# COMP9331 report

## ***Implementation***:

There are six mainly threads for implementing LSR protocol:

broadcasting_thread: This thread broadcast the routers packet  at an interval of 1 sec . These msgs are put in a queue called sq .

listening_thread: This thread listens to incoming messages and put it in a queue called rq if it a msg containing the link_state_packets and puts it in a queue called emqr if it a error msg(msg about death of node).

transferring_thread: This thread puts the incoming packets which are to be transferred to neighboring nodes in to a different queue called tq.

check_alive: This thread keeps track of whether a node died or not, and creates an error msg to be sent to all other neighboring nodes if one of the neighboring node dies.

sending_thread: This thread keeps processes sq, emq and tq and sends packets in these queues to their respective destinations.

dijkstra_thread: This thread prints the computes the dijkstras algorithm and prints the output every 30 seconds

## ***Restricting link-state broadcast implement:***

Each node will transfer packets of other nodes a certain number of times, i have allocated a limit of 50 , this way transferring of packets will come to an end after the limit and the whole network wouldn’t collapse. For this i have implemented a dictionary called transfer_limit: where the key is the node name and it will correspond  to the number of times of transmit of that nodes packet, which will increment each time that node’s packet is sent , and would not transfer if it has reached 50.

## ***Node failures implement:***

A node broadcasts its packet every one second, so this is used as a heartbeat msg, and if i do not receive the node after 3 heart-beats, then an error msg is made and send to the all other neighboring nodes which in turn will transfer the error msg to its neighboring nodes excluding the one it received the msg from.To do this i have a dictionary called heartbeat: whose keys are the neighboring nodes and the value corresponding to them will increment each time it do not receive a heartbeat , and it will create and send an error msg after it have a certain number  missed consecutive heartbeats . and this msg will have a TTL value field, which is checked while transferring  this message ,which have a  limit of 10 after which there wont be any more transferring of this error message. All the nodes which will eventually receive this error msg will have their known_nodes_link dictionary updated from which they calculate dijkstras algorithm .

all the nodes will also have the transfer_limit of the node which are killed to be set to zero, so once the node rejoins , transferring will take place up up to the set limit again.

## ***Features***:

1.Extract data from the text file and represent it as a dictionary

2.Send broadcast messages to neighboring nodes

3.Listens to incoming messages

4.Known_nodes_link get updated according to the messages received (Global view of the topology)

5.Prints least cost path to all other nodes every 30 seconds

6.Checks for death of other nodes by keeping the count of the heartbeat messages (broadcast messages received) and updates known_nodes_link accordingly

## ***Data structure:***

### **Link state data structure:**

I, have represented my link state information as a dictionary,I think it would be best if i can give an example.

node A which has the ConfigA.txt of

```
A 5000
2
B 6.5 5001
F 2.2 5005
```

is represented as

```
{“router”: ”A”  ,  ”port”: 5000  , “transporter”: ”A” , “type”: 0 , “TTL”:0 ,
“no_of_neighbours”:2 , “B”:(6.5,5001),”F”:(2.2,5005) }
```

here the ”TTL” value do not have any significance as we are limiting transferring by another mechanism using transfer_limit dictionary as mentioned earlier above.

transporter is the node which send this packet to other nodes , “transporter” value gets updated as hops along the topology.

## Error Messages data structure:

I have represented my error message also as a dictionary which has a “Type” value of 1 to differentiate the type of message.

Error msg of NODE a would contain

```
{"type":1, "router”: ”A”, "transporter”: ”A”, "port”:5000, "TTL":0}
```

this message is propagated along the topology until the TTL value is 10, this error message tells that node A is killed and each node can update their known_nodes_list accordingly

## ***REFERENCE:***

I Have ONLY  referred  the idea of representing the nodes as a graph from the below link:

https://www.sanfoundry.com/python-program-implement-dijkstras-shortest-path-algorithm/