Hop Limit Configuration for Repeaters

The hop limit is one of the most important and most misunderstood parameters in a Meshtastic mesh. Setting it correctly reduces unnecessary rebroadcasts, controls how far a message propagates, and prevents broadcast storms that saturate the channel. This page explains exactly what `hop_limit` does, when to change it, and the CLI commands to apply it.

What hop_limit Controls

Every Meshtastic packet carries a **hop count field** in its header. This field is initialised to the `hop_limit` value configured on the *originating* node at the time the packet is sent. Each relay node (ROUTER, ROUTER_CLIENT (deprecated; use ROUTER or REPEATER instead), or REPEATER) decrements this field by 1 before rebroadcasting. When a node receives a packet with a hop count of 0 it delivers the packet locally but does *not* forward it further.

The formula for maximum relay chain length is:

```
Maximum relaying nodes = hop_limit
Total nodes that hear the original packet = hop_limit + 1 (originator counts as hop 0)
```

With the default `hop_limit = 3`, a packet originating at node A can reach:

- All nodes within direct radio range of A (hop 0 -> 1)
- All nodes reachable via one relay (hop 1 -> 2)
- All nodes reachable via two relays (hop 2 -> 3)
- All nodes reachable via three relays (hop 3 -> consumed, not forwarded)

In practice the default of 3 covers most community-sized networks with reasonable relay density.

The Default: hop_limit = 3

The firmware default of 3 is a careful balance. It is high enough to traverse a typical multi-node mesh with a few infrastructure repeaters, and low enough that a single rogue or misconfigured node cannot cause runaway rebroadcasting. Leave it at 3 unless you have a specific and measured reason to change it.

When to Increase the Hop Limit

Large geographically spread networks

If your community network spans a large geographic area - for example a county-wide emergency communications mesh with repeaters spaced 20 - 30 km apart - a hop count of 3 may not be sufficient to bridge the entire path. In this case, increasing to 4 or 5 gives messages the relay budget to traverse more infrastructure nodes.

Before increasing, first verify the actual hop count needed by running a traceroute between the two furthest nodes:

```
meshtastic --traceroute '!abcd1234'
```

Count the intermediate hops in the output. Set `hop_limit` to at least that number plus one for margin.

```
# Increase hop limit on the originating/infrastructure node
meshtastic --set lora.hop_limit 4
```

When to Decrease the Hop Limit

Small, dense urban networks

In a dense neighbourhood deployment where every node can hear at least 2 - 3 others directly, a hop limit of 3 generates significant redundant retransmissions. Consider reducing to 2 if traceroutes show no path requires more than 2 relays.

```
meshtastic --set lora.hop_limit 2
```

Local-only repeater that should not propagate far

If you deploy a repeater specifically to bridge two buildings on the same campus - not to reach the wider regional mesh - you can reduce its effective forwarding range by setting `hop_limit = 2` on packets it originates (its own NodeInfo, position, telemetry). This does not prevent it from

forwarding transit packets that arrive with a remaining count of 3, but it limits the blast radius of traffic the repeater itself generates.

For even stricter isolation, combine a low hop limit with the `lora.ignore_incoming` filter to whitelist specific node IDs that this repeater will serve.

The Broadcast Storm Risk Above hop_limit = 5

Values above 5 are explicitly discouraged by the Meshtastic project and should never be deployed on production networks. Here is why:

- Each relay creates a geometric increase in rebroadcasts. With 10 nodes each hearing 3 others, a hop_limit of 7 can generate hundreds of retransmissions per original packet.
- LoRa is a half-duplex medium. Each retransmission blocks all other transmissions in range for the duration of the air-time. Long SF12 packets at 250 bps can occupy the channel for 2 - 3 seconds each. A broadcast storm at hop_limit = 7 can push channel utilisation to 100% and render the network completely unusable.
- There is no back-off mechanism in the standard Meshtastic forwarding algorithm analogous to Ethernet CSMA/CD. Nodes will continue retransmitting even into a fully saturated channel.

If you feel you need a hop limit above 5, the correct solution is to add more infrastructure repeaters to shorten the required path, not to increase the hop counter.

Configuring hop_limit via CLI

```
# Set hop limit to 3 (default - recommended for most networks)
meshtastic --set lora.hop_limit 3

# Set hop limit to 4 (for large spread-out networks after traceroute verification)
meshtastic --set lora.hop_limit 4

# Set hop limit to 2 (for small dense networks or local-only repeaters)
meshtastic --set lora.hop_limit 2

# Verify the applied value
meshtastic --get lora.hop_limit
```

hop_limit Is a Per-Node Origination Setting

An important subtlety: `hop_limit` controls the initial value placed in packets that *this node originates*. It has no effect on packets that arrive from another node already carrying a hop count - those are forwarded as received (after decrement). Therefore:

- Setting `hop_limit = 2` on a relay node limits only that node's own originations.
- A packet originating elsewhere with `hop_limit = 5` will still traverse your relay with a remaining count of 4 after decrement - your local setting does not cap it.
- To limit how far foreign traffic propagates through your infrastructure, you must coordinate `hop_limit` settings across all nodes in your deployment, or use channel-level policies.

Monitoring Channel Utilisation After Changes

After adjusting hop_limit, monitor channel utilisation via the Meshtastic app or web UI. The target is below 25% utilisation on busy channels, below 10% on quiet channels. If you see utilisation spike after a hop_limit increase, your network density may not support the change and you should revert.

```
# Read current channel utilisation from device (shows as a percentage in debug output)
meshtastic --info | grep -i "channel util"
```

Revision #6
Created 2026-05-03 05:50:05 UTC by Mesh America Admin
Updated 2026-05-03 13:57:50 UTC by Mesh America Admin