University of Nevada, Reno
CPE 400 Fall 2022

Final Report: Dynamic Routing Mechanism Design in Faulty Network

Authors:
Gerard Bensadoun Gutsens
Zephyr Vincent

Instructor:
Prof. Shamik Sengupta

Protocol Functionality:

Part I - Simulated Network Topology

Our protocol simulates a basic network topology with 10 nodes and 19 interconnecting links, with weights as shown below in Figure 1, in a C++ environment through the implementation of a weighted graph. Our simulated network has a list of nodes N = {A, B, C, D, E, F, G, H, I, J} and a list of links L = {(A, B), (A, C), (A, D), (B, C), (B, D), (C, H), (C, G), (D, E), (D, F), (E, G), (E, I), (E, J), (F, J), (G, H), (G, I), (G, J), (H, I), (H, J), (I, J)}.

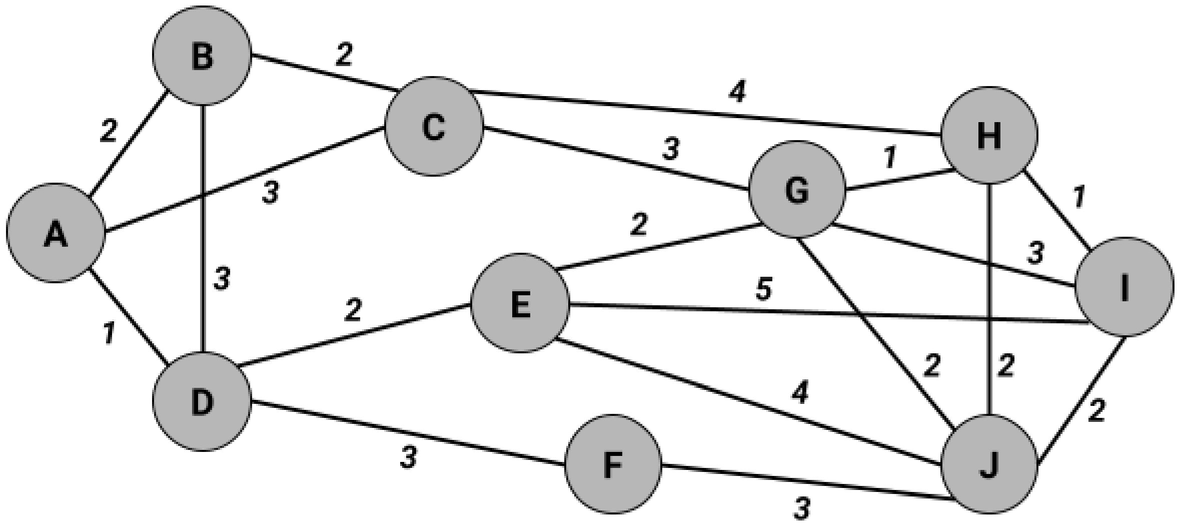
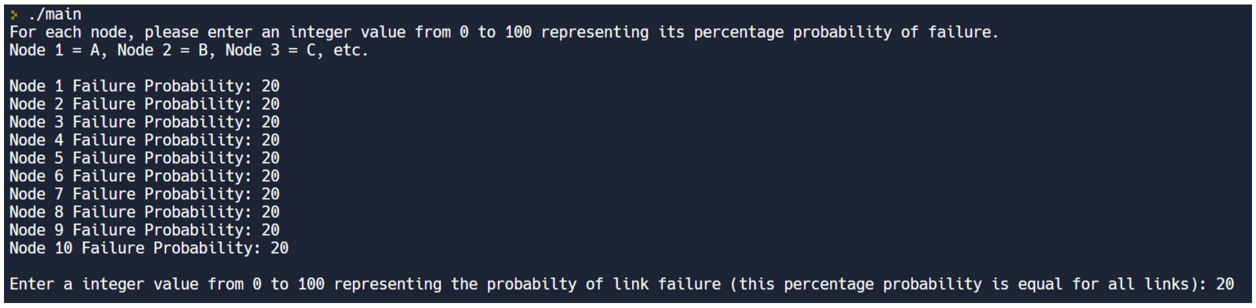


