Bohui Pan z5014389

COMP3331 Assignment 2 Report

My program implements all features have been mentioned on specification and uses multi-threading to maintain my program. The main threads including Receiver, Flooding, Alive Checking and least Cost path. And all of them would be running automatically until close the program or keyboard interrupt.

> Receiver: keep receiving the different type of packets(link-state packet and heart-beat packet) from neighbour nodes and updated the graph.
>
> Flooding: Send link-state packet to their neighbour nodes every second periodically.
>
> Alive Checking: send heart-beat packet to their neighbour every 0.3 second periodically to detect whether their neighbour is dead or not.
>
> Shortest Path: Find the least-cost path to each destination node along with the cost of this path using Dijkstra's algorithm every 30 secs.

The program uses dictionary to store each neighbours and their port. And it also two-directional dictionary to store the edge between two nodes and their costs. For example: the cost between A and B is 5.0, edges[A][B] = 5.0 which is convenient to find and obtain the information.

For each link-state packet I convert all the information of current graph(ie: every edges in the graph including both direction like A B 5.0, B A 5.0) into string and put in the packet which is be able to easily tell other nodes how the graphs have this node known already and quickly broadcasted.

For detecting the dead node, the program would keep sending heart-beat packet to their neighbour nodes every 0.3 second. And if the program haven't received any heart-beat packet from the same neighbour nodes for 3 second, that neighbour node would be counted as dead. Then it will delete that node in the graph and relative edges, also broadcast a notification to other nodes. And once the other nodes receive this notification, they will update their graph and prepare the new link-state packet. Since finding the least cost path is basing on the graph, the least cost path to others node would update afterwards.

The program basically uses two conditions to restrict the excessive like-state broadcasts: 1. Stop broadcasting when received a linked-state packets that sent from itself

2. Each received packet would be stored into a set without duplicate. And it would stop broadcasting that packet when observer that packet has been stored already.

For the improvement, I think we can use a better broadcast way like MST to improve the programme and allow it to be more efficient.