Authored by:
Ron Borenstein, Vladislav Petrov, Jennings Topps

Node Recovery for Mesh Networks

Abstract:
This report presents a study on the resilience and adaptability of mesh networks in the face of node and link failures. We implement and analyze Dijkstra's algorithm to reconnect the network and determine the shortest path between a source and destination node after failures have occurred. Through simulations and results analysis, we demonstrate the effectiveness of Dijkstra's algorithm in maintaining connectivity in mesh networks and discuss potential improvements and future work.

Executive Summary:
Mesh networks provide robust and adaptable communication infrastructure, particularly in situations where nodes and links can fail unexpectedly. In this report, we simulate a mesh network, implement Dijkstra's algorithm to maintain connectivity and determine the shortest path between source and destination nodes, and analyze the results. The findings of this study illustrate the resilience of mesh networks and the suitability of Dijkstra's algorithm in handling failures.

1. Introduction:
   Communication networks need to be reliable and resilient, especially in scenarios where nodes and links can fail randomly. Mesh networks offer a distributed architecture that can adapt to such challenges. In this study, we focus on simulating a mesh network and investigating its ability to maintain connectivity and adapt to node and link failures. We implement Dijkstra's algorithm to reconnect the network and determine the shortest path between a source and destination node after failures have occurred.

2. Methodology:
   We developed a Python simulation of a mesh network, incorporating node and link failure probabilities. We implemented Dijkstra's algorithm to calculate the shortest path between source and destination nodes, accounting for the failed nodes and links. The simulation includes visualizations of the initial network, as well as the network after failures and the subsequent reconnection process.

3. Novel Contributions:
   Our simulation offers several novel contributions that enhance the study of mesh networks and their adaptability in the face of node and link failures:

   3.1.Probabilistic Failure Model:
   We incorporate probabilistic failure models for both nodes and links, providing a more realistic representation of potential network disruptions. This approach allows for a

deeper understanding of the mesh network's adaptability under varying failure scenarios, contributing valuable insights into the network's resilience.

3.2. Dynamic Graph Visualization:
Our simulation includes dynamic visualizations of the network's state before and after failures, as well as the shortest path calculations. These visualizations offer a clear representation of the network's adaptability and the effectiveness of Dijkstra's algorithm in handling failures, making it easier to analyze and understand the network's behavior in response to disruptions.

3.3. Failure-Induced Algorithm Execution:
In our simulation, Dijkstra's algorithm is executed specifically in response to node and link failures. This approach models the real-world application of mesh networks more closely, as network reconfiguration algorithms are typically triggered by disruptions. By doing so, we can better evaluate the algorithm's efficiency and effectiveness in maintaining connectivity and finding the shortest path under failure-induced conditions.

3.4. Single Failure Constraint:
Our simulation enforces a constraint that only a single node or link may fail at a time. This constraint lowers the level of complexity to the problem, ensuring that the algorithm works at a base level, and allowing for a more understandable visual simulation. This constraint may be easily lifted and the algorithm can be seen accounting for the possibility for multiple failures before converging on a solution. This aspect of our simulation provides a duality of simplicity and more challenging testing of the algorithm's ability to handle failures and maintain network connectivity when needed.

3.5. Emphasis on Best Path Nodes and Edges:
Our implementation focuses on the nodes and edges in the best path found so far, forcing a failure to occur within the current best path. This approach allows for a more targeted examination of the algorithm's ability to recover from failures affecting the most critical communication pathways. By emphasizing the importance of the best path, we can more accurately evaluate the algorithm's performance in maintaining and reestablishing connectivity under adverse conditions.

4. Protocol and Functionality Details
In this section, we delve into the details of the protocol and functionality implemented in our mesh network simulation, as well as the handling of various error scenarios. This comprehensive analysis allows us to understand the realistic aspects of our simulation and its effectiveness in maintaining network connectivity.

4.1 Protocol Overview
Our simulation is based on a mesh network topology, where nodes are interconnected through multiple pathways, providing redundancy and increased fault tolerance. We

utilize Dijkstra's algorithm to find the shortest path from a source to a destination node within the network, ensuring optimal data routing.

