Adam McQuilkin - Meshtastic Network Centralization Proposal

#meshtastic #dartmouth #cs

Summary

We are looking to introduce a new mesh packet that would allow for advanced network analysis to be performed on Meshtastic networks. This packet would contain information on the connection each node has to other nodes in the network. Some key takeaways:

1. This functionality would be disabled by default, and would require a user to manually enable the transmission of this type of packet.
2. Assuming a constant network density d, the payload size of such packets would increase on the order of $O(dn)$, **not** on the order of $O(n^2)$.
3. There is the potential for large routing improvements within networks if the routing protocol utilizes this network state data.

This is a proposal, and as such all respectful feedback is welcome!

Introduction

My name is Adam McQuilkin[1] and I've been involved with Meshtastic for just under six months. I'm on a student team designing and building a Meshtastic desktop client[2] with the goal of allowing users to effectively administer large LoRa networks. We began the project with the intended use case of emergency response in mind, hence the name, but over time we've realized that our client has utility outside of purely the emergency response space.

One of the core components of traditional network administration is being able to accurately judge the vulnerability of a network to failure, either from direct attack or by simple equipment failures. Within a mesh network, the problem space expands to include other non-traditional issues, such as ensuring network coverage within a rapidly changing network topology. While this changing topology is often an advantage of Meshtastic networks, we are looking to give users of our client informed insight into future network vulnerabilities[3]. We seek to do this through something akin to traditional network analysis, in which we run network algorithms[4] on a graph built up over time from incoming mesh packets.

The major challenge we are encountering within this project is the decentralized nature of a mesh network. By definition mesh networks are distributed, which leads to obvious problems when we attempt to centralize network information within a single client. We are able to determine which nodes are connected to the mesh network through standard location metadata packets, but we are not able to determine the strength of the connections between any two nodes in the network[5]. To collect this information, we would need to know the strength of the connections from any given node a in the network to all of its neighbors, or at least its x most recent neighbors.

Due to the distributed nature of the network and LoRa's low bandwidth, we have no guarantee that we will have the most updated data on a given node's ability to connect to any other given node in the network. As such, we are currently defining the "strength" of the connection between a node a and its neighbor b as proportional to the quality[6] Q of the most recent packet that a has received from b, and inversely proportional to the time $t_{curr} - t_{recv}$ since a has last heard from b. While it is possible that the nodes a and b will still be able to transmit at high reliability a long time after t_{recv}, the likelihood that the edge (a, b) has been severed increases logistically with $t_{curr} - t_{recv}$.

Network Analysis and Meshtastic

The core issue within the Meshtastic ecosystem is that, to the best of our knowledge, there is no way to collect this type of information on network topology other than being directly connected to each node. A method for solving this would be to create a new protobuf packet that would collect data from each node a on the last time a message M was received from any other node b in the network, as well as the SNR of M and the time M was received. This information could then be periodically transmitted into the network.

The primary issue we're considering with a potential "network state" packet would be the risk of potential $O(n^2)$ growth of the amount of data on the network. Assuming a network were to be strongly connected, every node would have $n - 1$ connections to other nodes in the network, and each of these n nodes would then place this information onto the network. This would give us a payload data size complexity of $O(n \cdot (n-1)) \equiv O(n^2)$. The notable assumption here is that each node in the network is within range of all $n - 1$ nodes in the network. This may be the case in smaller networks, but I would argue this will not be the case on larger networks.

Let's start with a theoretical smaller network ($n \leq 25$). Let's say that in this network, each node is connected to m other nodes, where $0 \leq m \leq n$ and $\frac{m}{n}$ is close to 1. In this case, let's assume each node will transmit this proposed "network state" packet once every 15 minutes. Even if $m = n$, we will get 25 new packets every 15 minutes, each containing information on 24 other nodes. Let's assume a connection between nodes takes up 12 bytes of payload (4-byte id of the connected node, 4-byte time last packet received, 4-byte last packet received snr). From this, we get the following:

$$25 \text{ packets} \cdot 24 \text{ connections per packet} \cdot 12 \text{ bytes per connection} = 7200 \text{ bytes of payload}$$

This means, in the worst case of a smaller network, that getting this information on the network would add an additional 7200 bytes per 15 minute period to the network. Note that this excludes metadata contained within mesh packets for simplicity. This would certainly be a bandwidth investment for the network, and as such this functionality would need to be completely opt-in (see Network Status Packet Transmission without User Knowledge).

