Dynamic Rerouting in a Mesh Network Due to Node Failure

Kurt Andersen, Alex Hansen, Joshua Volkmar
CSE Undergraduates, University of Nevada, Reno
CPE 400

Abstract: In modern times mesh-networks are not used the most often. There are many ways that mesh-networks can be deprecated in their efficiency. One of the ways is when a node within the network fails and can not receive or pass data packets. This paper explores an idea and simulates how to bypass the above issue within a mesh-network. By being able to avoid failing nodes within a mesh-network, it provides more integrity for the network overall and ensures packet deliveries.

Introduction: Given the issue of nodes failing within a mesh-network, this paper explores one such idea to work around and maintain network integrity. This problem is only expected to worsen in the next several years due to an increase in networked device density [1]. The solution proposed in this paper is somewhat computationally heavy on a large scaled network, but reliably maintains the network. The proposed solution is for each node in the network to know the layout of the mesh. Given this information, each node will be able to calculate an optimal path from itself to the destination node. On top of the optimal path, each node will also calculate numerous sub-optimal paths in order to maintain a durable network in the case of node failure.

This paper will discuss how the network sends out an initial message so the entire network knows a message will be sent from one node to another. The mesh network is setup such that each node in the mesh is aware of the layout of the network. Each node in the network will then calculate its optimal and suboptimal paths to the destination node. Each of the paths have a different priority based off of the weights of the given route. The simulation checks for nodes that have failed every time the current message arrives. The simulation will then reroute the message to a sub-optimal path if needed.

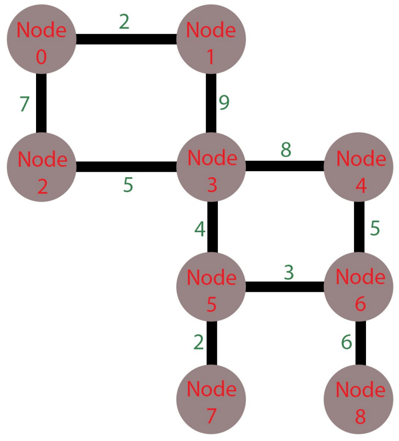
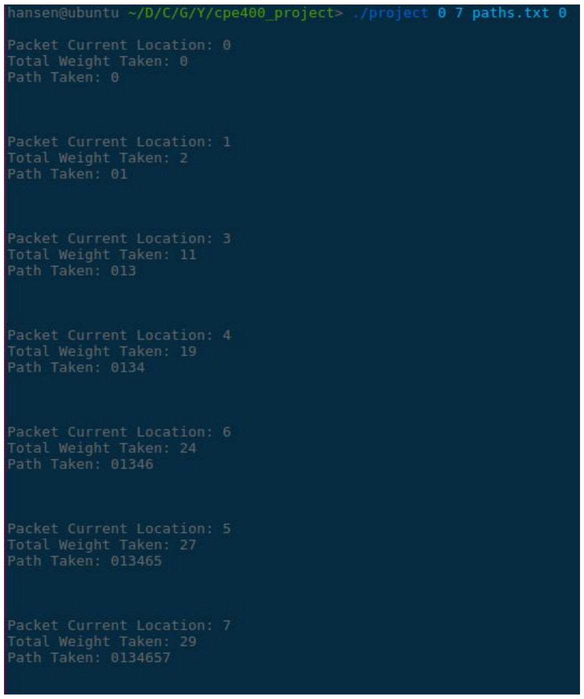


Figure 1: The above image is the layout of the used mesh network. It also displays the weights of all pathways as well as being able to visibly see where the bottleneck is (node 3).

Description and Algorithms:

When the simulation is executed from the command line, multiple arguments are read in. The arguments read in are as follows: starting node, ending node, the node configuration file, and simulator running flag. The start node will define where the packet will originate from in the mesh. The end node defines the destination in the mesh for the packet. The node configuration file will give the simulation the layout of the mesh. It also tells each node who it is connected to. The final parameter will let the simulation run all at once, or in a step by step process. When running in a step by step process, it states where the packet is currently at, what the path it has taken is, as well as the total weight of the given path.

