



Testbed for LoRa Mesh Networks

Building a unified test header and pipeline to compare mesh behaviors under identical conditions

Abdullah Almanie
abdullahalmanei@g.ucla.edu

Mentor: Dr. Moez Altayeb

Mentor: Dr. Marco Zennaro

Mentor: Salah Abdeljabar

August 27, 2025

Abstract

This report documents an end-to-end effort to (i) instrument the *Meshtastic* firmware with a compact, mesh-agnostic test header to produce reproducible, analyzable run logs and (ii) design and implement a standalone mesh protocol ("LoRa QTree") that embeds the exact same header and testing workflow. The outcome is a fair A/B methodology where both systems can be exercised and compared along the same Key Performance Indicators (KPIs): end-to-end Packet Delivery Ratio (PDR), hop count, airtime utilization (duty-cycle compliance), and energy proxy via battery voltage. We detail the code changes in both stacks (TX path header insertion, RX path detection, hop incrementing, duplicate handling), the gateway accounting logic, and a lenient duty-cycle token-bucket that respects EU 1% constraints while allowing realistic responses (ACK/QUERY). We conclude with a roadmap for multi-node trials and open-sourcing a browser-based flasher to lower the barrier to replication.

Contents

Abstract **1**

1 Introduction **3**

2 System Overview **3**

3 Methodology and Approach **4**
3.1 Meshtastic Instrumentation 4
3.2 LoRa QTree Protocol 4
3.3 KPIs (Definitions) 5

4 Progress and Milestones **5**

5 Preliminary Results **6**

6 Challenges and Obstacles **6**

7 Next Steps and Future Work **6**

8 Resources and Tools **6**

1 Introduction

Motivation

LoRa-based meshes exist in many flavours, yet systematic, apples-to-apples comparisons remain rare. A key blocker is that different stacks expose different telemetry and use different logging conventions, which confounds comparisons.

Objectives and Goals

- Introduce a mesh-agnostic test header that can be embedded into any payload without breaking application semantics.
- Instrument Meshtastic to emit and relay frames carrying this header, with minimal intrusion.
- Build a separate, clean-room LoRa QTree implementation that uses the same header and a comparable gateway accounting loop.
- Provide a repeatable KPI pipeline: PDR, hop-count distribution, airtime use vs. EU 1%, energy proxy via battery mV, and parent-churn.
- Release a web-based flasher to simplify replication on commodity boards.

2 System Overview

Hardware

T-Beam Supreme (ESP32-S3 + SX1262 + AXP2101 PMU), Heltec V3 (ESP32-S3 + SX1262). Common LoRa PHY: 868 MHz, BW=125 kHz, SF=12, CR=4/5.

Software Components

- **Meshtastic (instrumented):** stock mesh with TX/RX hooks to handle the Test Header; logs via serial for a Python collector.
- **LoRa QTree:** small mesh with JOIN/ACK/QUERY/STATE and constrained hop-forwarding; gateway maintains per-node liveness and misses.
- **Web Flasher:** static site to flash target binaries from browser (Flasher).
- **Python tools:** real-time tail/aggregator and offline analyzer to compute KPIs from logs.

Unified Test Header

```
struct test_hdr_t {
  uint8_t  ver;         // 1
  uint32_t test_id;     // campaign/version marker
  uint32_t seq;         // per-sender monotonic
```

```
  uint32_t src;         // sender ID
  uint32_t tx_epoch_ms; // sender millis() at TX
  uint8_t  hop_cnt;     // incremented by each relay
  uint16_t batt_mV;     // energy proxy (0 if unknown)
};
```

This header rides *inside* the application payload and does not require PHY/MAC changes. Both Meshtastic and LoRa QTree write/read the same layout.

3 Methodology and Approach

3.1 Meshtastic Instrumentation

TX path (insertion). We adjust the message assembly to move the original payload forward by `sizeof(test_hdr_t)`, write the header, cap total to 255 bytes, and preserve encryption/auth if enabled. We stamp `ver, test_id, seq, src, tx_epoch_ms, batt_mV`; `hop_cnt` is set to 0 at source.

RX path (detection + relay semantics). In the interrupt-driven receive handler, we check if the first bytes match a plausible header, then:

1. Increment `hop_cnt` before forwarding.
2. Apply duplicate filtering using (`src`, `seq`).
3. Log a canonical `TEST` line containing Received Signal Strength Indicator (RSSI)/Signal-to-Noise Ratio (SNR), hop count, and header fields.

This does not bypass Meshtastic routing; it only annotates payloads to create a comparable trace across stacks.