Figure 1: Router Network Topology Design Used in Simulation, Including Link Costs

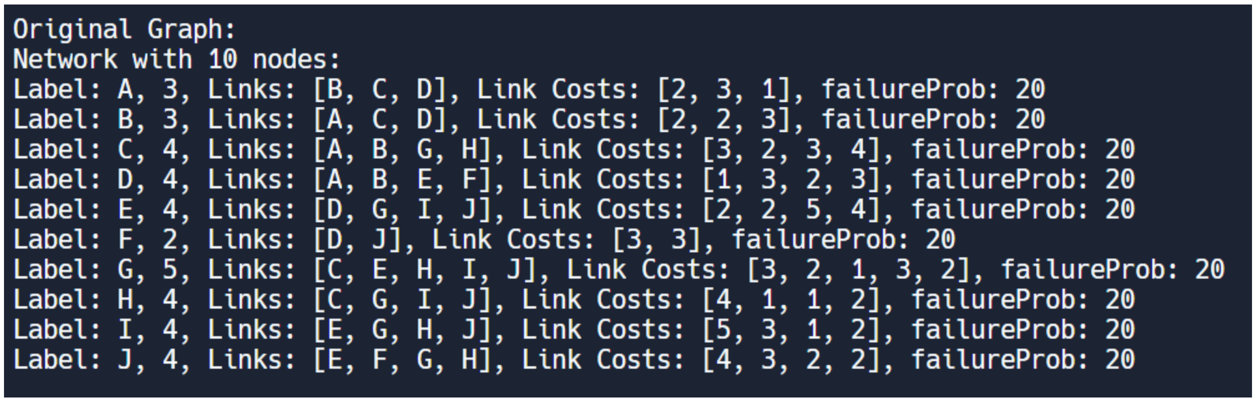
As part of our simulation, each node and link in the network has a set probability of failing, which is inputted by the user (as shown below in Figure 2). Our simulation protocol prompts the user to enter an integer probability from 0% to 100% of each node failing as well as an integer probability of link failure. All links across the network have the same probability of failure.



```
> ./main
For each node, please enter an integer value from 0 to 100 representing its percentage probability of failure.
Node 1 = A, Node 2 = B, Node 3 = C, etc.

Node 1 Failure Probability: 20
Node 2 Failure Probability: 20
Node 3 Failure Probability: 20
Node 4 Failure Probability: 20
Node 5 Failure Probability: 20
Node 6 Failure Probability: 20
Node 7 Failure Probability: 20
Node 8 Failure Probability: 20
Node 9 Failure Probability: 20
Node 10 Failure Probability: 20

Enter a integer value from 0 to 100 representing the probabilty of link failure (this percentage probability is equal for all links): 20
```

Figure 2: Showing User Input of Failure Probabilities

The next step of our protocol is to print out a visual display of the network nodes, links, link costs, and failure probabilities to the user, as shown below in Figure 3. This is the textual representation of the router network topology shown above in Figure 1.