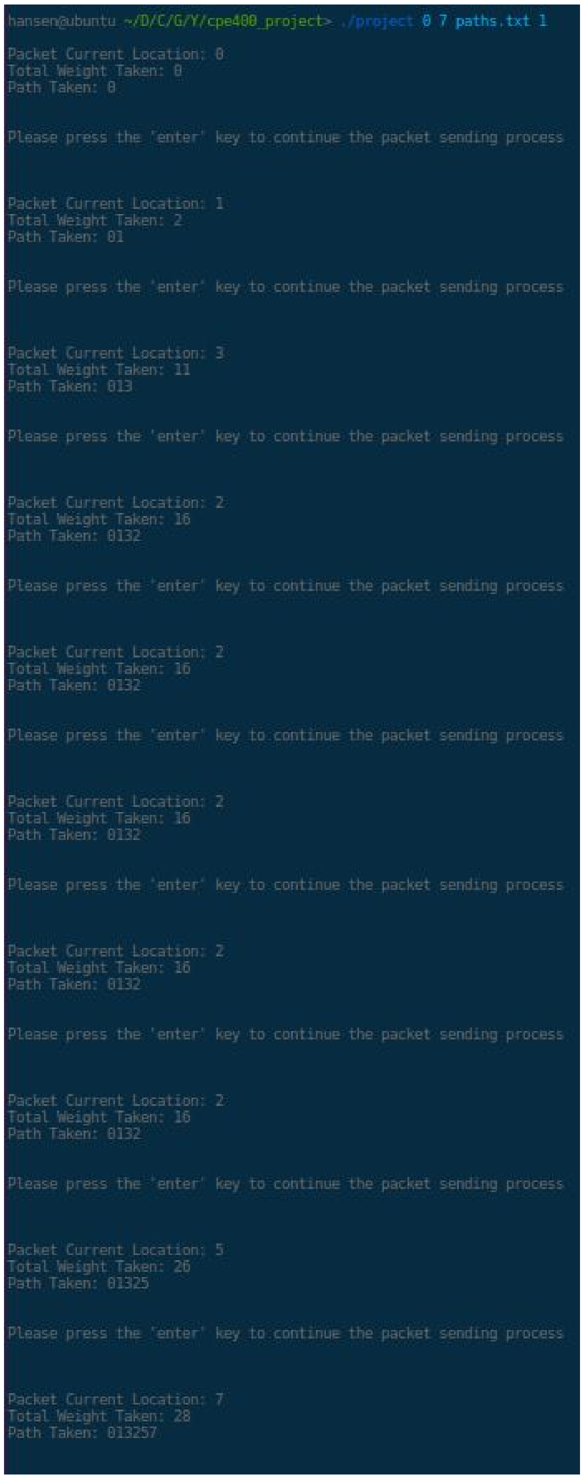
At the beginning of the simulation, the start node will broadcast to all connected nodes where the destination will be. Every subsequent node will then do the same thing until every node in the mesh is aware of where the destination node is. Once every node knows where they will be sending a packet, they compute their optimal pathway to the destination node, as well as a number of suboptimal pathways. Each node in the network finds their pathways using a depth first search algorithm. Each iteration of the search will only return a pathway if the final node is in the search path. Once it reaches the final node, that iteration of the search is stopped, and the pathway is returned and stored in a list for the current node. The priority of the paths found are based off of the total weight of the pathways used to get to the destination.



```
hansen@ubuntu ~/D/C/G/Y/cpe400_project> ./project 0 7 paths.txt 0

Packet Current Location: 0
Total Weight Taken: 0
Path Taken: 0


Packet Current Location: 1
Total Weight Taken: 2
Path Taken: 01


Packet Current Location: 3
Total Weight Taken: 11
Path Taken: 013


Packet Current Location: 4
Total Weight Taken: 19
Path Taken: 0134


Packet Current Location: 6
Total Weight Taken: 24
Path Taken: 01346


Packet Current Location: 5
Total Weight Taken: 27
Path Taken: 013465


Packet Current Location: 7
Total Weight Taken: 29
Path Taken: 0134657
```

Figure 2: The above image shows what the output will look like from the simulation if the user does not want to progress the simulation step by step.

Once every node has stored the data of its pathways, the packet is sent out from the start node. During each step in the simulation, every node has a randomly generated chance to fail, losing its connection to all nodes. If one of the nodes in the optimal pathways fails, the current node with the packet will reroute to one of its suboptimal pathways. The pathway that is chosen is based off of the priority of the pathway. As stated before, the priority is based off of the overall weight of the pathway to the destination. The pathway is also selected if all nodes are not currently down in the selected path. This allows the network to continue running without any issues passing the data packet.



```
hansen@ubuntu ~/D/C/G/Y/cpe400_project> ./project 0 7 paths.txt 1

Packet Current Location: 0
Total Weight Taken: 0
Path Taken: 0


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 1
Total Weight Taken: 2
Path Taken: 01


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 3
Total Weight Taken: 11
Path Taken: 013


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 2
Total Weight Taken: 16
Path Taken: 0132


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 2
Total Weight Taken: 16
Path Taken: 0132


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 2
Total Weight Taken: 16
Path Taken: 0132


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 2
Total Weight Taken: 16
Path Taken: 0132


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 2
Total Weight Taken: 16
Path Taken: 0132


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 5
Total Weight Taken: 26
Path Taken: 01325


Please press the 'enter' key to continue the packet sending process


Packet Current Location: 7
Total Weight Taken: 28
Path Taken: 013257
```

Figure 3: The above image shows what the output will look like from the simulation if the user wants to be able to step the simulation by itself.