Why secure. If a channel key is configured, the header sits inside the encrypted application payload; only endpoints with keys can decode the fields. For plaintext tests, the header is still self-describing for the sniffer/logger.

3.2 LoRa QTree Protocol

Header. `MeshHeader` contains magic, src, dst, hops, type, len (6 bytes). Types: `BEACON, JOIN_REQ, JOIN_ACK, DATA_UP, DATA_ACK, QUERY, STATE`, plus `CHILD_ADD/GONE, JOIN_NACK`.

Gateway loop. Tracks children with fields `lastSeen, lastRssi, lastQuery, misses, answeredS` sends `QUERY`; increments `misses` only if a `QUERY` was successfully sent and no subsequent frame arrives within timeout; purges on absolute silence. JOIN_ACK is rate-limited and queued if deferred by duty-cycle, then activates the child *only after* the ACK actually transmits.

Duty-cycle control. A lenient token-bucket enforces EU 1% over a one-hour window: we earn 1 ms of TX per 100 ms wall time, allow temporary borrowing (up to 12 s), and delay future TXs until the debt is repaid. This allows timely ACK/QUERY without long dead-times while still respecting policy on average.

Battery proxy. On T-Beam Supreme, AXP2101 PMU exposes battery voltage; Heltec V3 falls back to 0 if PMU unavailable. The value is written into `batt_mV` in the test header for energy trending.

3.3 KPIs (Definitions)

We use identical definitions across both stacks:

- **PDR** per origin: PDR $= \frac{\#\text{ unique TEST seq received at GW}}{\#\text{ TEST seq transmitted by origin}}$. Duplicates are excluded via (`src`, `seq`).
- **Hop count distribution:** histogram of final `hop_cnt` at GW.
- **Airtime utilization:** per-node TX time over wall time in the DC window; we report fraction vs. 1% cap.
- **Energy proxy:** slope of `batt_mV` over time, normalized to mV/hour.
- **Parent churn (LoRa QTree):** number of JOIN/DETACH events per hour; **miss streak** as consecutive unanswered queries before recovery.
- **Latency (optional):** if clocks are profiled, end-to-end latency $\approx$ GW_rx_ms $-$ src.tx_epoch_ms with caveats on unsynchronized timebases.

4 Progress and Milestones

1. Meshtastic firmware modified to embed/read the Test Header; live serial logger captures canonical `TEST` lines.
2. LoRa QTree implemented with the same header semantics; JOIN/ACK/QUERY/STATE and multi-hop forwarding with capped `MAX_HOPS`.
3. Gateway accounting corrected to count *misses only after a sent QUERY*; reset liveness on *any* frame.
4. Token-bucket DC control and per-node TX queue (JOIN_ACK/QUERY) to avoid starvation under DC constraints.
5. Web flasher prototype deployed for frictionless flashing.

5 Preliminary Results

We validated small topologies (2–5 nodes) indoors:

- Under identical PHY settings, both stacks produced comparable PDR ($> 95\%$ in line-of-sight).
- LoRa QTree's explicit QUERY loop made parent-loss visible via miss streaks and triggered re-attachment; Meshtastic relied on its native route maintenance but the shared header still exposed hop dynamics.
- The DC token-bucket enabled responsive ACK/QUERY bursts while keeping average airtime within 1% over an hour.

A larger campaign is planned to characterize scaling and lossy links.

6 Challenges and Obstacles

The main blocker was the inability to run long, controlled trials with many nodes and realistic obstacles due to time and facility constraints. That limited statistical confidence for PDR/latency curves. We mitigated by unifying the telemetry so later trials can be run without code churn.

7 Next Steps and Future Work

- Expand to outdoor multi-hop with obstructions and distance sweeps; collect PDR vs. hop count curves.
- Add clock-profiling to enable proper latency measurement.
- Publish a dataset and analysis notebooks for reproducibility.
- Generalize the Test Header into a library crate usable by other LoRa stacks.

8 Resources and Tools

PlatformIO (ESP32), RadioLib, XPowersLib, U8g2, PCF8563 RTC, Python (serial + pandas). Firmware releases and a browser flasher are available: GitHub repo, Web Flasher.

Acknowledgments

This work was created during the **ICTP TRIL Fellowship** and the **KAUST Academy**, supervised by **Dr. Moez Altayeb**, **Dr. Marco Zennaro**, and **Salah Abdeljabar**. We also acknowledge the foundational ideas in RSSI-based hop-counting and mesh maintenance [1].

References

[1] Huang-Chen Lee and Kai-Hsiang Ke. "Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation". In: *IEEE Transactions on Instrumentation and Measurement* (Mar. 2018). DOI: 10.1109/TIM.2018.2814082.