

Troubleshooting a Misbehaving Repeater

Infrastructure repeaters are expected to operate unattended for months. When behaviour deviates from normal - excessive channel utilisation, duplicate node entries, relay failures, or complete silence - rapid and systematic diagnosis avoids unnecessary site visits and minimises community impact. This page catalogues the most common symptoms, their likely causes, and the remediation steps including serial console diagnostics.

Symptom 1: High Channel Utilisation Caused by This Node

Diagnosis

In the [Meshtastic app](#) or web UI, open the node list and check the channel utilisation percentage. If it is above 25% and you suspect a specific node is the primary contributor, inspect each candidate node's reported airtime/channel-utilisation telemetry, or watch the serial/debug log of the suspect node to see how often it is transmitting and rebroadcasting. (Traceroute shows the routing path/hops to a destination - it does not measure which node rebroadcasts most frequently, so it is not the right tool for isolating a chatty node.) You can also check the MQTT stream for an unusually high publication rate from one node ID.

```
# On the suspect node: check current telemetry broadcast interval
meshtastic --get telemetry.device_update_interval
meshtastic --get telemetry.environment_update_interval

# Check position broadcast interval
meshtastic --get position.position_broadcast_secs
```

Common causes and fixes

Short telemetry intervals: If device_update_interval has been shortened well below the default, a node generates a telemetry packet every interval plus the corresponding rebroadcasts. (Note: a connected client app always receives device metrics once per minute regardless of this setting - that local behaviour is separate from the over-mesh broadcast interval, whose default is 1800 seconds / 30 minutes.) Set it back to 30 minutes for infrastructure nodes:

```
meshtastic --set telemetry.device_update_interval 1800
meshtastic --set telemetry.environment_update_interval 1800
```

Short position broadcast interval: A fixed repeater broadcasting position every 5 minutes generates 288 position packets per day. Increase to 24 hours:

```
meshtastic --set position.position_broadcast_secs 86400
```

Smart broadcast enabled on a non-moving node: Smart position broadcasting transmits whenever the node moves more than a configured distance. On a fixed node, GPS jitter can cause false movement detection and trigger frequent transmissions. Disable it:

```
meshtastic --set position.position_broadcast_smart_enabled false
```

Symptom 2: Node Appearing Twice in the Node List

Diagnosis

Two entries in the node list with the same name or similar names, but different node IDs, indicates that the same physical device has been flashed or configured with two different node identities. This typically happens when:

- A backup config from device A was imported onto device B, causing B to broadcast device A's NodeInfo until B's radio generates its own unique ID advertisement.
- A node was factory-reset, generating a new node ID, but other mesh nodes still have the old ID cached.
- Two devices were cloned by copying the flash image directly (not supported - never duplicate node configurations this way).

Remediation

If two physical devices have the same node ID (the critical case - never do this): One device must be factory-reset to generate a new unique ID. Factory reset clears all configuration including channel keys - re-configure from scratch:

Warning: A factory reset erases the channel PSK and all configuration - the node will fall off your mesh and emergency channel until it is manually re-keyed, which on a remote node means a site visit. NEVER factory-reset a sole-path or remote infrastructure node without (a) a current exported config backup (`meshtastic --export-config > config.yaml`) and (b) physical or admin access to restore it. Do not perform this during an active incident.

```
meshtastic --factory-reset
```

If stale entries persist after a legitimate device reset: Other mesh nodes cache node entries and expire stale ones on their own over time - there is no fixed published TTL, but the entry will age out without intervention. There is no remote command to force-clear another node's cache. Note that `--remove-position` only unsets a device's own fixed position (latitude/longitude/altitude) - it does NOT clear the node database. To reset a node's own NodeDB requires either the dedicated NodeDB-reset command or a factory reset; clearing a node's own DB does not remove a stale entry that other nodes are caching, which simply expires on its own.

```
# Clears only this device's own fixed position - NOT the node database:
meshtastic --remove-position
```

Symptom 3: Node Offline Despite Power Being Present

Diagnosis

The node is powered (LEDs lit, no alarm on power supply) but is not appearing in the mesh and is not publishing to MQTT. This indicates a firmware hang, a radio module fault, or a configuration issue that prevents the radio from transmitting.

Step 1: Attempt remote recovery

Remote administration over `--dest` supports only the `--set` and `--get` commands - there is no supported remote `--reboot` command, so a hung node generally cannot be rebooted over the air. You can remotely read or adjust settings if the radio stack is still responding:

```
# Remote admin supports only --get / --set over --dest:
meshtastic --dest '!abcd1234' --get device.role
```

If the radio stack is functional but the application layer is hung, remote reads may still respond. If the node is fully unresponsive, you will need a physical power cycle or serial console access (below). Meshtastic has no configurable hardware-watchdog setting to enable.