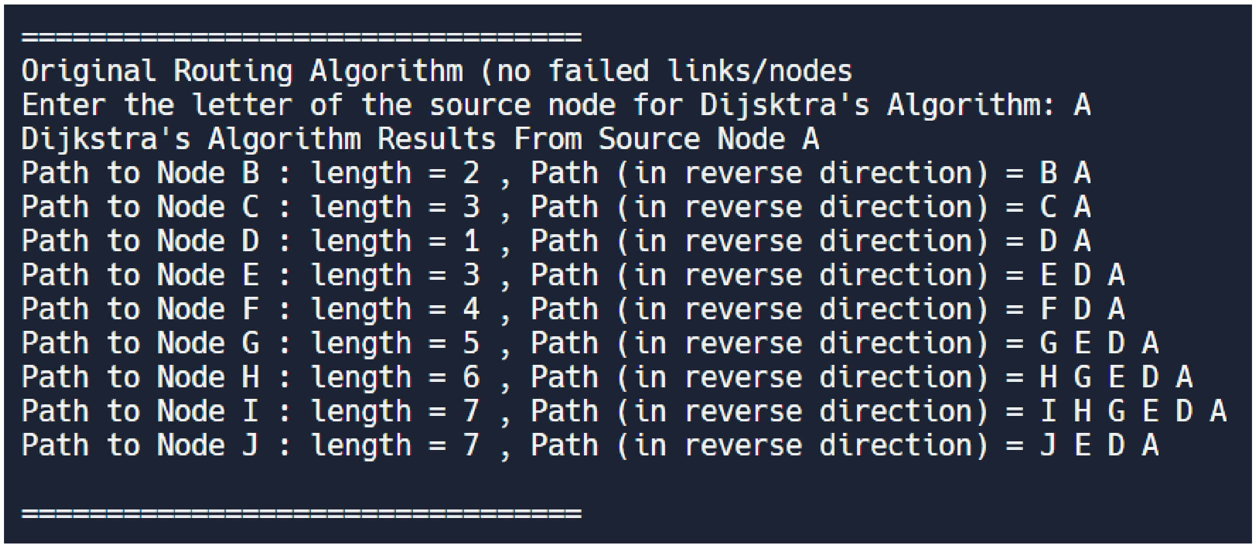


```
Original Graph:
Network with 10 nodes:
Label: A, 3, Links: [B, C, D], Link Costs: [2, 3, 1], failureProb: 20
Label: B, 3, Links: [A, C, D], Link Costs: [2, 2, 3], failureProb: 20
Label: C, 4, Links: [A, B, G, H], Link Costs: [3, 2, 3, 4], failureProb: 20
Label: D, 4, Links: [A, B, E, F], Link Costs: [1, 3, 2, 3], failureProb: 20
Label: E, 4, Links: [D, G, I, J], Link Costs: [2, 2, 5, 4], failureProb: 20
Label: F, 2, Links: [D, J], Link Costs: [3, 3], failureProb: 20
Label: G, 5, Links: [C, E, H, I, J], Link Costs: [3, 2, 1, 3, 2], failureProb: 20
Label: H, 4, Links: [C, G, I, J], Link Costs: [4, 1, 1, 2], failureProb: 20
Label: I, 4, Links: [E, G, H, J], Link Costs: [5, 3, 1, 2], failureProb: 20
Label: J, 4, Links: [E, F, G, H], Link Costs: [4, 3, 2, 2], failureProb: 20
```

Figure 3: Program Text Output of Original Network Design Shown in Fig. 1

Part II - Routing Algorithm

The routing algorithm implemented in this simulated network design was Dijkstra's algorithm. Each time the network routing algorithm runs, it outputs the path to each node in the network from the user inputted source node along with the cost of that path (the length). The output of running Dijkstra's algorithm from source node 'A' on the original network topology (with no node or link failures) is shown below in Figure 4.