In a larger network, we can make the assumption that each node would not have $m \to n$ connections, and would instead have $m << n$ connections as $n \to \infty$. Let's assume that, across the entire network, that the ratio of nodes n on the network to the number of connections m per node is approximately constant. Let's call this ratio d, where $d := \frac{m}{n}$. If we assume that d is constant, this means that the amount of data on the network becomes $O(dn)$. Excitingly, $O(dn)$ is asymptotically equivalent to $O(n)$ since d is a scalar multiple that is independent of network size n. This means, as the size of a network grows, the relative overhead of this proposed packet would be proportional to the number of nodes n on the network, not proportional to n^2.

Note that m could be artificially limited based on the memory availability on each node, meaning that d could also be kept artificially low. An example of this could be a queue of fixed size that evicts information on the oldest connection to a given node when a new connection is received. This would also serve to limit the packet overhead on the network, say to 10 connections per node. This max number of saved connections could be defined as a protobuf constant, meaning it would be standardized across all nodes in the network.

Proposed Form of Packet

To run network analysis on Meshtastic networks using this proposed network status packet, we are looking for the following data on each node connection (a, b):

- **SNR**: The signal to noise ratio of the last packet that node a has heard from node b.
- **RX Time**: The time that node a received a packet from node b. This would be in the form of seconds since unix epoch, which is standard across Meshtastic.
- **Node ID**: The unique network ID of node b. This will be matched to other packets sent over the network, such as `Position` packets.

An example of the form of this structure is shown below, given in Rust. This code could be automatically generated from the updated protobuf definitions.

```rust
pub struct NetworkStatusPacket {
        pub connections: Vec<NetworkConnection>,
}

pub struct NetworkConnection {
        pub remote_node_id: u32,
        pub last_packet_snr: u32,
        pub last_packet_rx_time: u32,
}
```

Network Security

One concern with any new packet that has the possibility of reduced network security. This is something that we've tried to be very conscious of, especially for users of Meshtastic who require heightened network security for their use cases.

Unauthorized Access of User Location Data

When working with user location data, it is very important to ensure that the data is only able to be accessed by the intended recipient of the data. In this case, the intended recipients would be individuals in the same group as the node transmitting the packet. What defines that group is up to the user, meaning key-sharing similar to channels would be necessary to make this packet useful.

This encryption could be simplified through implementing this packet as a new `Data` variant within the `MeshPacket` message, meaning that the encryption flow would not have to be recreated from scratch, and these messages could be encrypted as with any other `MeshPacket` payload.

Network Status Packet Transmission without User Knowledge

Another concern that would need to be addressed would be packets being sent without the knowledge of a given user. This would give a trivial way to track specific nodes on the network, with no way for the node's user to know. To address this, this proposed packet would need to be controlled with a config field that would default to `disabled`. This would mean that each node would need to **manually** enable the transmission of this information.

Future Expansion

Assuming this proposal is acted upon, there is an interesting potential avenue for significant routing improvements. Our understanding of the current routing protocol[7] is that it is a naive flooding algorithm in which every packet will be sent to every node in a given network, assuming a path to such a node exists. With additional network topology information available on the network, it is possible that large routing performance improvements could be made on networks that enable such functionality. For example, being able to determine which direction a given node is from a given other node would be valuable information for reducing redundant network traffic.

This routing capability could be improved via an additional configuration field, in which nodes could specify how often they will send out such network status packets. If a group is willing to use more network bandwidth for routing packets, they would be able to improve the quality of their routing protocol. Theoretically, there is some retransmission interval that would lead to a net routing performance improvement, where the increased number of packets on the network would be outweighed by the routing improvements they enable.[8]

---

1. https://www.adammcquilkin.com/ ↩
2. Built in Rust and TypeScript, https://github.com/ajmcquilkin/Meshtastic-emergency-response-client ↩
3. Mid-network disconnection, disconnection of a single operator, bandwidth-limiting edges, etc... ↩
4. Examples of algorithms include SCC detection, Global Minimum Cut detection ↩
5. This is excluding any nodes that the radios connected to our client can directly communicate with, since our client can reliably gain information on the state of such connections. ↩
6. Signal quality in this case refers to the SNR of the last packet node a has received from node b. ↩
7. This document was first drafted in January 2023. ↩
8. This interval would be highly dependent on how dense the network was, as well as how much of a performance improvement a theoretical new routing algorithm could be. ↩