Dynamic Routing and Network Monitoring for the Polywog Protocol

A Senior Project
presented to
the Faculty of the Computer Science Department
California Polytechnic State University, San Luis Obispo

In Partial Fulfillment
of the Requirements of the Degree
Bachelor of Science

by

Ryan Lawrence & Josh Pfeffer

June, 2018

Abstract

This document analyzes the implementation of dynamic routing for router nodes in a mesh network. It examines the algorithm used to populate both the minimum-distance and the routing tables. Furthermore, it describes the node configurations used for testing and applications for mesh networks.

Additionally, this document describes the implementation and testing of network monitoring in a mesh network. It discusses the steps used to forward active nodes in the network between router nodes, a gateway node, and a network monitor.

Lastly, this paper details our future plans for implementing additional features for a network monitor.

1.0 Introduction

Polywog[1], a protocol used for wireless communication with an internet data server for microcontrollers, provides networking capabilities for these devices in multi-node mesh networks. This protocol is useful because of the low cost of its components and the limited packet overhead compared to network communication that relies on TCP/IP or UDP protocols for its underlying transport process.To achieve networking capabilities with Polywog, data must be transmitted between nodes in the network. This is done through “router” devices within the network that can forward data from one node to another.

Router nodes, low-power microcontrollers used in Polywog, connect to the network using a radio adapter transceiver device. These radios are of the same transceiver type and all nodes in a given network are initialized with matching radio frequencies. Furthermore, all nodes are initialized with a unique 8-bit physical node address. Transmissions operate by specifying a source and destination node address in the packet header.

Stored within each node is a routing table for packet transmission. When a router node receives a packet, it checks the destination node address in the packet header and searches the routing table for the next node to transmit the packet to. Per our implementation, the routing tables are created dynamically by means of control packets exchanged among the nodes.

One of our goals for the project was to implement a program to display all active nodes in a mesh network. This program could be run on a web server, allowing the server to act as a network monitor, displaying information, such as the active nodes, to the user. This is useful, especially in large mesh networks, as information about the network can be reported to a central location. Due to packet size restrictions, the low-end devices used in a Polywog network cannot

[1] https://polywog.it/docs/polywog_P1_protocol_r1_4.pdf

support Internet protocols such as IP or TCP. Thus, there is a need for communication without these protocols. To do this, information is collected on the network, forwarded to a gateway node, and encapsulated in a UDP packet, allowing transmission through the Internet.

In our senior project proposal, we outlined a very clear plan as to which tasks we expected to complete by the end of winter and spring quarters: dynamic routing and a user-friendly network monitor, respectively.

2.0 Dynamic Routing

2.1 Routing Table Format

Static routing tables may be loaded into memory from the EEPROM of each router nodes. The section of EEPROM dedicated for the routing table immediately follows the Polywog network setup portion of the EEPROM in a node. The routing table contains an array of up to 20 routing table entries and begins with a 1-byte “node type” field that determines whether the device is a router or gateway node.

Dynamic routing described here allows the routing table to be determined automatically. The network is also configured to automatically adapt to nodes that are inactive, fail, or change connectivity due to mobility.

Each entry in the routing table contains three fields: indexed destination address, next hop address, and number of hops from the destination. The indexed destination address is the index in the routing table at which a router node checks for the next node to forward the packet to. Next, the next hop address field is the 8-bit physical node address of the next node to forward the packet to. Lastly, the number of hops field is the number of hops away from the source node. An entry is not required for a node’s own destination address as this information is stored in the Arduino’s memory as part of the network setup. Below is an example of a routing table that would be stored in a router node:

Indexed Destination Address	Next Hop Address	Number of Hops
0x31	0x32	1
0x39	0x41	4
...	...	...
0x45	0x38	6
0x47	0x45	1

An example routing table

2.2 Implementation

Each router node in the mesh network runs the routing program. When this program first runs, a minimum distance table with entries containing node number, distance, and age fields is initialized. In this initialization, the entries for each node number in the network are initialized with a distance of a constant that represents that the node is not in the network and an age value of 0 that represents that the node is active in the network. However, the entry with the node number of the current router node is initialized to have a distance of 0 hops away from itself. Also in the beginning of the program, the router node's minimum distance table is broadcasted to all nodes in range.