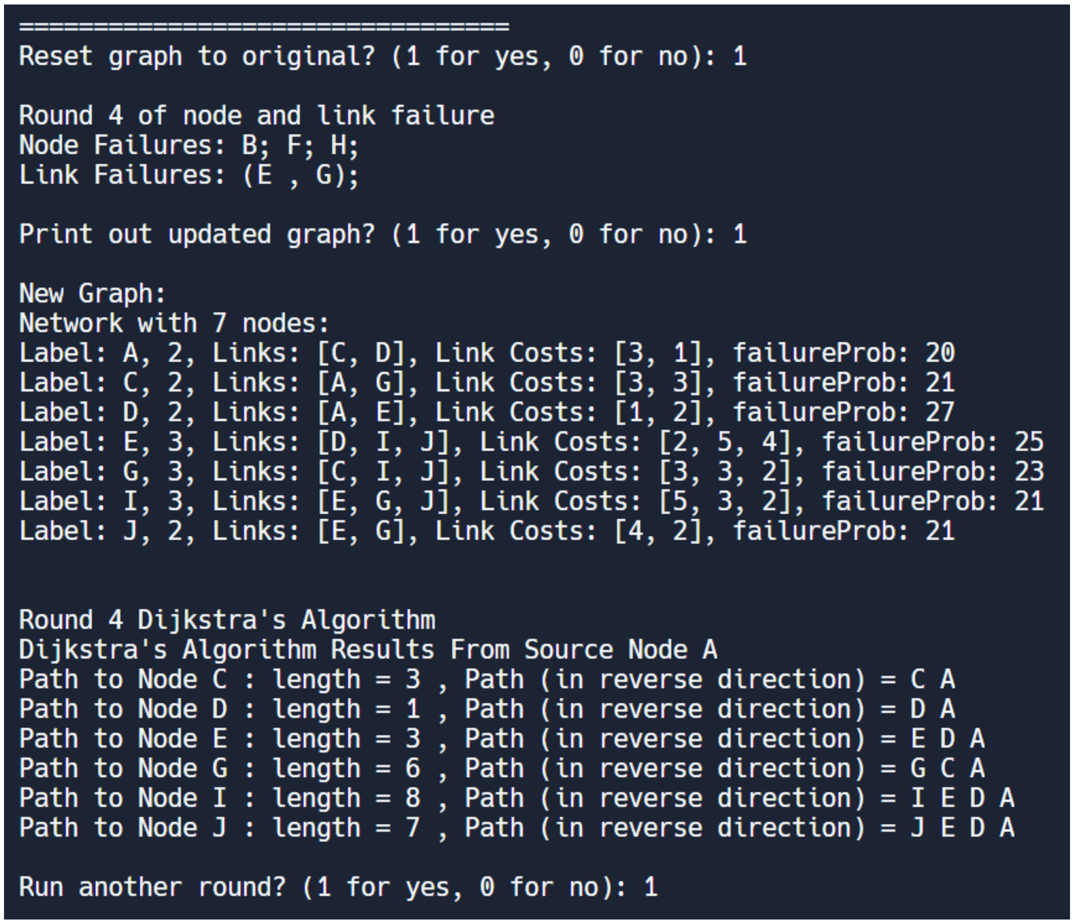


```
=================================
Original Routing Algorithm (no failed links/nodes
Enter the letter of the source node for Dijsktra's Algorithm: A
Dijkstra's Algorithm Results From Source Node A
Path to Node B : length = 2 , Path (in reverse direction) = B A
Path to Node C : length = 3 , Path (in reverse direction) = C A
Path to Node D : length = 1 , Path (in reverse direction) = D A
Path to Node E : length = 3 , Path (in reverse direction) = E D A
Path to Node F : length = 4 , Path (in reverse direction) = F D A
Path to Node G : length = 5 , Path (in reverse direction) = G E D A
Path to Node H : length = 6 , Path (in reverse direction) = H G E D A
Path to Node I : length = 7 , Path (in reverse direction) = I H G E D A
Path to Node J : length = 7 , Path (in reverse direction) = J E D A

=================================
```

Figure 4: Output of Dijkstra's Routing Algorithm with no Node or Link Failure

Part III - Node and Link Failure

The last major portion of our protocol was implementing node and link failure across the network. Our protocol functionality allows the user to run multiple rounds of node/link failure, using a random number generator to simulate the probability of failure. Our protocol outputs the labels of any nodes or links which failed and provides the option for the user to display an updated network graph before running Dijkstra's algorithm on the updated network. An example of the output from a complete round of node/link failure is shown below in Figure 5. As seen from the output of Dijkstra's algorithm, our network protocol was successfully able to route around the failed nodes and links.