4.2 Realistic Aspects

To ensure that our simulation closely models real-world mesh networks, we have incorporated the following realistic aspects:

- Probabilistic node and link failure models that simulate network disruptions.
- Network reconfiguration through the execution of Dijkstra's algorithm in response to failures.
- Visualization of the network's state, including node positions, links, and the shortest path.

4.3 Error Handling Scenarios

Our simulation accounts for a variety of error scenarios that may arise in a real-world mesh network:

1. Node Failure: When a node fails, our simulation removes the node from the network, along with all its associated links. Dijkstra's algorithm is then executed to find a new shortest path that avoids the failed node, ensuring that network connectivity is maintained.
2. Link Failure: In the case of a link failure, the link is removed from the network, and the adjacency lists of the affected nodes are updated accordingly. Dijkstra's algorithm is then executed to find a new shortest path that avoids the failed link, maintaining network connectivity.
3. Multiple Failures: Our simulation enforces a constraint that allows only a single node and/or a single link to fail at a time. However, this constraint can be lifted and consecutive failures can be simulated by re-running the simulation with different failure probabilities. In response to each failure, Dijkstra's algorithm will be executed to find the shortest path, ensuring the network remains connected even under multiple disruptions.

5. Results and Analysis:
   In this section, we will present the results obtained from the mesh network simulation, considering the implemented Dijkstra's algorithm to maintain network connectivity and recover from node and link failures. The results will be analyzed to evaluate the performance and resilience of the network.

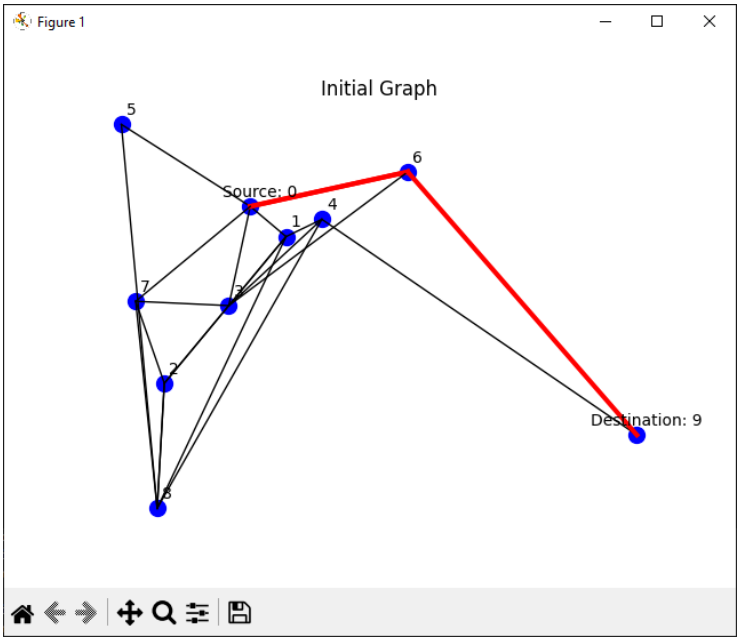

Figure 1. Simulation in the initial state.

