# Building a Meshtastic Network Map

A network map shows you which nodes can hear each other, the quality of each link, and how messages actually route through your network. Building and maintaining an accurate map is essential for optimizing coverage and identifying problem areas.

# What to Map

A useful network map shows:

- Geographic position of all nodes (lat/lon from GPS or fixed position setting)
- Link quality between neighboring nodes (SNR)
- Hop counts between all node pairs
- Node roles — common roles include CLIENT, ROUTER, and ROUTER_LATE; the full set also includes CLIENT_MUTE, CLIENT_HIDDEN, TRACKER, SENSOR, TAK, TAK_TRACKER, LOST_AND_FOUND, and REPEATER (REPEATER is deprecated). Do not assume only three exist.
- Last-heard timestamps (to distinguish active vs. stale nodes)

# Using meshmap.net

meshmap.net aggregates position data from Meshtastic nodes that have position reporting and the MQTT gateway enabled. It provides a global view of the Meshtastic community:

- Zoom to your region to see local node density
- Click a node for details: ID, last heard, firmware version, battery level (if reporting telemetry)
- Use the time filter to see nodes active in the last 24 hours vs. all time

**Limitation:** Only shows nodes that uplink to the public MQTT broker (GPS/position enabled and a reachable MQTT gateway). meshmap.net is a third-party, opt-in service. Many private community networks don't use the public MQTT server.

**Privacy warning:** enabling position reporting plus MQTT to the public broker publishes your node's location to the public internet (meshmap.net and any other broker subscriber), potentially indefinitely. Only do this for nodes whose location you intend to be public. For fixed sites you don't want pinpointed, use a coarsened or deliberately offset fixed position.

# Building a Private Community Map

For a private community network not connected to the public MQTT, build your own map:

```
# Option 1: Self-hosted Meshtastic map
# Run a local MQTT broker + a self-hosted map application.
# See github.com/brianshea2/meshmap.net or github.com/liamcottle/meshtastic-map
# for self-hostable map projects (verify the repo before deploying).

# Option 2: Grafana Geomap panel with InfluxDB
# Telemetry data (positions) -> InfluxDB -> Grafana Geomap panel
# Shows all nodes with position and telemetry data

# Option 3: Simple Python script to generate KML
import json, paho.mqtt.client as mqtt

nodes = {}
def on_message(client, userdata, msg):
 # Parse position packets and build nodes dict
 # Export to KML for Google Earth or KMZ for Google Maps
 pass
```

# Link Quality Analysis

Meshtastic's Neighbor Info module reports each node's heard neighbors and the SNR of those links (it carries SNR, not per-neighbor RSSI). Enable it on key backbone nodes — but note it is airtime-intensive and is discouraged on the default public channel:

```
# Enable Neighbor Info module:
meshtastic --set neighbor_info.enabled true
# Interval cannot be set below 14400 s (4 hours); default is 21600 s (6 hours).
meshtastic --set neighbor_info.update_interval 21600  # 6 hours (default)
```

With Neighbor Info data flowing through MQTT, you can build a heat map of link quality across your network. The usable SNR floor depends on the modem preset (roughly -7.5 dB at ShortFast/SF7 down to about -20 dB at SF12; around -17 dB on the default LongFast/SF11). A fixed "-10 dB" rule of thumb only fits mid presets — on LongFast a -10 dB link still has healthy margin. Treat links within a few dB of the preset's decode floor (on LongFast, roughly worse than -14 dB) as weak candidates for repeater insertion or antenna improvement.

# Topology Visualization Tools

- **Gephi** - Open-source network visualization. Export node list and edge list from your MQTT data; import into Gephi for force-directed layout visualization of your mesh topology.
- **NetworkX (Python)** - Build a graph of your mesh programmatically; use matplotlib for visualization; calculate graph properties (diameter, clustering coefficient, cut vertices).
- **MeshView** - Community tool that parses Meshtastic position and neighbor data to produce a topology map. Check GitHub for current maintained versions.

---

Revision #3
Created 2026-05-03 10:48:07 UTC by Mesh America Admin
Updated 2026-06-09 01:58:48 UTC by Mesh America Admin