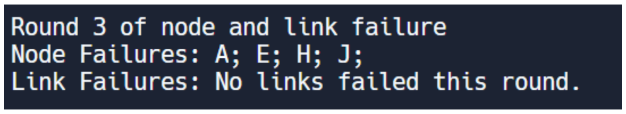


```
=================================
Reset graph to original? (1 for yes, 0 for no): 1

Round 4 of node and link failure
Node Failures: B; F; H;
Link Failures: (E , G);

Print out updated graph? (1 for yes, 0 for no): 1

New Graph:
Network with 7 nodes:
Label: A, 2, Links: [C, D], Link Costs: [3, 1], failureProb: 20
Label: C, 2, Links: [A, G], Link Costs: [3, 3], failureProb: 21
Label: D, 2, Links: [A, E], Link Costs: [1, 2], failureProb: 27
Label: E, 3, Links: [D, I, J], Link Costs: [2, 5, 4], failureProb: 25
Label: G, 3, Links: [C, I, J], Link Costs: [3, 3, 2], failureProb: 23
Label: I, 3, Links: [E, G, J], Link Costs: [5, 3, 2], failureProb: 21
Label: J, 2, Links: [E, G], Link Costs: [4, 2], failureProb: 21


Round 4 Dijkstra's Algorithm
Dijkstra's Algorithm Results From Source Node A
Path to Node C : length = 3 , Path (in reverse direction) = C A
Path to Node D : length = 1 , Path (in reverse direction) = D A
Path to Node E : length = 3 , Path (in reverse direction) = E D A
Path to Node G : length = 6 , Path (in reverse direction) = G C A
Path to Node I : length = 8 , Path (in reverse direction) = I E D A
Path to Node J : length = 7 , Path (in reverse direction) = J E D A

Run another round? (1 for yes, 0 for no): 1
```

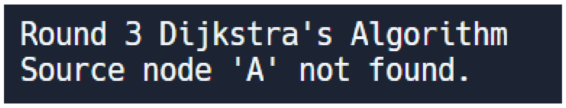
Figure 5: Example of a Complete Round of Node and Link Failure with Algorithm Output

Part IV - Error Handling

Part of the process of programming a functional network simulator involved error handling, especially in the case of failed nodes or links. An error output from a case in which no nodes or links fail despite their probabilities is shown in Figure 6 along with an example error output from the implementation of Dijkstra's algorithm. Some errors which were addressed for dijkstra's algorithm include cases in which the source node is a failed node or for the case in which all links connecting to the source node have failed.



```
Round 3 of node and link failure
Node Failures: A; E; H; J;
Link Failures: No links failed this round.
```



```
Round 3 Dijkstra's Algorithm
Source node 'A' not found.
```

Figure 6: Examples of Protocol Error Handling

Novel Contribution:

Added User Functionality

Part of the novel contribution added to our protocol simulation was a multitude of added user functionality. For instance, our protocol allows the user to select which source node Dijkstra's algorithm will use and the user has direct control over the number of rounds of node/link failure simulated. Additionally, for each round of node/link failure, we added the option for the user to restore the graph to it's original state, for cases in which prior failed nodes or links are restored. The result of these added user features can be seen above in Figure 5.

Node Degradation

Another novel contribution added to our protocol simulation involved adding results of node degradation into the experiment. The idea behind this protocol is that as more data passes through a node, the probability of node failure increases by a slight factor due to "wear and tear." In order to simulate this, we added a helper function which increased the probability of failure by a small factor every time the node is visited within a completed path of dijkstra's algorithm. The effect this has on the failure probabilities can be seen above in Figure 5, as the failure probability of all the nodes is no longer the initial assigned value of 20%.

Results and Analysis:

Overall, our protocol did an excellent job at routing around failed nodes and links and handling error cases in which routes are not possible. The shortest path tree results of Dijkstra's algorithm implement on the original graph with no node or link failure is shown below in Figure 7 along with its corresponding forwarding table in Table 1.

