. Look for hints with low consistency counts being rapidly replaced.	Raise consistency threshold. Increase RSSI/SNR floor to exclude marginal links from hint learning.
Clock drift	Gradual increase in packet loss on long-running links, especially at high SF.	Compare node uptimes. Monitor symbol-timing errors (if available in debug output).	Ensure hop-level preamble regeneration is functioning (default behavior). Replace nodes with crystal-oscillator drift exceeding ±20 ppm.
Duty-cycle exhaustion	Node stops transmitting for extended periods (EU deployments). Telemetry shows duty-cycle utilization near 100%.	Check duty-cycle counters. Identify sources of excessive traffic (e.g., sensor nodes transmitting too frequently).	Reduce sensor polling interval. Reduce TTL to limit flood propagation. Use direct routing where possible.