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 signal-to-noise ratio (SNR) of each link. 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
- The **timestamp** of the last reception from that neighbor

Note: Neighbor Info reports **SNR only** per neighbor - the NeighborInfo payload does not carry a per-neighbor RSSI value. RSSI is available locally for each packet your own node receives, but it is not part of the broadcast neighbor report.

Reports are sent at a configurable interval (default 21600 seconds / 6 hours). The firmware enforces a minimum of 14400 seconds / 4 hours - values below this are rejected. The data is broadcast on the mesh and visible to any node that receives it.

Privacy note: enabling Neighbor Info publicly discloses which nodes your node can hear - an RF "social graph" of who-can-hear-whom. On any channel that is uplinked to MQTT, that graph is published to the internet (and can appear on third-party maps). For sensitive or EmComm deployments, consider leaving Neighbor Info disabled on nodes whose neighbor relationships are operationally sensitive.

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; minimum 14400 / 4 hours). 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 21600` (any value >= 14400; lower values are rejected by firmware).

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 each neighbor carrying its node ID and `snr`. (Confirm exact serialized field names against a live NEIGHBORINFO_APP JSON sample or the `neighbor_info.proto` before relying on them in a parser.)

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. Useful SNR thresholds are **preset-dependent**, because LoRa's decode floor varies with spreading factor (roughly -7.5 dB at SF7 up to about -20 dB at SF12; ~-17 dB on the default LongFast/SF11 preset). As a suggested visualization convention for the default LongFast preset (not an official spec): green > 0 dB, yellow -10 to 0 dB, and red only when within a few dB of the ~-17 dB decode floor.

Exporting to CSV for Analysis

Using a simple Python MQTT subscriber. Note that JSON packets are published to `msh/REGION/2/json/CHANNELNAME/USERID` - segmented by channel name and gateway user ID, not by packet type. Subscribe to the JSON tree (or a specific channel) and filter on the JSON `type` field instead of using a type-named topic:

```
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/#")  # all channels; or msh/US/2/json/LongFast/# for one
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. Treat a link as fragile when its SNR is within a few dB of the decode floor for your active modem preset - not at a fixed cutoff. On the default LongFast preset (SF11, floor ~-17 dB) a -5 dB link still has roughly 12 dB of margin and is healthy; the fragile links there are the ones worse than about -14 dB. A -5 dB cutoff is only appropriate near SF7 (floor ~-7.5 dB). For genuinely weak links, 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 #5
Created 2026-05-03 06:45:38 UTC by Mesh America Admin
Updated 2026-06-09 12:13:03 UTC by Mesh America Admin