COMP9331 Assignment Report

Yuhan He z5224997

2019 Term2

1 Discussion of implementation

In order to implement the link state routing protocol, I used multi-threading strategy. There are five threads are used. One for sending link-state packet, one for receiving link-state packet, one for checking the status of router's neighbours, one for calculating shortest path between routers using Dijkstra algorithm and one for recording heartbeat message.

I have successfully implemented the following features:

- Routers are able to send(receive) link-state packets to(from) its direct neighbours.
- Routers are able to broadcast link-state packets to all the other routers in the network. Based on the content of packets to calculate shortest cost using Dijkstra algorithm.
- Routers are able to detect their neighbours' status, and then adjust their broadcasting path according to the status of neighbours.
- Routers are able to deals with excessive link-state broadcasts which they are not going to broadcast.

Considering there might have a concurrency issue when the treads accessed the shared data in the process, I acquire lock every time before the threads access into the critical region.

2 Data structure of representation of network topology and link-state packet format

The data structure used to represent the network is a dictionary. The key is a tuple stored two associated routers, the value is the cost between them.

The data structure used to represent the link-state packet is a string. An example of packet format is shown in *Figure 2*. The first four characters means that

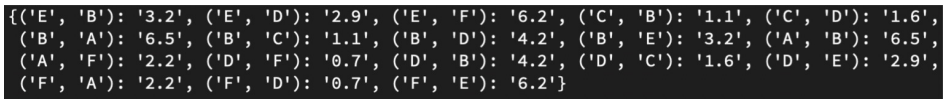

Figure 1: Dictionary of routers and cost

the sending router and destination router, followed by two characters which means the original sending/destination router of this packet(Packets can be broadcast between routers). Then the packet contains the direct neighbours information of sending router and corresponding cost. The last character is the length of packet which helps to determine the integrity of data.

Figure 2: Router A receives a link-state packet sending from router F

3 Dealing with node failures

To deal with the node failures, I keep sending the heartbeat messages from a router to its direct neighbours. For example, router A is sending the heartbeat messages to its direct neighbours every 0.3 seconds. If router A's neighbours did not receive this heartbeat message for 0.4 seconds, router A will be considered that it is turned off. And a Boolean will be set to 1 denoting the status of router A has been changed. The cost between A and its neighbours will be set as -1, in my case the negative value of cost indicating the infinity and will not participate in Dijkstra algorithm for further calculations.

4 Restricting excessive link-state broadcasts

For every broadcast case, the only change of the link-state packet is the sending/destination router in the first four characters. Which means the parts from original sending/destination routers before the length of packet is unchanged in every packet. I created a tuple with four elements under iteration: 1. current router id, 2. one of router's neighbour id, 3. original sending router, 4. original destination router. I also created a dictionary to record the appearance of tag-Tuple. The following is the pseudo-code of this procedure with current router A:

```
metaData = unchanged parts in every packet
for i is the neighbour of router A do
    tagTuple = (current router id, i's id, original sending router, original
     destination router)
    acquire lock before accssing shared data
    if i just broadcast the packet then
        release lock
        continue
    end
    else if tagTuple is not in tagDic then
        tagDic[tagTuple] = (metaData, timeStamp)
        release lock
    end
    else if tagTuple is in tagDic and the status of i has not been changed
     then
        release lock
        continue
    end
    else
        tagDic[tagTuple] = (metaData, timeStamp)
        set status of i to 0
        release lock
    end
    current router broadcasts packet
end
```

Algorithm 1: Broadcasting and restrict the excessive link-state packet

5 Design trade-offs and improvements

The design trade-off had been made is the detection of heartbeat messages. The checking time interval is 0.3 seconds, if a router restarts within 0.3 seconds, there will be no awareness of its changed status. Therefore, their neighbours will not re-transmit the packets to them because of the detected unchanged status. Another design trade-off is that the same packet will only broadcast once, based on the assumption I made that if the neighbours of a router is on, they will receive packet. I think the format of the packet content and the way to restrict the massive packets is special. And I also check the integrity of the packet by checking the length of packet at sending side and receiving side. The improvement could be made is adding acknowledgements and sequence numbers to guarantee the detection of packets loss.