# COMP3221 Assignment 1 Report

## Network Topology:

### Creating Our Network Topology

* Firstly, we add a specific number of nodes to the graph with the first node starting at the letter A.
* Secondly, since all the nodes are disconnected, we pick a random node in the connected graph (that has a degree of less than 3) which will, at first, be the first node in the disconnected graph. We then pick another random node in the disconnected graph and add an edge between the two chosen nodes. Finally, the node chosen from the disconnected graph is removed and this process is repeated until all nodes are connected.
* If the number of edges wanted is greater than the number of nodes, a second loop will happen to add edges between nodes at random.
* After all edges are assigned, they are then given a random weight from 0.1 to 9.00.
* A configuration file is then written for all nodes where it contains the degree of their node, it’s neighbours and the ports and weights associated with those nodes.

The first stage results with a fully connected graph with minimum one link between each node:

A network of dots and lines

Description automatically generated

The next stage prioritises nodes with less than 4 connections and adds connections randomly:

A diagram of a network

Description automatically generatedIt is observable that the topology has the original structure, but simply has more connections (such as B now connecting to C etc.

Finally, weights are added to the diagram randomly as discussed previously.

A diagram of a network

Description automatically generated

As such, the program always creates a fully connected graph. Additionally, built in functions allow for the user to specify how many nodes and links they would like in the graph. For basic testing of the node functionality a standard 3 node graph with 2 connections was used.

## Routing Algorithm:

### Outline the routing algorithm(s), explaining your selection and providing an overview of its functionality. Highlight any modifications or optimizations made to standard algorithms.

### 2.1 Generalized implementation

* Through implementing Dijkstra’s algorithm, we were able to create a shortest path routing algorithm.
* Firstly, we set a node dictionary where it would start at a given node and will hold the least cost path to every other node as well as the nodes that are visited on the path.
* Secondly, the algorithm will find all the neighbours of the current node.
* We will greedily explore the least cost path iteratively, until all paths have been searched.
* We update path costs when they are better than the original pathing.
* To implement the greedy nature of the algorithm we used a priority queue that allows the user to pop the least great distance and explore that node etc.
* As new nodes are discovered, their distance and the node themselves are added to this priority queue and looped through accordingly (whilst there are still nodes to discover.
* Similar to the original Dijkstra’s algorithm that ours is modelled off, it has not been tested for negative weights (as we assumed there would be no negative weights) but can handle infinite weightings.

### 2.2 Modifications and Optimizations

* Instead of finding the least cost from a node to a target node. It finds the least cost to every node. Dijkstra’s algorithm will stop exploring when it finds paths that exceed its current best path. Our algorithm continues to explore as we are searching for all nodes, not just a target node. Hence our algorithm is tailored to the specific needs of finding the shortest path from one node to all the other nodes.

## Implementation Methodology:

### 3.1 Node Object Implementation

In our program we have implemented our own node object, so we are able to keep track of its state (whether it is online or not) and the state of its neighbours.

The functionality inside the node object include:

* Its own map of the nodes its aware of and their edge costs.
* The ability to take itself online and offline (for testing purposes)
* Removing and adding a connection for when a node it is connected to goes offline or comes back on respectively. This allows for the network to handle node disconnections, and being aware of which nodes are online and should be included in the least cost path algorithm.

### Describe the programming approaches and tools used, with a focus on specific techniques applied to implement the routing protocols, handle link cost changes and node failures, and ensure continuous operation of the network.

The overall

Listening

Sending

Routing

The program makes the use of a CLI that helps test the networks ability to handle nodes that

## 4 Simulation Results:

### Summarize key findings from your simulations, stating what you have and haven’t completed. Provide specific examples of how effective your system is in routing and adapting to link-cost changes and failures