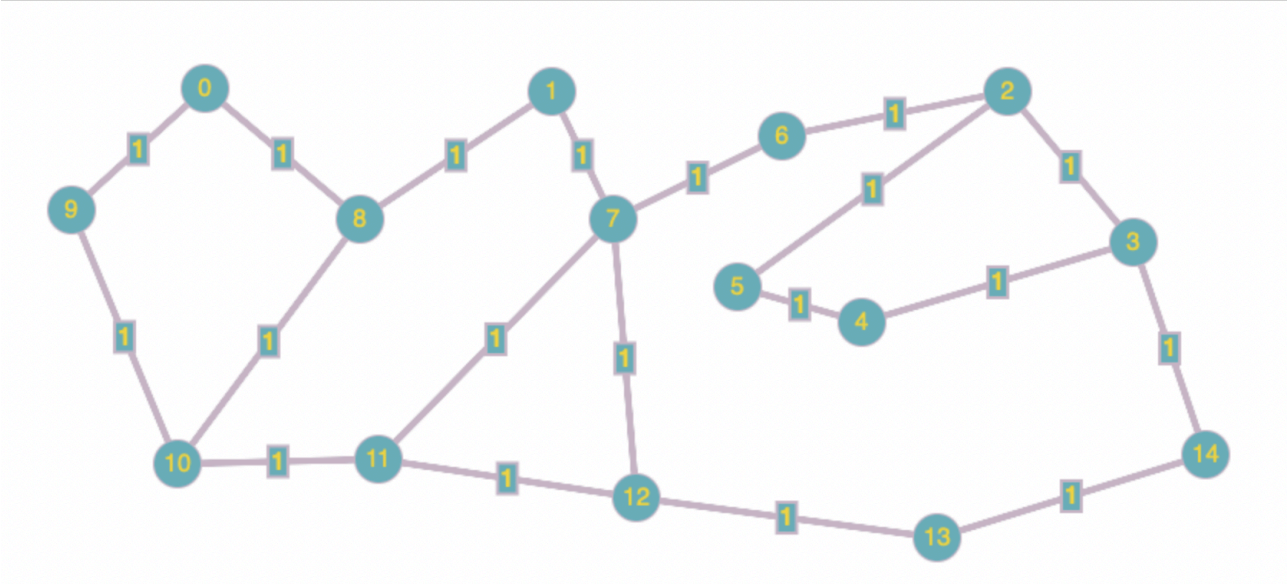
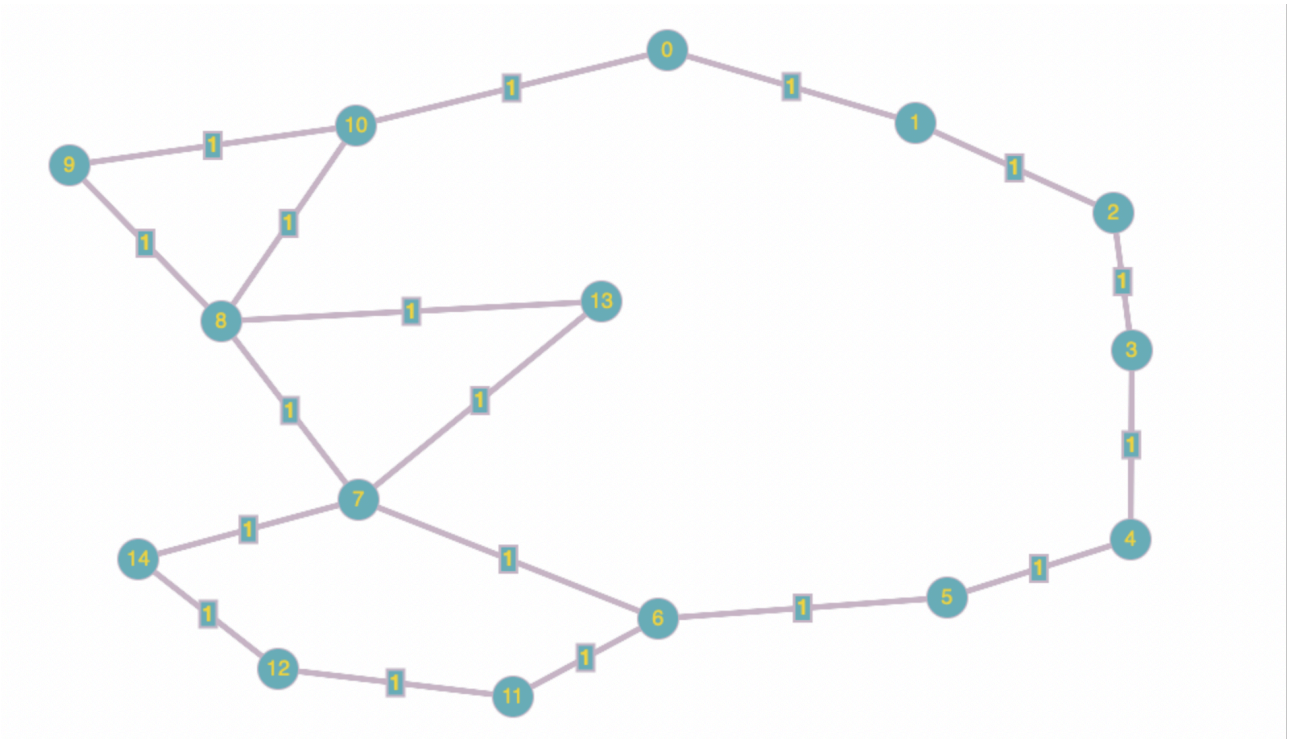
Distributed Algorithms CW Joseph Straw

