Meshtastic Repeater Network Patterns

- Designing for Reliability: N+1 Redundancy
- Planned Maintenance Procedures for Live Networks
- Channel Utilization Management

Designing for Reliability: N+1 Redundancy

Designing for Reliability: N+1 Redundancy

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.

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 visual tool meshmap.net can import your mesh topology and highlight connectivity. Alternatively, 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.

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 or REPEATER node at the same location as the critical node. Both nodes cover the same area, so if one fails, the other continues forwarding. This costs hardware but is simple to maintain.
- **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 more geographically robust - it protects against site-level failures (power outage, physical damage) rather than just single-node failures.

Testing Redundancy

Testing is non-negotiable. 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 off the backbone node (or unplug its antenna to simulate failure).
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.

Schedule this test annually, or any time the physical environment changes significantly.

Network Mapping Tools

meshmap.net provides a visual overlay of node positions and can render connectivity based on neighbor data exported from your mesh. Use it to spot topological bottlenecks before they become outage events. The `meshtastic --traceroute <destination_id>` CLI command reveals the actual hop-by-hop path packets take in real time, which is the ground truth for redundancy verification.

Planned Maintenance Procedures for Live Networks

Planned Maintenance Procedures for Live Networks

Taking a backbone node offline for maintenance - whether for firmware updates, hardware replacement, or antenna adjustments - affects the users routing through it. With proper planning, that impact can be reduced to a brief interruption rather than a prolonged outage. This page describes a repeatable maintenance procedure for Meshtastic backbone nodes in active community networks.

Pre-Maintenance Checklist

Complete these steps before any planned maintenance window:

- **Notify the community:** Post advance notice (24-48 hours) in your community communication channels - Discord, Signal group, or whatever your community uses. Include the node name, scheduled time, and estimated duration. Users who depend on that backbone link can plan accordingly.
- **Verify backup paths:** Run traceroutes from nodes on either side of the target to confirm alternate routing exists. If no backup path is available, consider deploying a temporary node before taking the backbone node offline.
- **Schedule during low-traffic hours:** 2-5 AM local time is typically the quietest window for community mesh networks. Emergency networks may have different quiet windows - check your message logs to identify the lowest-traffic period.
- **Document current configuration:** If replacing hardware, record all node settings (node name, channel configuration, role, hop limit) so the replacement can be configured identically before going live.

During Maintenance

Where possible, power down gracefully rather than performing a hard power-off. A graceful shutdown allows the node to stop transmitting cleanly and reduces the chance of a neighbor node entering a routing loop while searching for the missing node. For solar-powered nodes, the practical approach is to disconnect the load output of the charge controller rather than physically unplugging the node in the dark.

Perform the maintenance task - firmware flash, hardware swap, antenna replacement - as quickly as practical. Every additional minute of downtime increases the chance of a user encountering a failed message delivery.

Post-Maintenance Verification

Before declaring the node returned to service, verify it from multiple directions:

1. Send a test message from a node that previously routed through the maintained node and confirm delivery.
2. Run traceroutes from multiple directions to confirm the node is routing normally.
3. Confirm the room server sees the node as online and shows recent activity.
4. Update your network maintenance log with the date, work performed, and any configuration changes.

Emergency Rollback Procedure

If a replacement node does not work as expected - wrong firmware, hardware fault, or configuration error - act quickly. Restore the original node if it is still functional, or connect a known-good spare configured to match the original settings. If neither is possible, notify the community immediately that the outage is extended and provide an estimated restoration time. Keeping one spare node per critical site is strongly recommended for networks where reliability is important.

After any unplanned extended outage, conduct a brief post-incident review: what failed, why, and what process change would prevent recurrence. Even a one-paragraph note in the maintenance log is valuable for future operators.

Channel Utilization Management

Channel Utilization Management

Channel utilization (CU) is one of the most important health metrics for a Meshtastic network, yet it is frequently misunderstood or ignored until problems become severe. Understanding what CU measures, what causes it to rise, and how to bring it back down is essential knowledge for any network operator running more than a handful of nodes.

What Is Channel Utilization?

Channel utilization is the percentage of time the radio channel is occupied by transmissions, measured over a rolling window (Meshtastic uses a 15-minute window by default). A CU of 10% means the channel was actively transmitting for 90 seconds out of every 15 minutes. Meshtastic calculates CU locally on each node by monitoring the time its own radio is keyed up, plus time spent sensing channel activity from other nodes. The reported CU figure is visible in the Meshtastic app under the node detail view and in the device telemetry metrics channel.

Healthy, Warning, and Critical Thresholds

- **Under 15%:** Healthy. The channel has ample headroom for message traffic, routing overhead, and telemetry. Packet loss due to collisions is rare.
- **15-25%:** Warning zone. You may begin seeing occasional packet loss, especially during bursts of activity. Investigate the causes and plan remediation before the situation worsens.
- **Over 25%:** Critical. At this level, the probability of two nodes transmitting simultaneously and causing a collision is high enough to cause significant packet loss on a sustained basis. User-visible symptoms include messages that appear to send but are not received, slow acknowledgements, and missed position updates.

Common Causes of High Channel Utilization

Several factors drive CU up. Understanding which ones apply to your network guides the correct fix:

- **Too many ROUTER nodes in close proximity:** Each ROUTER node retransmits packets it hears, and in a dense cluster, those retransmissions pile on top of each other. Five ROUTER nodes covering the same area produce five times the retransmission traffic of

one - without providing five times the coverage benefit.
- **Hop limit too high for network geography:** A hop_limit of 5 in a network where the farthest node is only 2 hops from the origin means packets are retransmitted up to 5 times unnecessarily. Every extra retransmission is wasted airtime.
- **High message traffic:** Active communities that send many messages, combined with frequent position and telemetry updates from many nodes, can saturate even a well-configured channel.
- **Routing loops:** A misconfigured network can cause packets to cycle through a set of nodes repeatedly. This is rare but produces dramatically elevated CU when it occurs. Check for loops by examining traceroute output - if a node appears twice in the path, a loop is present.

Remediation Strategies

- **Reduce hop_limit:** Set hop_limit to the minimum value that still delivers packets to all nodes in your network. For most community deployments, three to four hops is sufficient.
- **Convert excess ROUTER nodes to CLIENT_MUTE or CLIENT:** Audit your node roles. Nodes in the same area do not all need to be ROUTERs. Designate one or two strategically placed nodes as ROUTER and set the rest to CLIENT or CLIENT_MUTE to suppress retransmissions from densely packed areas.
- **Increase channel_num spacing:** If two active networks share physical proximity, configuring them on different channel numbers reduces interference between the two populations of nodes.
- **Consider MeshCore for large networks:** Meshtastic uses a flooding approach where every ROUTER retransmits every packet. In networks with dozens of active nodes and high message traffic, this flooding behavior is the fundamental cause of high CU. MeshCore uses a different routing architecture that avoids flooding, making it a better fit for large, dense, high-traffic community networks.