Common Network Problems and Solutions

Meshtastic networks are remarkably self-organizing, but they are not self-diagnosing. When the mesh stops working well - messages drop, nodes disappear, delivery becomes unreliable - there are a small number of root causes that account for the vast majority of problems. This page covers the four most common issues: high channel utilization, ghost nodes, routing loops, and clock skew, along with specific remediation steps for each.

Problem 1: High Channel Utilization

Symptoms

- Channel utilization percentage consistently above 25%.
- Messages sometimes fail to deliver even between nearby nodes.
- The node list shows many nodes as "last heard" minutes ago even though they are known to be online.
- ACKs are not received for sent messages despite short distances.

Root Cause

The main drivers of channel utilization are **position, nodeinfo, and telemetry broadcasts**. Every node broadcasts its GPS position on a smart interval (minimum interval) and an update interval, alongside periodic NodeInfo and telemetry. On a large mesh with 30+ nodes all broadcasting position every 15 minutes, the channel quickly fills. Each position packet is ~50 bytes; at LongFast (SF11/BW250/CR4-5) this is roughly 400 - 700 ms of airtime (use a LoRa time-on-air calculator for the exact value at your payload size). With 30 nodes broadcasting every 900 seconds, position alone consumes on the order of 30 * 0.5 / 900 ≈ 1.7% airtime continuously for a single transmission each - and that understates the real load, because each broadcast is rebroadcast by multiple nodes across the mesh, and additional overhead (ACKs, route discovery, telemetry) multiplies the effective utilization.

Solutions

1. **Increase minimum and maximum position broadcast intervals.**

   In the Meshtastic app: Settings → Radio Configuration → Position → *Broadcast Interval*. For a community mesh where nodes are mostly stationary, intervals of 30 - 60 minutes are appropriate. Mobile nodes can use smart position which triggers a broadcast only when the node has moved more than a threshold distance.

   ```
   meshtastic --set position.position_broadcast_secs 3600
   meshtastic --set position.position_broadcast_smart_enabled true
   meshtastic --set position.gps_update_interval 120
   ```

2. **Disable unnecessary telemetry modules.**
   Environment telemetry (temperature, humidity) and power telemetry broadcast at their own intervals. If sensors are not connected, disable the modules to stop unnecessary transmissions. Note: setting an update interval to 0 does *not* disable telemetry - a value of 0 reverts to the 30-minute (1800s) default. To actually stop the broadcasts, disable the measurement:

   ```
   meshtastic --set telemetry.environment_measurement_enabled false
   meshtastic --set telemetry.power_measurement_enabled false
   ```

3. **Reduce hop limit on client nodes.**
   Most messages on a well-designed mesh reach their destination in 1 - 2 hops. Reducing the hop limit lowers the number of times a packet is relayed across the mesh, reducing airtime. Tune carefully: setting it too low can cut off nodes that genuinely need 3 hops to reach the rest of the network.

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

4. **Switch to a lower-duty-cycle modem preset for dense networks.**
   The MediumFast preset (SF9/BW250) offers significantly shorter air time per packet at the cost of reduced range. For a dense urban mesh where all nodes are within 2 - 3 km of a relay, this trade-off is often worthwhile.

Problem 2: Ghost Nodes

Symptoms

- The node list contains nodes with "Last Heard" timestamps hours or days in the past that never update.
- The node count shown in the app is higher than the number of known active devices.

- Sending a message to a ghost node never gets ACKed.

Root Cause

Meshtastic stores a local database of every node it has ever heard. Nodes that leave the area, run out of battery, or are turned off permanently remain in the database indefinitely until manually cleared or the database fills and the oldest entry is evicted. These stale entries are "ghost nodes."

Ghost nodes are not just a cosmetic issue. On a network with strict channel utilization budgets, any mechanism that sends directed packets to ghost nodes (e.g., automated pings or position requests) wastes airtime. Additionally, some firmware versions attempt to route through recently-known paths, which can cause messages to fail if those paths included a now-offline ghost.

Solutions

1. **Remove ghost nodes via the CLI:**

   ```
   # Remove a single node by its node ID. Replace !deadbeef with the
   # target node's actual ID (the !-prefixed hex shown in the node list):
   meshtastic --remove-node !deadbeef

   # Clear the entire node database (nuclear option)
   meshtastic --reset-nodedb
   ```

   Caution: `--reset-nodedb` removes all known nodes including active ones. The mesh will rebuild the database over the next few hours as nodes broadcast again. Only use this when the database is severely polluted.
2. **Remove ghost nodes via the app:**
   In the Android app, long-press a node in the node list and select *Remove from node list*. iOS uses a swipe-left gesture on the node row.
3. **Note on node-info broadcast cadence:**
   The `device.node_info_broadcast_secs` setting controls how often *this* node broadcasts its own NodeInfo (default 10800s / 3 hours, minimum 3600s). It does *not* configure automatic eviction of other nodes - there is no documented "not heard within this window, auto-evict" setting. Remove stale entries manually via the app or `--remove-node` as above.

