MeshCore Routing Explained

- Path Discovery and Route Learning
- Why MeshCore Scales Better Than Flooding

Path Discovery and Route Learning

MeshCore uses a hybrid **flood-first / direct-route-after** approach rather than persistent flooding. For stable repeat-unicast traffic the network quiets down as paths are learned; under mobility, link churn, or heavy group/broadcast traffic it can get louder again because of re-flooding.

How Path Discovery Works: Step by Step

1. **First message to an unknown destination** - MeshCore floods the network to locate the target node. The flooded message carries the payload, and every node relays it until it reaches the destination, recording the path it travelled as a byproduct.
2. **Destination returns a path record** - The destination sends back a `PAYLOAD_TYPE_PATH` packet containing the recorded path (not merely a generic ACK).
3. **Sender stores the path** - The original sender records the returned path and uses it (`ROUTE_TYPE_DIRECT`, embedding the path) for subsequent messages. Relay nodes forward based on the path embedded in each packet; they do **not** maintain per-destination route tables. This is the key step: the sender has now "learned" a path.
4. **Subsequent messages use the established path** - Only nodes on the known route retransmit. All other nodes stay silent.
5. **Retry and re-discover** - After 3 consecutive failed retries on a known path, MeshCore discards the cached route and floods again to find a new one.

Group and Public Channel Messages

Group messages and public channel broadcasts **always flood** the network, because they are addressed to multiple destinations and no single path can serve all recipients. Path caching only applies to direct (unicast) messages.

Path Hash Mode

The `path.hash.mode` setting controls the path-hash size used in a node's own advert broadcasts (on a repeater it affects only its advert broadcasts, not which packets it forwards). On companion nodes, the message path-hash size is set in the app's Experimental Settings.

```
set path.hash.mode 0 # 1-byte path hash (default; low overhead)
set path.hash.mode 1 # 2-byte path hash (balanced)
set path.hash.mode 2 # 3-byte path hash (highest precision)
```

A larger hash reduces the chance of path collision in dense networks at the cost of slightly larger packet headers. The firmware default is mode 0 (1 byte); only raise it after confirming the whole network runs firmware that supports the larger size.

Advertisement Broadcasts

Nodes periodically broadcast advertisements so neighbors can discover them. The flood-advert interval is configurable (in hours):

```
set flood.advert.interval 6 # broadcast every 6 hours (default is 12)
```

To trigger an immediate advertisement (useful after changing location or name), use `advert` for a flood advert, or `advert.zerohop` for a zero-hop (neighbors-only) advert:

```
advert
```

Why MeshCore Scales Better Than Flooding

Understanding the difference between MeshCore's flood-first, direct-route-after routing and Meshtastic's flood routing explains why the two protocols behave differently in large networks.

Flood Routing (Meshtastic)

- In a managed flood, each relay rebroadcasts each new message once — duplicates are suppressed (a node will not re-send a message it has already seen), and non-relay roles do not rebroadcast at all.
- Air time consumed grows with every additional relay node.
- Channel utilization increases as the network grows - more nodes means more congestion.
- Works well in small networks; degrades in large or dense deployments.

Flood-First, Direct-Route-After Routing (MeshCore)

- The first message to a destination is flooded (and the path is learned as a byproduct); subsequent unicast messages follow the learned path, so only nodes on that path retransmit them.
- Air time per unicast message is bounded once a path is known.
- For stable links carrying repeat unicast traffic, established paths reduce retransmissions over time. Under mobility or link churn, paths break and the sender re-floods to re-learn them, which can increase traffic — the quieting effect is not guaranteed.
- Targeted unicast forwarding avoids the per-message flooding cost, but advertisements and group/broadcast traffic still flood and grow with network size. Real-world scaling depends heavily on topology stability and broadcast volume; no specific large-network guarantee is implied.

Side-by-Side Comparison

Attribute	MeshCore (flood-first, direct-route-after)	Meshtastic (flooding)
First message to unknown node	Floods (once)	Floods (always)
Subsequent messages to known node	Path-only retransmissions	Floods (always)
Congestion as network grows	Lower for repeat unicast; group/broadcast still floods	High - grows with nodes
Average power per message at scale	Lower	Higher
Group / broadcast messages	Flood	Flood
Route failure recovery	On repeated ACK failure the path is reset to flood, and the next message re-discovers a route (the exact retry count is firmware/config dependent). Each recovery incurs a full flood, briefly increasing channel load.	N/A - always floods

Practical Implications

- **Power consumption** - Infrastructure repeaters on MeshCore consume less power per forwarded unicast message at scale because they are silent when not on an active path.
- **Frequency reuse** - Less channel congestion from repeat unicast traffic means the same frequency can support more simultaneous conversations.
- **Latency** - Once a path is established, latency is lower than re-flooding each message, but it still varies with channel congestion, half-duplex contention, duty-cycle limits, and hop count, and resets to a full re-discovery delay whenever a path breaks. Do not assume consistent or bounded latency.
- **Dense deployments** - In city-wide or event-scale deployments, MeshCore's targeted forwarding reduces the broadcast-storm pressure that can make large Meshtastic networks unreliable, though group/broadcast traffic still floods.