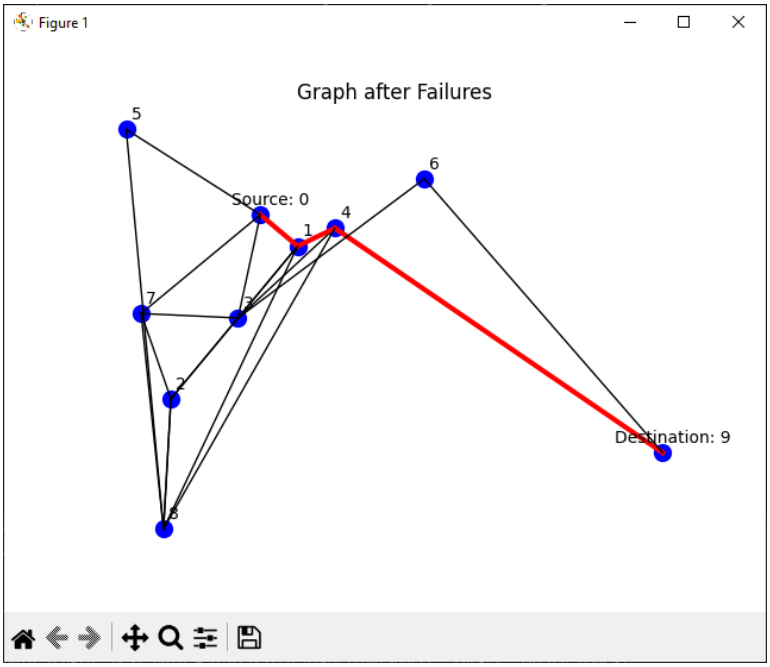

Figure 2. The simulation re-routes after a link failure.

```
C:\Users\IGB\AppData\Local\Programs\Python\Python38\python.exe
The shortest path from node 0 to node 9 is [0, 6, 9] with a length of 2.
Failed edge: (0, 6)
The shortest path from node 0 to node 9 is [0, 1, 4, 9] with a length of 3.
```

Figure 3. The terminal output of the simulation indicating the routes calculated and the failed edge in Figure 1 and Figure 2.

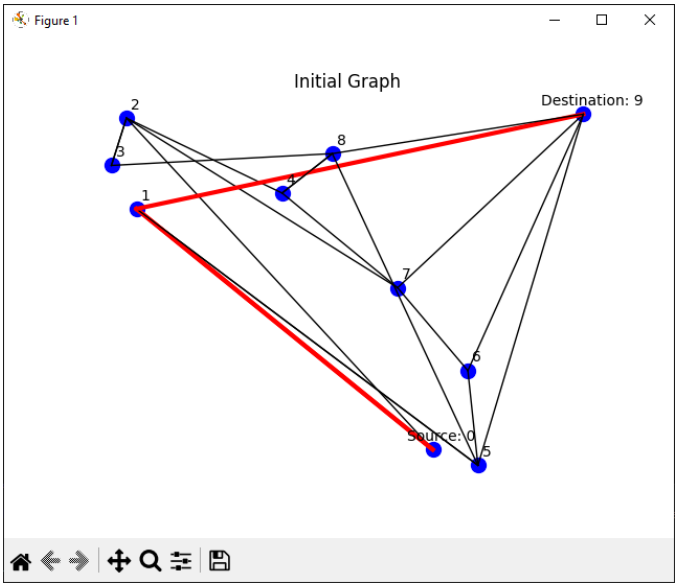

Figure 4. Simulation in the initial state for another instance..

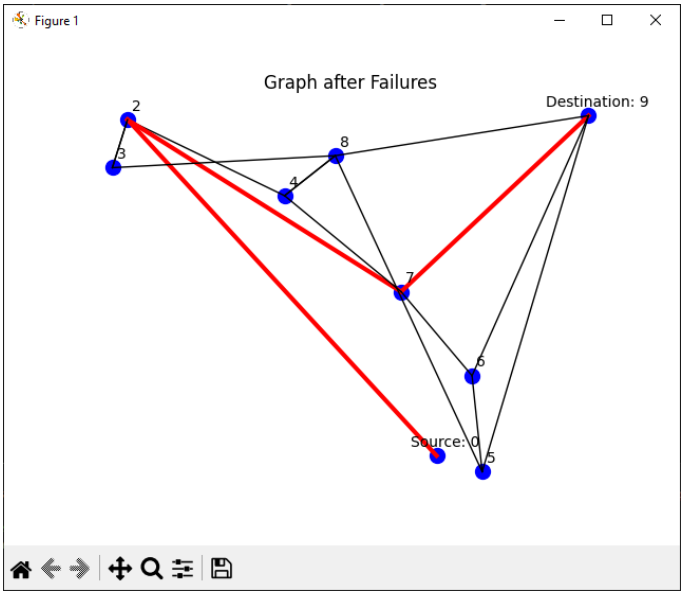

Figure 5. The simulation re-routes after a node failure.