The bulk of the dynamic routing occurs in the main loop of the program. The program constantly attempts to receive packets and broadcasts entries of its own minimum distance table every 30 seconds. Control packets designated for communicating minimum distance table contents have a function code of 0x5 (MDT). If the node receives a packet with a MDT function code, then the packet's data field with minimum distance table entries is copied into a buffer to compare with the entries in the current node's minimum distance table. For each entry in the buffer, if the received distance is less than that in the current node's minimum distance table, then the value is updated in the current node's minimum distance table. Also, for each entry in the buffer, the respective node's age is set to 0, showing that the node is currently active in the network.

At the same time that the current node's minimum distance table is updated for the entry of a node, the current node's routing table is also updated. Each routing table entry contains the indexed destination address, the next hop's address, and the number of hops to the destination. The next hop field in the routing table for the node number of the updated entry is set to the source address of the node that triggered the update of the minimum distance table. Furthermore, the number of hops to the destination field is set to the updated minimum distance.

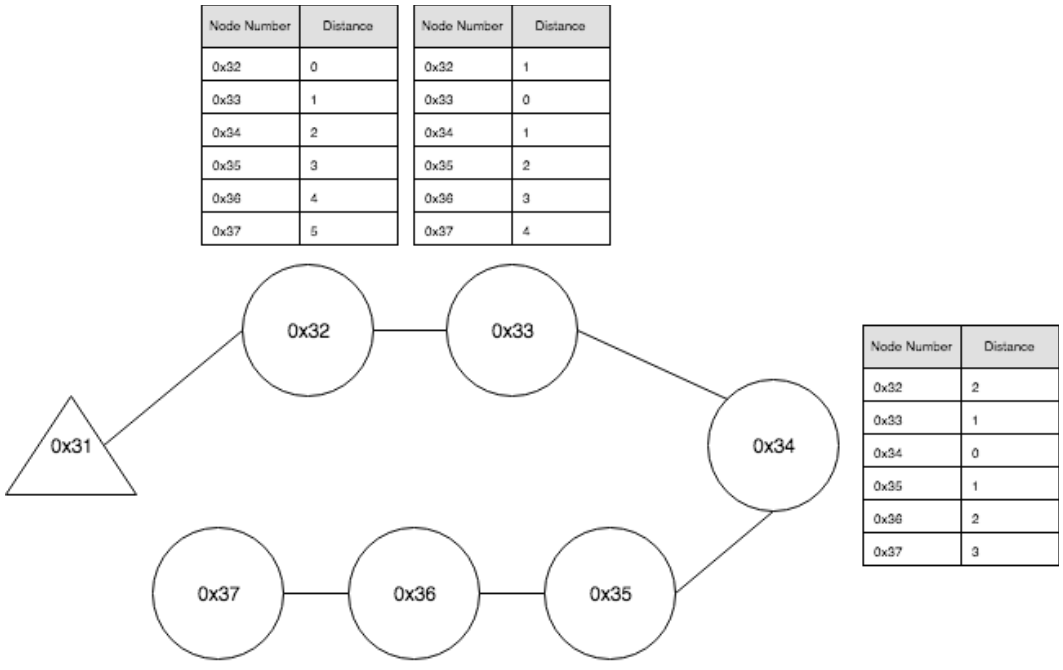
After the received packet is processed, the age values of all active nodes in the minimum distance table are incremented. If a node's age reaches the maximum age value, 15 in our case, then that node is removed from the minimum distance table. This handles the case where a node is removed from the network as it's age value will not be reset to 0 in the minimum distance tables of other router nodes. The removed node will eventually reach the maximum age value and be removed.

