Designing for Reliability: N+1 Redundancy

Designing for Reliability: N+1 Redundancy

In plain terms: make sure that no single node failure can split your network into two halves that can no longer reach each other. A mesh network is only as reliable as its weakest single point of failure. In graph theory terms, a node whose removal splits a connected graph into two or more disconnected components is called a cut vertex. Real-world Meshtastic deployments often develop cut vertices without operators realizing it - especially as networks grow organically. N+1 redundancy means ensuring that for every critical backbone node, at least one backup path exists so that the loss of any single node does not partition the network.

Important caveat: a second node at the SAME site protects only against single-device failure (firmware crash, board death). It does NOT protect against the failures most likely in an emergency - site power loss, lightning, tower collapse, fire, flooding - because both nodes share that fate. For incident-grade redundancy, the backup MUST be at a separate site on independent power.

Identifying Critical Backbone Nodes

Start by drawing your network on paper or a whiteboard. Place each node as a dot and draw lines between nodes that can hear each other directly. Now ask: if I erase this dot, does the network split into two disconnected groups? Any node where the answer is yes is a cut vertex and a single point of failure.

For larger networks, the most reliable analysis is manual: run `meshtastic --traceroute` between nodes on opposite sides of a suspect backbone node - if the only path routes through that one node, it is critical. Third-party visualizers such as meshmap.net can help you eyeball topology, but they only show nodes that report to the public MQTT server (a gateway is required) and are not authoritative; do not rely on them as a complete picture of your mesh.

Providing Backup Paths

Once critical nodes are identified, the fix is straightforward in concept: ensure each one has a backup. Two approaches work well in practice:

- **Second node at the same site:** Place a second ROUTER node at the same location as the critical node. Both nodes cover the same area, so if one fails at the device level, the other continues forwarding. This costs hardware but is simple to maintain. Be aware of two limits: (1) two co-located relays both rebroadcasting add airtime and can raise channel utilization in that area - prefer one active ROUTER plus a standby, or stagger roles, and reconcile this against the channel-utilization guidance; and (2) as noted above, a same-site pair does NOT protect against site-level failures.
- **Alternate mesh path:** Add a node at a different location that bridges the same two network segments. This is more work to plan but is far more robust - it protects against site-level failures (power outage, lightning, physical damage) rather than just single-node failures. This is the approach required for incident-grade redundancy.

Testing Redundancy

Testing is non-negotiable. Make sure no single node failure splits your network: a backup path that exists on paper but has never been verified may fail in practice due to marginal signal, wrong node roles, or misconfigured hop limits. The test procedure is simple:

1. Identify two nodes on opposite sides of the critical backbone node you want to test.
2. Run a traceroute between them and record the path.
3. Briefly power the backbone node down to simulate failure. (Power it off - do NOT simply unplug its antenna while it is transmitting; transmitting into a disconnected antenna can damage the radio.)
4. Run the traceroute again. The path should route around the powered-down node.
5. If routing fails, the backup path is insufficient and must be improved before the critical node is trusted.

A passing reroute test proves the backup path *exists*, not that it has *capacity*. Under incident load the backup path carries the failed node's traffic on top of its own and may saturate. Verify the backup path also has channel-utilization headroom for surge traffic, not just connectivity.

Test redundancy at least quarterly, and ALWAYS as part of pre-incident readiness checks or before any planned event you expect to rely on the network. Annual testing is insufficient for networks used in emergencies - paths degrade silently between tests, and any time the physical environment changes significantly you should retest.

Network Mapping Tools

Third-party visualizers such as meshmap.net provide a visual overlay of node positions for nodes that report to the public MQTT server, which can help you spot obvious topological bottlenecks - but they are not authoritative and will not show nodes that lack an internet gateway. The ground truth for redundancy verification is the `meshtastic --traceroute <destination_id>` CLI command, which reveals the actual hop-by-hop path packets take in real time.

Revision #3
Created 2026-05-03 06:58:52 UTC by Mesh America Admin
Updated 2026-06-09 00:27:37 UTC by Mesh America Admin