

How Meshtastic Works

The Meshtastic protocol, routing, and architecture explained.

- [Meshtastic Protocol Overview](#)
- [Meshtastic Flooding Mesh Protocol Explained](#)
- [Meshtastic Node IDs, Addresses, and Naming](#)

Meshtastic Protocol

Overview

Meshtastic is an open-source mesh networking project built on LoRa radio hardware. It has a large, active global community and is one of the most widely deployed LoRa mesh platforms available.

How Meshtastic routes messages

Meshtastic uses a **flooding** approach to message delivery. When a node sends a message, nearby nodes rebroadcast it, and those nodes rebroadcast to others, until the message has propagated through the reachable network. A hop limit controls how many times a message can be relayed.

This approach is simple and effective for small networks, but it is best-effort - there is no guaranteed delivery. Confirm critical messages with an explicit ACK or acknowledgement. On larger, denser networks, the rebroadcast traffic can create congestion. Meshtastic has introduced optimizations over time (such as managed flooding) to reduce unnecessary retransmissions.

Node roles

Meshtastic devices can be assigned different roles that control how they participate in the network:

- **Client** - standard personal device paired with a phone via Bluetooth
- **Client Mute** - receives but does not retransmit (reduces traffic in dense areas)
- **Router** - infrastructure node that always rebroadcasts each packet once with prioritized routing; visible in the node list. Power Saving is enabled by default (ESP32)
- **Repeater** - dedicated relay node, does not appear in node lists. Note: REPEATER was deprecated as of firmware 2.7.11; prefer ROUTER or ROUTER_LATE for new infrastructure
- **Tracker** - broadcasts GPS position frequently
- **Sensor** - designed for environmental sensor data reporting

Note: the older **Router Client** (ROUTER_CLIENT) role was deprecated in firmware 2.3.15 and is no longer a current role. For infrastructure, use ROUTER or ROUTER_LATE; for ordinary nodes, use CLIENT.

Channels and encryption

Meshtastic uses a channel-based system. Each channel has a name and a pre-shared key (PSK). Devices on the same channel with the same key can communicate. Channels are encrypted with AES-CTR; the PSK may be 16 bytes (AES-128) or 32 bytes (AES-256). The default public channel uses a well-known 1-byte key (AQ==), which is effectively AES-128, making messages readable by any Meshtastic node - it is not private. Private channels use custom keys shared only among trusted users.

Technical specifications

Parameter	Value
Frequency (US/Canada)	902-928 MHz ISM band (commonly called the "915 MHz" band); license-free under FCC Part 15 / ISSED RSS-247. Meshtastic uses defined frequency slots within the band
Modulation	LoRa (Long Range)
Channel encryption	AES-CTR; default public channel uses a 128-bit key (the public AQ== key, so it provides no confidentiality). AES-256 available when a 32-byte PSK is set
Routing	Managed flooding with hop limit
GPS support	Optional (device-dependent)
Protocol	Open source, Protobuf-based

Meshtastic Flooding Mesh Protocol Explained

Understanding how Meshtastic actually routes messages helps you make better configuration decisions and troubleshoot problems faster.

Flooding: The Core Mechanism

Meshtastic uses a **flooding broadcast** approach to message delivery. When you send a message:

1. Your node broadcasts the packet over LoRa
2. Every node that receives it checks whether it has seen this packet before (via a packet ID hash)
3. If not seen before: the node rebroadcasts the packet (with hop count decremented by 1)
4. If seen before, or if hop count = 0: the node discards the packet

This is called "flooding" because the packet spreads outward like a flood, covering every node in range up to the hop limit. Every node participates in delivery - there's no designated router.

Hop Count and Hop Limit

Every Meshtastic packet carries a hop count. When a packet is first sent, it starts at the configured `hop_limit` (default: 3). Each time a node rebroadcasts the packet, it decrements the hop count. When hop count reaches zero, nodes still receive but don't rebroadcast.