One problem that can occur with a mesh network is a bottleneck point. The bottleneck occurs where there is a single specific node that data must travel through to get to the rest of the network. If the node prior to the bottleneck point currently has the packet, and the bottleneck node goes down. The current node with the packet holds onto the packet and waits until the bottleneck reopens. This is the most efficient way to handle the bottleneck point because the packet is maintained in a certain location, and excess communication is not required. This also helps maintain the integrity of the network because it is not risking running into another downed node somewhere else in the mesh.

Once the packet is received by the final node, the pathway the packet took is printed into the terminal window. The simulation will also state the overall weight of the path that was taken by the data packet. The simulation does not do an acknowledgement message for simplification of the simulation, since this simulation is focused on dynamic routing rather than basic network functionality. The simulation then ends and the user is able to check the terminal window to see each step of the process as the packet was simulated being sent between the start and final node. The user will also be able to see each nodes optimal pathway to the destination node, as well as a number of the suboptimal pathways to reach the destination.

Data and Results:

In a random failure situation like that of a mesh network it can be very difficult to measure meaningful results from a new protocol. While the obvious goal is to improve speed and reliability, the randomness of the network means that results between simulations cannot be simply compared to one another. For instance, referring to figure 1, one iteration of the simulation could see node 2 fail, while the next iteration has node 4 fail. Since different nodes failed, it is difficult to find meaning when comparing the total path weight of these two simulations.

While comparison between any two given iterations of the simulation may be insufficient, there is significance in comparing the total weight of several iterations to that of the optimal path between any two given nodes. The average loss of speed from the optimal path is considered to be the cost of extensive reliability.

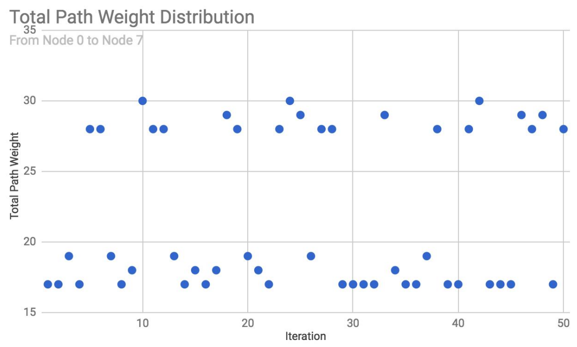


Figure 4: The total traversed path weight over 50 iterations of the simulation. Different results are caused by nodes in the optimal path failing, forcing packets to take a longer route.

Figure 4 illustrates a distribution of results over 50 iterations of the mesh network simulation from node 0 to node 7 (see figure 1). It is important to note that the optimal path (disregarding failed nodes) from nodes 0 to 7 results in a total weight of 17 in arbitrary time units. Analyzing these results yields an average of just 21.96, a relatively low inherited cost for reliability in the mesh network. While the increased average cost of 4.96, or 29.17% may be considered worthwhile to many, more research should be conducted before widespread implementation.

Conclusions:

The dynamic rerouting algorithm introduced in this paper involves increased computation, memory space, and speed overhead. In return it offers substantially improved reliability for mesh networks. Given that mesh networks are considered to be dynamic and unreliable by nature, it is the opinion of this team that this solution is worth future exploration. This is especially true when considering the continuously falling cost of memory space and CPU performance in modern computing devices [2].

This mesh network solution is still in its infancy. Future work will involve significantly more testing, specifically on larger simulated mesh networks and across more varied terrains in those networks. More data from different starting and ending nodes in various network configurations, starting with the one illustrated in figure 1, will also be necessary. Once simulated networks have been sufficiently explored, it will be time to begin development of deployable software for various mesh network systems. The rigorous testing performed on the simulated systems will have to be repeated for the real world systems.

References:

[1] A. Al-Saadi, R. Setchi, Y. Hicks and S. M. Allen, "Routing Protocol for Heterogeneous Wireless Mesh Networks," in *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9773-9786, Dec. 2016.

[2] M. Golio, "Fifty Years of Moore's Law [Scanning Our Past]," in *Proceedings of the IEEE*, vol. 103, no. 10, pp. 1932-1937, Oct. 2015.