   ```
   # Adjust how often THIS node advertises its NodeInfo (does not evict others):
   meshtastic --set device.node_info_broadcast_secs 10800
   ```

Problem 3: Routing Loops

Symptoms

- Channel utilization spikes suddenly and stays high.
- The `--listen` output shows the same packet ID appearing many times from many different node IDs in rapid succession.
- A high proportion of received packets are duplicates of ones already seen.
- Battery drain on relay nodes accelerates noticeably.

Root Cause

Meshtastic uses managed flooding: before rebroadcasting, a node listens briefly to see whether another node has already rebroadcast the packet, and it skips packets it has recently seen (tracked in a packet history). Each hop also decrements the hop limit, which bounds how long a packet can live. Apparent "loops" - the same packet circulating repeatedly - usually trace back to:

- **Congestion / collisions**: when the channel is busy, a node may not hear the earlier rebroadcast it would otherwise have suppressed against, so it rebroadcasts anyway.
- **Misconfigured infrastructure roles**: too many always-rebroadcasting nodes in one area increases redundant transmissions.
- **Firmware bug**: certain firmware versions had rebroadcast-suppression bugs that caused excess relaying. Updating firmware is the primary fix.

Solutions

1. **Update to the latest stable firmware.**
   Many loop-related bugs have been fixed in 2.3+ firmware. Check github.com/meshtastic/firmware/releases for the current stable release and update all nodes.
2. **Reduce hop limit to 2 or 1 for the duration of the incident.**
   A lower hop limit reduces the number of relays that participate, helping the rebroadcast-suppression mechanism contain the excess traffic:

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

3. **Identify and temporarily disable the node that triggered the loop.**
   During a `--listen` session, the node ID that appears most frequently as the source of duplicate bursts is often the loop originator or an amplifying relay. Temporarily taking that node off the air allows the loop to die out.
4. **Do not cluster multiple infrastructure relay nodes in the same RF coverage area.**
   Infrastructure roles (ROUTER, and the deprecated REPEATER) rebroadcast with higher priority - they rebroadcast even when they hear another rebroadcast. Clustering several of

them in the same area increases collision probability and redundant relaying. Note: the older ROUTER_CLIENT role was removed in firmware 2.3.15; for infrastructure use ROUTER (or ROUTER_LATE), and for ordinary nodes use CLIENT. ROUTER nodes appear in the node list and run a full client; REPEATER nodes do not appear in the node list (and REPEATER itself was deprecated in firmware 2.7.11).

Problem 4: Clock Skew

Symptoms

- Message timestamps in the app appear wrong (hours off or showing Unix epoch 0).
- Nodes show "Last Heard" timestamps in the future.
- Position packets are ignored or treated as stale even for active nodes.

Root Cause

Meshtastic nodes use GPS time when GPS is available. Without GPS or NTP, most ESP32 and nRF52 boards have no battery-backed real-time clock, so the clock resets entirely on every power loss (it does not merely drift) and timestamps cannot be trusted until re-synced from GPS, NTP, or a connected app/PC. A node with a wrong clock produces incorrect "last heard" calculations and misleading message timestamps. (Clock skew does *not* cause packet-ID collisions - packet IDs are 32-bit identifiers, not timestamp-derived.)

Solutions

1. **Enable GPS time synchronization.**
   Nodes with GPS hardware will sync their RTC from GPS time lock. Ensure GPS is enabled and the device has a clear sky view for at least 5 minutes after power-on to acquire a time fix.
2. **Use NTP time synchronization via Wi-Fi.**
   ESP32 nodes connected to Wi-Fi can sync time via NTP. This happens automatically when Wi-Fi is connected. Ensuring your gateway node has working Wi-Fi keeps its clock accurate and propagates timestamps to the mesh through its packets.
3. **Check and correct time via the Python CLI:**

```
import meshtastic
import meshtastic.serial_interface
import time
```

```
iface = meshtastic.serial_interface.SerialInterface()
# The library sets the RTC on connect if the local system clock is accurate
# You can also check the current node time:
info = iface.getMyNodeInfo()
print("Node time:", info.get("localConfig", {}).get("device",
{}).get("nodeInfoBroadcastSecs"))
iface.close()
```

Simply connecting with the Python library sets the node's time to the host computer's system clock, which is a quick fix for a clock-skewed node when you have USB access.

4. **For nodes without GPS or Wi-Fi: use the Meshtastic app to sync time on connect.**
   The mobile app automatically sends the phone's current time to the node when it connects over BLE. Briefly connecting the app to a clock-skewed node is often the easiest fix in the field.

Quick Reference: Problem-to-Solution Matrix

Symptom	Most Likely Cause	First Action
Channel utilization > 25%	Frequent position, nodeinfo, and telemetry broadcasts (plus message traffic and hop count)	Increase position broadcast interval to 1800 - 3600s; trim telemetry
Nodes in list never heard	Ghost nodes in DB	Remove stale nodes via app or `--remove-node`
Same packet seen 10+ times	Excess rebroadcasting (congestion or misconfigured roles)	Update firmware; reduce hop_limit to 2
Timestamps wrong / future dates	Clock skew (no GPS/NTP)	Connect app or CLI to sync time from phone/PC
ACKs missing, good SNR	Congested channel saturating ACK window	Reduce channel utilization; check for loops

---

Revision #7
Created 2026-05-03 05:39:38 UTC by Mesh America Admin
Updated 2026-06-09 01:59:50 UTC by Mesh America Admin