This means a `hop_limit` of 3 allows a message to be relayed by up to 3 intermediate nodes, reaching a destination up to 3 "hops" away from the source (source → relay 1 → relay 2 → relay 3/destination).

```
# Check current hop limit:
meshtastic --get lora.hop_limit

# Change hop limit (3 is default, 7 is maximum):
meshtastic --set lora.hop_limit 5
```

Duplicate Suppression

Without duplicate suppression, flooding would create infinite loops - every node would keep rebroadcasting forever. Meshtastic prevents this by:

- Maintaining a "recently seen" packet ID cache (stored in RAM)
- Each packet is identified by the sender NodeID plus a 32-bit packet ID; a node treats a (sender, packet-ID) pair it has recently seen as a duplicate
- If a node has seen a packet ID recently, it discards the duplicate without rebroadcasting

Addressing: Broadcast vs Direct

Meshtastic packets can be addressed two ways:

- **Broadcast (channel messages)** - Addressed to all nodes (0xFFFFFFFF). Every node in the network receives and can read the message.
- **Direct (DM messages)** - Addressed to a specific node ID. On firmware v2.5+, DMs use per-recipient public-key cryptography (PKC), so only the destination node can read the content. On older firmware a DM is encrypted only with the channel key, so anyone holding that channel's PSK can read it - it is not truly private one-to-one. Either way, intermediate nodes still rebroadcast the packet (they just may not be able to read it).

Why Flooding Works Well for Small Networks

For networks up to ~50-100 nodes, flooding is remarkably effective:

- No routing tables to maintain or synchronize
- Extremely simple - even new, unconfigured nodes automatically participate in delivery
- Redundant paths - if one relay node is down, another may bridge the gap
- Works well in dynamic environments where nodes join and leave frequently

Flooding's Limitations in Large Networks

As networks grow, flooding creates a "broadcast storm" problem:

- One broadcast can trigger a rebroadcast from many nodes - up to roughly one per node within hop range that does not hear an earlier rebroadcast first. Managed flooding (listen-before-rebroadcast, SNR-based contention) suppresses many of these, but in dense meshes the cumulative retransmissions can still saturate the channel.
- Channel utilization increases with each new node; at high utilization, packets collide and delivery rates drop
- A 200-node network with active users can saturate the LoRa channel, causing significant packet loss

This is why MeshCore's path-based routing scales better for large networks - but for most community networks under 100 active nodes, Meshtastic flooding works well.

Meshtastic Node IDs, Addresses, and Naming

Understanding how Meshtastic nodes are identified helps you interpret the node list, configure direct messaging, and troubleshoot network issues.

Node ID Format

Every Meshtastic node has a 32-bit node ID derived by default from its hardware MAC address. It is not guaranteed to be globally unique. The ID is displayed in several formats:

- **Decimal:** 2882343476 (rarely used in user interfaces)
- **Hex with prefix:** !abcd1234 (most common display format; the "!" prefix denotes a node ID)
- **Short ID:** The last 4 hex digits, e.g., "1234" - used as the default short name

The node ID is normally fixed (MAC-derived) and is effectively permanent for typical users, but it is not strictly immutable: it can change on certain firmware events and can be overridden in firmware or config on some boards.

Long Name and Short Name

In addition to the hardware node ID, each node has two user-configurable names:

- **Long name** - Up to 40 bytes (about 37-39 ASCII characters; fewer if multibyte/UTF-8 or emoji are used, since those count as multiple bytes). Used in the full node list. Example: "Portland-Burnside-Bridge-Repeater" or "KG7XYZ Mobile"
- **Short name** - Up to 4 characters. Displayed on the map and in compact node list views. Example: "PDX1", "K7AB"

```
# Set long name:
meshtastic --set-owner "Your Long Name Here"

# Set short name:
```

Channel Name and Hash

Meshtastic channels have both a name and a PSK (encryption key). The channel name is displayed to users; the PSK is what actually controls which nodes can communicate on the channel. Two nodes can have different channel names but the same PSK - they'll still communicate (though the name mismatch may confuse users).

The "channel hash" is a short numeric identifier derived from the channel name AND the PSK, included in packet headers to quickly identify which channel a packet belongs to without decrypting the payload. Because the channel hash and the source/destination node IDs sit in the plaintext header, a passive RF observer can tell which channel a packet belongs to and who sent and received it without the PSK - only the message content is hidden.

How Nodes Discover Each Other

Meshtastic nodes don't have a directory or registration system. Discovery happens organically:

1. Nodes periodically broadcast "NodeInfo" packets containing their node ID, name, hardware model, and (optionally) GPS position
2. Any node that receives a NodeInfo packet adds the sender to its local node database
3. NodeInfo packets propagate through the mesh via the same flooding mechanism as messages
4. Nodes that haven't been heard from in a configurable time period are marked as "stale" in the node database

This means in a well-connected mesh, a new node will typically appear in all other nodes' lists within a few minutes of powering on, even if it's multiple hops away.