Optimizing MeshCore for Large Networks

Deploying MeshCore at scale of 50 or more nodes requires deliberate planning of repeater placement, advertisement strategy, and congestion avoidance.

Repeater Placement for Path Diversity

Path diversity is the single most important design principle for a resilient large-scale MeshCore network. Without path diversity, a single repeater failure can partition the network.

- Place at least two independent elevated repeaters within mutual radio range for backbone redundancy. A backbone arranged as a ring (each node connected to two neighbours so they form a single cycle) is 2-edge-connected: it stays connected after any single link fails. Note that minimum degree 2 at every node is necessary but not sufficient for 2-edge-connectivity on its own (e.g., two loops joined by a single bridge link still has a single point of failure).
- Gateway repeaters bridging two disconnected clusters should always have a backup gateway repeater or a direct link if path loss permits.
- Use a network graph tool to identify any repeater whose removal disconnects the graph. These cut vertices require remediation through additional repeater placement.

Minimum Viable Topology for 50 Nodes

- 4-6 elevated backbone repeaters, each with at least 2 other backbone repeaters in range
- 10-15 mid-tier repeaters, each within range of at least 2 backbone repeaters
- 30+ client nodes connecting to the nearest mid-tier repeater or directly to backbone

Advertisement Flood Strategy

In large networks, poorly tuned advertisement intervals become a significant source of congestion. MeshCore supports two advert modes: a **flood advert** (propagated hop-by-hop to all reachable nodes; CLI `advert`) and a **zero-hop advert** (broadcast to in-range neighbours only, not repeated; CLI `advert.zerohop`). Cadence is set with `set flood.advert.interval <hours>` for flood adverts and `set advert.interval <minutes>` for zero-hop adverts.

Recommended: backbone repeaters use flood adverts; mid-tier repeaters use flood adverts with a reduced interval; fixed client nodes use a long flood interval or zero-hop adverts; mobile client nodes use zero-hop or more frequent flood adverts; temporary nodes use zero-hop adverts to avoid polluting neighbour tables.

Advertisement Interval Tuning

Advert intervals must be entered in the units the firmware uses. The flood advert interval (`set flood.advert.interval`) is in **hours** (range 3-168, default 12). The zero-hop advert interval (`set advert.interval`) is in **minutes** (range 60-240). These repeater CLI settings apply to repeater/room-server nodes; the advert cadence for non-repeating client/companion nodes is controlled in the companion app, not via these CLI settings.

- Backbone repeaters (flood adverts): toward the shorter end of the range, e.g. 6-12 hours in busy networks
- Mid-tier repeaters (flood adverts): a moderate interval, e.g. 12 hours (the default)
- Fixed client nodes: a long interval, or app-controlled adverts for non-repeating nodes
- Mobile client nodes: more frequent app-controlled adverts; the repeater zero-hop CLI minimum is 60 minutes

Path Learning and Re-discovery

MeshCore uses a flood-first / direct-route-after model rather than a user-tunable route cache. The first message to a destination is flooded and the working path is learned as a byproduct; later messages reuse that learned path until it goes stale and a re-flood occurs. There is no `route-cache-timeout` setting or named "route cache" tuning profile in the firmware, so there are no minute-based cache lifetimes to configure. In high-mobility or rapidly-changing deployments, expect more frequent re-floods as learned paths break and are rediscovered.

Congestion Avoidance

LoRa is half-duplex: no node can transmit and receive simultaneously. Congestion manifests as elevated packet loss, increased delivery latency, and packet collisions. For flood retransmits, MeshCore uses a random backoff window (scaled by the `txdelay` / `direct.txdelay` factors) so that

repeaters hearing the same flood packet do not retransmit simultaneously; this is a random-backoff scheme rather than full carrier-sense CSMA/CA on the mesh routing path. (The separate KISS-modem firmware does implement p-persistent CSMA.)

Key mitigations: (1) reduce advertisement flood frequency -- the highest-leverage tuning parameter; (2) segment with frequency separation across two independent LoRa channels bridged by backbone repeaters; (3) on a sparse, long backbone link a higher spreading factor improves link budget and sensitivity, but note that all nodes on a MeshCore channel must share one SF/BW/CR to interoperate -- you cannot raise SF on individual links while staying on the same channel, and each +1 SF roughly doubles symbol airtime, which worsens congestion (current USA/Canada guidance actually uses a low SF7 / BW62.5 / CR5 preset on purpose); (4) use zero-hop adverts for high-density client clusters.

Monitoring Network Health with MeshCore Statistics Commands

```
neighbors
stats-core
stats-radio
stats-packets
advert
```

MeshCore does not expose AODV-style metrics such as a "route cache hit rate" or "RERR rate" -- it has no route-error (RERR) packets and no route table, because routing is flood-first / direct-route-after rather than reactive AODV. Monitor only the signals the firmware actually reports: the `neighbors` list (limited to the 8 most recent adverts, each encoded as `{pubkey-prefix}:{timestamp}:{snr*4}`), and the counters in `stats-core`, `stats-radio` (RSSI, SNR, noise floor) and `stats-packets` (packet counts, RX errors, airtime). When judging link quality, keep RSSI (an absolute received-power figure in dBm) separate from SNR (in dB relative to the noise floor); LoRa can decode several dB below the noise floor depending on spreading factor.

For persistent monitoring, you can build community tooling on top of the MeshCore Python companion API running on a Raspberry Pi attached to a local node. Note that the companion protocol talks to the locally-connected node, not arbitrarily to "all reachable repeaters" -- pulling stats from a remote repeater requires authenticated `login`/`cmd` queries over the mesh. Any InfluxDB/Prometheus/Grafana pipeline is a custom build, not a feature that ships with MeshCore.

Revision #8
Created 2026-05-03 06:20:57 UTC by Mesh America Admin
Updated 2026-06-09 14:26:44 UTC by Mesh America Admin