Neighbor Info and Signal Mapping

What Is the Neighbor Info Module?

The Neighbor Info module is a built-in Meshtastic feature that periodically broadcasts a summary of every node your device can hear directly, along with the received signal strength (RSSI) and signal-to-noise ratio (SNR) for each. This data lets you build an objective picture of your mesh's link quality without relying on anecdotal reports.

What Neighbor Info Reports

Each Neighbor Info packet contains:

- **Node ID** of each directly-heard neighbor
- **SNR** (signal-to-noise ratio in dB) - how cleanly the signal was received
- **RSSI** (received signal strength indicator in dBm) - the raw power level
- The **timestamp** of the last reception from that neighbor

Reports are sent at a configurable interval (default: 900 seconds / 15 minutes). The data is broadcast on the mesh and visible to any node that receives it.

Enabling Neighbor Info

1. Open the Meshtastic app and connect to your node.
2. Go to **Radio Configuration → Modules → Neighbor Info**.
3. Toggle **Enabled** to on.
4. Optionally adjust the **Update Interval** (in seconds). Shorter intervals give fresher data but increase channel utilization.
5. Save and reboot the node.

Via CLI: `meshtastic --set neighbor_info.enabled true --set neighbor_info.update_interval 900`

Reading the Data

In the Meshtastic app, go to **Node List → [select a node] → Neighbor Info** to see that node's reported neighbors. In the Python CLI, listen for neighbor info packets:

```
meshtastic --listen
# Look for portnum: NEIGHBORINFO_APP packets in the JSON output
```

The JSON payload includes a `neighbors` array with `node_id`, `snr`, and `last_rx_time` fields for each neighbor.

Building a Link Quality Map

Collect neighbor info data from all nodes over a period of time to build a directional link quality graph. Each directed edge in the graph represents one node hearing another, with the SNR as the edge weight. Nodes with low average SNR to all their neighbors are candidates for antenna upgrades or repositioning.

Tools like **Meshview** or a custom Python script consuming the MQTT JSON feed can automatically visualize this as a network graph, coloring links by quality (green > 5 dB SNR, yellow -5 to 5 dB, red < -5 dB).

Exporting to CSV for Analysis

Using a simple Python MQTT subscriber:

```
import paho.mqtt.client as mqtt, json, csv, time


rows = []
def on_message(client, userdata, msg):
 d = json.loads(msg.payload)
 if d.get("type") == "neighborinfo":
 for n in d["payload"].get("neighbors", []):
 rows.append({
 "reporter": hex(d["from"]),
 "neighbor": hex(n["node_id"]),
 "snr": n["snr"],
 "time": time.strftime("%Y-%m-%d %H:%M:%S")
 })
```

```
client = mqtt.Client()
client.on_message = on_message
client.connect("localhost", 1883)
client.subscribe("msh/US/2/json/neighborinfo/#")
client.loop_forever()
```

Write `rows` to a CSV and import into a spreadsheet to sort and filter weak links.

Identifying Weak Backbone Links

Sort the exported CSV by SNR ascending. Any link with SNR consistently below -5 dB is fragile and should be treated as unreliable for backbone routing. Consider: raising antenna height, switching to a high-gain directional antenna, or adding an intermediate relay node to split the hop into two shorter, stronger links.

Revision #4
Created 2026-05-03 06:45:38 UTC by Mesh America Admin
Updated 2026-05-03 13:38:28 UTC by Mesh America Admin