Algorithm for random topology generator:

Input:

1. *number of nodes,*
2. *number of uni-directional links (also referred to as links henceforth in the document)*
3. *number of random topologies to be generated*

Checks:

Minimum number of links required to generate a *strongly connected topology* in which each node can send message to any other node in the network *is equal to number of nodes*. The resulting strongly connected topology is a ring.

If number of links are less than number of nodes, then exit.

Maximum number of links required to connect all the nodes directly to all the other nodes in the topology is 2 \* nC2: (here *n* is the number of nodes present in the topology)

If number of links are more than maximum number of links, then exit

For each random topology:

*Generate a random ring*

1. Generate a random order of nodes containing all nodes using Fisher-Yates shuffle algorithm

1. Connect the order of nodes obtained in a ring topology by connecting the adjacent nodes

1. Store this ring topology thus generated in previous step in a *structure*

1. Repeat step 1, for next order of node
2. Check if the random order of nodes generated is already present in the *structure* such that there is no cyclic equivalent i.e., random ring topology generated with this order of nodes does not represent the same ring, topology generated before
3. If an equivalent node order found present in the *structure*, then discard this order of nodes and repeat step 1, to generate next order of nodes
4. If equivalent node order is not found present in the *structure,* then repeat step 2.
5. Keep generating unique rings; until the number of unique rings equal to number of random topologies to be generated (given by user input)

Generating an underlying unique random ring for each random topology ensures that the topology generated by adding extra links to the unique ring is strongly connected.

For each random ring topology generated:

1. Choose two distinct random nodes
2. If they are not already connected in the ring connect them with a link
3. Repeat step 1 until all links are exhausted (given by user input); Output the random topology generated

For each random topology generated from an underlying random ring topology:

1. Generate the ‘connectivity matrix’ which contains the information of how each node is connected to another node. For example, for a 4x4 Mesh, connectivity matrix is 16x16, where sender node represents the row and receiver node represent the column.
2. If there is 1 present in the connectivity matrix then there is a link which connects the sender node to the receiver node. Else there is 0 present if there is no direct link
3. Dijkstra’s Algorithm is then applied to the connectivity matrix to find the shortest path in terms of number of hops from given sender node to all the destination nodes in the topology.
4. The resulting matrix is called as *hop matrix*