2.3 Testing

To validate that the dynamic routing was implemented correctly, we tested dynamic routing table creation in various configurations. To determine the success of dynamically creating routing table in our configurations, we looked at the minimum distance tables in each router node after multiple broadcasts. The routing tables are generated from the minimum distance tables so at this phase in our dynamic routing implementation, we looked at the correctness of

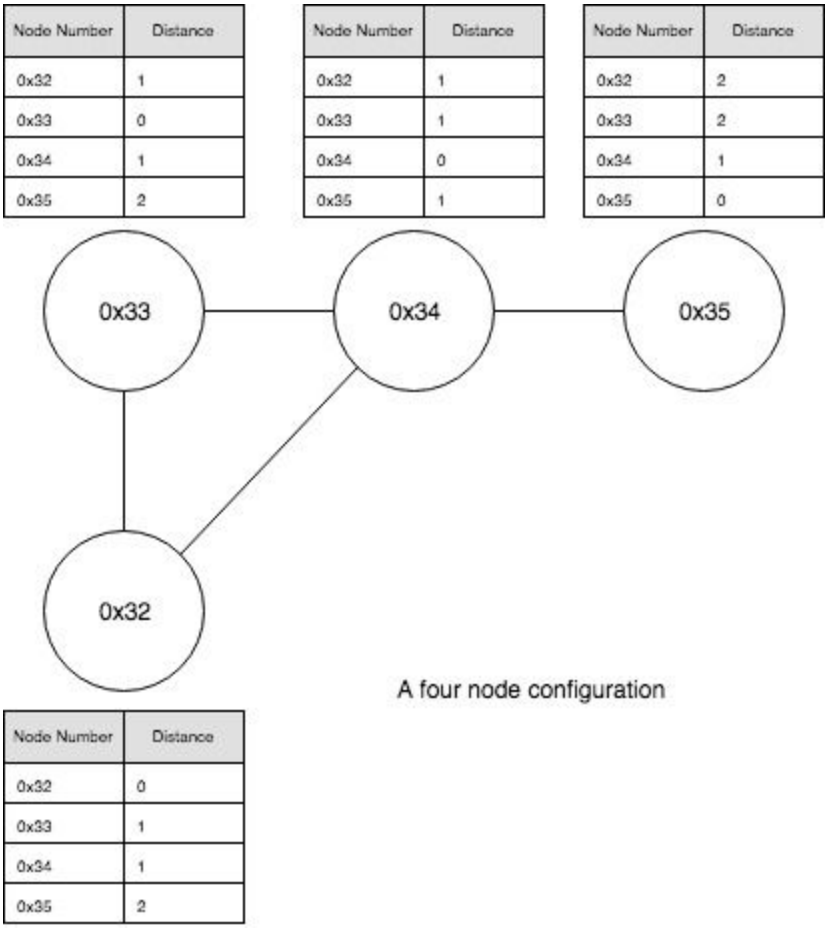
the minimum distance tables. In our configuration diagrams, triangles represent terminal nodes, circles represent router nodes, and the tables by each node represent their minimum distance table.

The first configuration we test was with six router nodes and one terminal node. All of the router nodes were placed far enough away from each other so that they could broadcast their minimum distance tables to a maximum of two other nodes. This test was done to verify that the minimum distance tables are properly updated when broadcasting over a chain of multiple router nodes. Additionally, we checked that when nodes were removed from the network, they were removed from the minimum distance tables. Below is a diagram of the configuration:



A seven node configuration

Another example we tested was a configuration of four router nodes. In this configuration, three of the nodes were in range of each other and one node in range of one of the other three nodes. This test was conducted to verify that entries in the minimum distance tables are updated when a smaller distance value is received, as will be the case with the three nodes in range of each other. This is shown below:



Improving upon the tedious static routing process, our dynamic routing algorithm identifies the best routing table based on node proximity, handles node removal, and automates the setup process.

3.0 Network Monitoring

3.1 Implementation

The first step in implementing a network monitor was to have each router node send its minimum distance table to a gateway node. Because the router nodes use dynamic routing, the nodes contained in their minimum distance tables represent the router nodes that are active in the network. In addition to router nodes broadcasting their minimum distance tables to other nodes at a fixed interval, we modified the routing program to also broadcast minimum distance table nodes to a gateway node at a fixed interval.