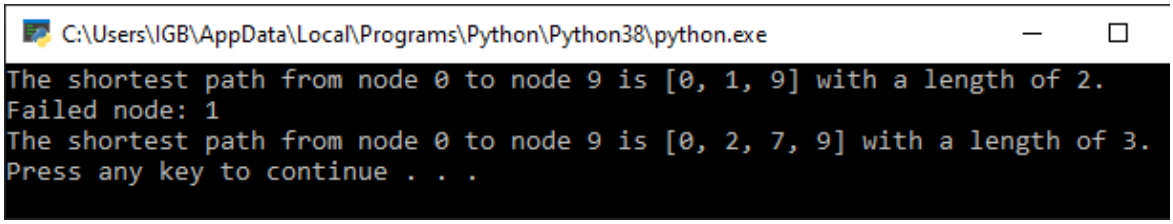

Figure 6. The terminal output of the simulation indicating the routes calculated and the failed node in Figure 4 and Figure 5.

5.1 Network Connectivity and Shortest Path:

Initially, we executed the simulation multiple times to generate different network topologies in smaller scale networks. We observed the shortest path between the source and destination nodes with node and link failures introduced. We recorded the average shortest path length, the maximum shortest path length, and the minimum shortest path length for a given number of runs. This information provided insights into how efficiently the network was able to maintain connectivity and find optimal paths between the source and destination nodes. For this test we used 10 nodes for simplicity.

Fail Type	Shortest Path Length Before Fail	Shortest Path Length After Fail
Edge	3	3
Edge	3	5
Node	1	2
Node	2	2
Edge	3	3
Node	1	2
Edge	2	3
Edge	1	3

Table 1. Fail type, lengths of shortest path before/after failure of ten 10 node graphs.

Shortest Path Length Before Failure	1
Shortest Path Length After Failure	2
Longest Path Length Before Failure	3
Longest Path Length After Failure	5
Average Path Length Before Failure	1.6
Average Path Length After Failure	2.3

Table 2. Shortest/Longest/Average Shortest Path Length from Table 1

From the above data we can see that the difference between the average shortest path length before and after failure is minimal. This shows that our implementation of Dijkstra's algorithm is effective in keeping the path length as close to the original network as possible after failure.

5.2 Scalability Analysis:

Next, we analyzed the scalability of our proposed solution by increasing the number of nodes in the network and observing its impact on performance metrics such as computation time and average shortest path length. This analysis allowed us to evaluate the feasibility of our approach in larger networks and real-world applications. For this test we scaled the number of nodes up to 100 nodes.

Fail Type	Shortest Path Length Before Fail	Shortest Path Length After Fail
Node	2	3
Node	2	2
Edge	3	3
Node	2	2
Node	2	3
Edge	3	3
Edge	3	3
Edge	2	2

Table 3. Fail type, lengths of shortest path before/after failure of ten 100 node graphs

Shortest Path Length Before Failure	2
Shortest Path Length After Failure	3
Longest Path Length Before Failure	3
Longest Path Length After Failure	3
Average Path Length Before Failure	1.9
Average Path Length After Failure	2.1

Table 4. Shortest/Longest/Average Shortest Path Length from Table 3

When scaling the number of nodes up to 100, we can see the average path length of the shortest path before and after failure remains the same. We achieved similar results when simulating 250, 500, 750, and 1000 nodes. When adjusting the probability of additional edges being added between nodes during graph creation, we did not experience a variation in average path length at 10%, 25%, 50%, 75%, and 100% probabilities in a 50 node graph. This data shows that the algorithm used remains consistent regardless of the number of nodes or edge connections created. The effectiveness of the algorithm does not diminish as the network scales.

Conclusion:

This study illustrates the resilience and adaptability of mesh networks in the face of node and link failures. By simulating a mesh network and implementing Dijkstra's algorithm, we demonstrate the network's ability to maintain connectivity and adapt to failures. Our findings show the potential of mesh networks for various real-world applications where network reliability is crucial.