Step 2: Serial console inspection

Connect via USB and open a serial terminal at 115200 baud:

```
screen /dev/ttyUSB0 115200
# or on macOS:
screen /dev/cu.usbserial-0001 115200
# or using meshtastic CLI:
meshtastic --port /dev/ttyUSB0 --debug
```

Look for these patterns in the serial output (exact log wording varies by firmware version):

- `assert` or `Guru Meditation Error` - firmware crash, followed by register dump. Note the crash address for reporting to the Meshtastic GitHub issues.
- Messages indicating no packets are being received - the radio module may be locked up. Try power cycling.
- `nvs_flash` errors - NVS (non-volatile storage) corruption. Full flash erase followed by re-flash and reconfiguration is required.
- Looping reboot messages without the firmware reaching a completed-boot state - this can indicate a hardware fault; one possible cause is a damaged SPI bus between the ESP32 and the LoRa module.

Step 3: Full power cycle

Disconnect power completely (including USB) for 10 seconds. Many transient LoRa module hang states clear only with a full power cycle, not a software reset.

Symptom 4: Node Not Relaying Packets

Diagnosis

The node appears in the node list and is visible on the map, but traceroutes confirm it is not serving as a relay hop - packets that should transit through it are not being forwarded.

Check the device role

A node configured as `CLIENT_MUTE` does not relay packets from other nodes. A plain `CLIENT` node DOES relay - it rebroadcasts packets when no other node has already done so - so a `CLIENT` role is not, by itself, the reason a node fails to forward. Confirm the role:

```
meshtastic --get device.role
```

If the role returns `CLIENT_MUTE` (which suppresses rebroadcasting), change it. For a well-placed stationary infrastructure node, `ROUTER` is the recommended role:

```
meshtastic --set device.role ROUTER
```

Check the rebroadcast mode

```
meshtastic --get device.rebroadcast_mode
```

The rebroadcast mode is set under the `device.` namespace (`device.rebroadcast_mode`). The valid values are:

- `ALL` - rebroadcast all packets (the default).
- `ALL_SKIP_DECODING` - same as `ALL` but skips packet decoding and simply rebroadcasts. This only takes effect on the **REPEATER** role; it is not a general relay mode for other roles.
- `LOCAL_ONLY` - ignore packets from foreign meshes (other regions/modem presets), relaying only local-mesh traffic.
- `KNOWN_ONLY` - relay only packets from nodes already in the NodeDB, filtering out unknown/foreign nodes.
- `NONE` - do not rebroadcast (only valid for `SENSOR/TRACKER/TAK_TRACKER` roles).
- `CORE_PORTNUMS_ONLY` - rebroadcast only core protocol port numbers.

A mode of `LOCAL_ONLY` or `KNOWN_ONLY` filters foreign-mesh traffic and may explain why some transit traffic is being dropped.

Check hop count on arriving packets

If packets arrive at this node with a remaining hop count of 0, they are consumed and not forwarded - this is correct behaviour, not a fault. Verify by enabling debug output and watching the `hop_limit` field in received packets:

```
meshtastic --debug 2>&1 | grep "hop_limit"
```

If all arriving packets have `hop_limit = 0`, the originating nodes may need their `lora.hop_limit` increased (default 3, maximum 7). Raising the hop limit increases airtime and can worsen congestion, so do this cautiously and watch channel utilisation.

Check channel configuration

A relay forwards packets based on the unencrypted packet header (sender NodeID + packet ID), and it deduplicates using that same header - so a node can and does rebroadcast packets even when it does not hold the channel PSK. This is exactly why `ALL_SKIP_DECODING` can relay channels it cannot decrypt. A PSK mismatch prevents reading the message content, not relaying it. To actually exchange and read messages with the nodes you are serving, the channel 0 PSK must match:

```
meshtastic --info | grep -A 5 "channel_0"
```

Serial Console Diagnostic Reference

When physically on-site, the following serial console commands provide rapid triage:

```
# Full device info dump (version, channels, config, node DB)
meshtastic --info

# Export the full current configuration to a YAML backup file
meshtastic --export-config > config.yaml

# Watch live packet events (requires --debug flag)
meshtastic --debug

# Factory reset (last resort - wipes config and channel keys; back up first)
meshtastic --factory-reset
```

Reading the debug log for relay events

When `--debug` is active, relay events appear in the log. The exact wording varies by firmware version, but conceptually you will see rebroadcast events (the node forwarding a packet with a decremented hop limit) and duplicate-suppression events (the node ignoring a packet it has already heard), for example:

(illustrative, not verbatim)

```
Rebroadcasting packet from !abcd1234 ...
```

```
Ignoring duplicate packet from !abcd1234 (id=0x12345678)
```

If you see only duplicate-suppression lines and no rebroadcast lines, the node is receiving packets but treating them all as duplicates. This can happen if the duplicate detection window (stored in RAM) has stale entries from a previous session - a reboot typically clears this.

If neither message type appears for packets that other nearby nodes are hearing, the LoRa receive path is not functioning. Check antenna connection (a disconnected antenna is the single most common field repair) and SPI bus integrity.

Revision #6

Created 2026-05-03 05:50:08 UTC by Mesh America Admin

Updated 2026-06-09 00:23:13 UTC by Mesh America Admin