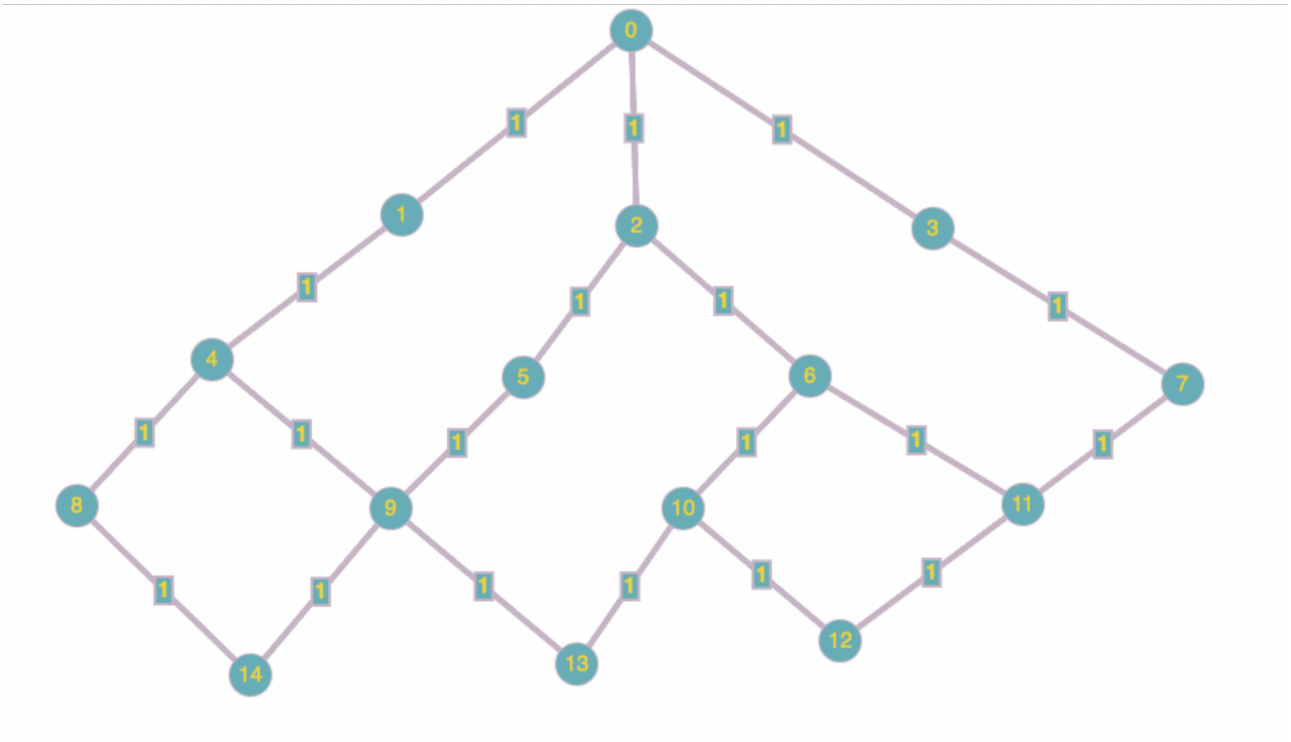
Network 1:



Network 2:



Network 3:



These three networks were made with the attempt to try and make them different topographically with 1 being more of a connected network, 2 being more circular and 3 being tree like in structure.

The included .TXT documents are the outputs from the program for each of these networks with the resultant number of messages sent and the average time taken for the execution.

These output files were generated by running the algorithm 20 times on each network from a random initiator node.

Network 1:
The total time taken for this execution was 276ms with an average time per iteration being 13ms with an average of 39 messages being sent. If the algorithm were recorded correctly it should be 2N where N is the number of edges. Network 1 has 19 edges so 38 messages should have been sent which means that my program was sending additional messages at some point and I think the recording was incorrect somewhere. The messages recorded in this case appear to have been 2N+1 which is 1 out on average.

Network 2:
The total time taken was 370ms with the average time being 18ms per iteration. This result makes sense to me due to the reduction in the connectivity in the network which results in longer chains of messages being sent and relayed, which ends up with more layers being generated that are not as wide. The average layer depth required was 10 which is

only 1 more than network 1 and 2 but it still took longer to execute. The most common number of messages recorded was 35. Network 2 has 18 edges so this time 2N-1 messages were received which is one under what it should have been.

Network3:
The total time taken for this execution was 265ms with an average of 13ms per iteration. This network averaged similarly to network 1 but there was more variance due to the start position having more influence on the end result in execution time. The layer depth average was also 9 so despite this network being more tree like the connectivity must have been similar which ended up giving similar results.

From this I can conclude that, my implementation at least, of the algorithm will execute marginally faster on networks that are more interconnected due to being able to generate trees that branch quicker with fewer layers. The networks that had less connectivity or were circular topographically forced the algorithm to generate more thin layers that are less effective at being multithreaded. The case where the network was very connected in some places and not so in others ended with a wider dispersal of time to execute the algorithm depending on the starting point, although most of the time the execution was constant. The structures of the networks that I did make could have been made better for differentiating such differences in the performance of the algorithm, and with further development using bigger networks or more networks could have shown this. Also increasing the number of iterations could have helped make a more informed conclusion.

(Instead of including tables in this document please refer to the network output documents for the results of the execution of the algorithm)