Surya Avinash Avala, z5096886
Report - Assignment 2

1.1 In addition you should submit a small report, report.pdf (no more than 3 pages) describing the program design and a brief description of how your system works.

Program Files:

1. graph.py - It is the file containg graph class
2. dijkstra.py - It is the file where dijkstra algorithm is applied to the graph
3. Lsr.py - Lsr as per assignment specifications
   - Link State advertisement, listening for packets, builds graph and runs dijkstra algo to determine LCP

graph and dijkstra are imported as header files in Lsr.py. So they have to be present in the same directory as Lsr.py while running Lsr.py.

```
python3 Lsr.py A 2000 config.txt
```

1.2 Describe the data structure used to represent the network topology and the link-state packet format.

1. Graph data structure is being used to represent network topology
2. Graph has two variables, Nodes and Edges
3. Nodes is a python list of all nodes present in the topology. Example: [‘A’,’B’,’C’,’D’]
4. Edges is a dictionary with keys being the tuple of nodes of a link and value being the cost of the node. Example: E[('A','B')] = 2
5. Link state packet originating at node ‘A’ is of following format. Example: my_lsr[‘A’] = [('A','B',2),('A','C',5),('A','D',1)] if B,C,D are neighbours of A.

1.3 Comment on how your program deals with node failures and restricts excessive link-state broadcasts.

Link-state broadcasts are restricted as follows:

1. Packets are not forwarded to the Node they are coming from.
2. I forward the packets to all my neighbours for the first time when I receive them
3. I remember all the packets I have received so far in a dictionary named lsr
4. If I receive the same packet as before, I don’t forward it.

Deal with node failures:

1. I have a timer of 5sec (approx. 5 lsr packets from my neighbour) before I determine they are dead.
2. If any of my neighbours are dead then I update my link state advertisement appropriately and reset my network topology graph.
3. My neighbours will update their topology accordingly.

1.4 Also discuss any design tradeoffs considered and made. Describe possible improvements and extensions to your program and indicate how you could realise them.

Link state advertisements could have been dealt with better with the use of sequence numbers and acknowledgements.
Remembering the packets will get complicated easily and I am only forwarding the packets once, so if the nodes are not started at the same, there is chance of losing some packets. Forwarding the packets until acks are received would be a better way of dealing with lost packets.

1.5 If your program does not work under any particular circumstances please report this here.

If a node dies then its neighbours are able to detect it correctly. But nodes other than the dead ones neighbour have trouble finding it out.
But I think it is still not a issue because being a router all you care about it which of your neighbours to forward a particular packet to.
For example: In a topology A->B->C. If A is dead then B knows it. If C gets a packet with destination as A, it forwards the packet to B and B drops it knowing that A is dead.

1.6 Also indicate any segments of code that you have borrowed from the Web or other books.

None. The code has been written from scratch by me. Progress of it can be seen at the following bitbucket repo:
git@bitbucket.org:saavala2/network.git
The repo will be made public after one week from the assignment due date.