To broadcast nodes in a router node's minimum distance table, we first created a Polywog packet. The header of this packet includes the gateway node's address as the destination address and 0xFF as the next node address so that the packet will be received by all nodes in range. Next, we iterate through the router node's minimum distance table and add all nodes that are active to the Polywog packet's data field. Because the data field is 26 bytes and we only send one packet to the gateway node, we are currently limited to sending 26 active nodes.

A gateway node is a device that assembles network packet fragments into full packets. These full packets are encapsulated into a UDP packet and forwarded to an IP v4 or IP v6 network. To ensure end-to-end reliability, the Polywog protocol includes end-to-end acknowledgement between the destination and source nodes. In the case of connectionless operations, end-to-end acknowledgement is provided by a raw packet response. Gateway nodes may be assigned multiple destination addresses. One of these destination addresses is the gateway itself, while the others are placed in a gateway routing table, mapping destination addresses to Internet or other Network Domain addresses. In the case of our network monitor, the Polywog packets with minimum distance table nodes are sent from routing nodes to the gateway node, encapsulated into UDP packets, and forwarded to the network monitor.

The last step for us was to create a network monitor display, which is console-based and written in C. In the program, we first create a server socket. Then, we assign the port on which the server will listen -- 4955 in our case. Next, we bound the name and address to the socket. Then, the program enters an infinite loop in which the following logic is performed:

- Once a packet is received, it is read into a buffer
    - Note: the various fields of this buffer can be accessed by casting it to a GW_PACKET pointer
- The following details of the received packet are displayed:
    - Network number
    - Source address
    - Destination address
- The data portion of the Polywog packet is extracted
    - The numbers of the active nodes in the network are displayed in hexadecimal

3.2 Testing

To validate the effectiveness and accuracy of our network monitor, we tested it using the same multi-node network configurations shown in the dynamic routing tests above as well as smaller configurations. The first step was to set up a Raspberry Pi that could act as our network monitor. The Raspberry Pi was connected to our local network with a static IP address where UDP packets could be forwarded to from a gateway node. The next step was to compile the aforementioned C program and run the generated executable on our network monitor. Upon running the executable, we were ready to see which packets the gateway was receiving.

We started out with a single-node network. Once we validated that the gateway was identifying this as the only node in the network, we added another node into the network. Due to our implementation of dynamic routing, both nodes' routing tables and minimum distance tables were automatically updated once this second router node was added. Eventually, our gateway program, running on the Raspberry Pi, displayed on the console that both nodes were active in the network. We repeated this same node-addition process a few more times, validating accuracy each time.

The next step was to perform the opposite process by removing nodes from the network. Gradually, we would power off nodes one-by-one. This, in turn, updated the age field in each active node's MDT. Once the maximum allowed age was reached, the removed-node was eliminated from all of the MDTs, which thus was reflected in the gateway program.

4.0 Conclusions and Next Steps

Per our implementation, the routing table is not populated during an initialization process, but rather is filled in dynamically. Ultimately, dynamic routing will allow for a greater ease of deployment, network flexibility, and installation of mobile sensors on vehicles or animals in a localized area.

In our proposal, we defined success by a functional dynamic routing implementation. This involved the design of the routing scheme and packet format, including inclusion of a routing table "age" for detection of node failures. Furthermore, we defined spring quarter's objectives as the addition of a network monitor. Ultimately, we achieved both of these goals in our final implementation.

Currently, there are a few limitations to our final implementation. At the moment, our gateway is only able to display which router nodes are active in the network. Additional development shall allow for sensor node detection on the gateway end as well. Furthermore, because the data portion of the Polywog packet is only 26 bytes, router nodes sending their MDTs to the gateway can only fit 26 node numbers. In many practical use cases of the Polywog protocol, an extension to this feature will need to be made to allow for larger networks.