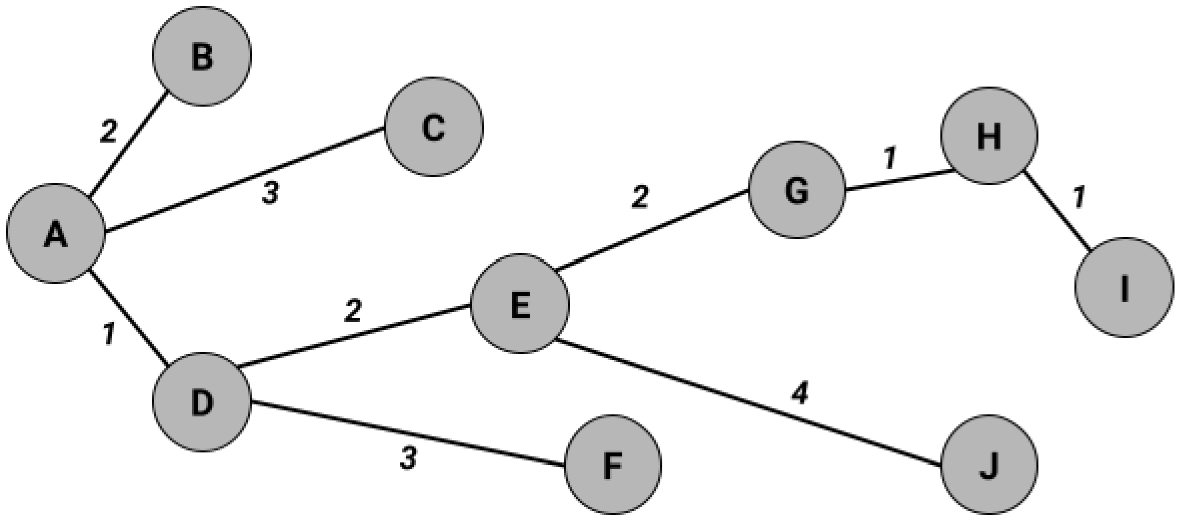


Figure 7: Shortest Path Tree for Source Node 'A' and no Node or Link Failure

Table 1: Resulting Forwarding Table for Source Node 'A' and no Node/Link Failures

Destination	Initial Link
B	(A, B)
C	(A, C)
D	(A, D)
E	(A, D)
F	(A, D)
G	(A, D)
H	(A, D)
I	(A, D)
J	(A, D)

The results of our protocols' adaptations to two rounds of node and link failure are shown in the tables below. In the first round, 2 nodes (C and J) failed and 1 link (G, I) failed. The resulting graph and dijkstra's algorithm path are shown below. In round 2, the node B also failed along with the link (A, D). This resulted in one of the error handling situations, as there are no longer any links connected to the source node 'A.'

Round 1 of node and link failure	
Node failures	C, J
Link failures	(G, I)

Round 1 Graph: Network with 8 nodes			
Label	Links	Link costs	failureProb
A, 2	[B,D]	[2, 1]	20
B, 2	[A, D]	[2, 3]	21
D, 4	[A, B, E, F]	[1, 3, 2, 3]	27
E, 3	[D, G, I]	[2, 2, 5]	25
F, 1	[D]	[3]	21
G, 2	[E, H]	[2, 1]	23
H, 2	[G, I]	[1, 1]	22
I, 2	[E, H]	[5, 1]	21

Round 1 Dijkstra’s Algorithm Results From Source Node A		
Path to Node	Length	Path (in reverse direction)
B	2	B A
D	1	D A
E	3	E D A
F	4	F D A
G	5	G E D A
H	6	H G E D A
I	7	I H G E D A

Round 2 of node and link failure	
Node failures	B
Link failures	(A, D)

Round 2 Graph: Network with 7 nodes			
Label	Links	Link Costs	failureProb
A, 0	[]	[]	20
D, 2	[E, F]	[2, 3]	33
E, 3	[D, G, I]	[2, 2, 5]	29
F, 1	[D]	[3]	22
G, 2	[E, H]	[2, 1]	26
H, 2	[G, I]	[1, 1]	24
I, 2	[E, H]	[5, 1]	22