COMP9331 Computer Networks and Applications – Assignment 2

Report

Jatin Gupta

Z5240221

1. Brief Discussion: How LSR protocol is implemented

My implementation is done using Python 3 language. My implementation of the protocol when a router starts, it binds itself on the port provided by the text file and keeps listening on the port. It will also create a message which will be sent to all the neighbour routers which will notify information about the sender.

Features Implemented:

- Link State protocol (Phase 1).
- Printing all nodes and path via Dijkstra Algorithm.
- When a node gets down, everyone updates their routing table.
- Updated Dijkstra Algorithm when a node is down.
- Make a router from dead to alive state and update the Dijkstra Algorithm and give out old answers.

2. Data Structure used for different tasks

Router Class has the following fields

- Name: string, name of the router
- Port: string
- Message: object of the Message class
- List of neighbours: List of neighbour class
- Global routers timestamp: dict, a time value which will tell the last message received from the key router
- Global routers: dict, key as router name and list of all its neighbours

Message Class contains an object of the Message class. This object contains:

- Sender name: string
- Sender port: string
- Sender neighbours: List of neighbours object
- Timestamp: when this message was originated: Float epoch value
- Last sender section which will be updated on every hop: string

Neighbour Class contains

- Name: string, Name of the neighbour
- Port: string
- Distance: String

Graph Class contains

- Global routers: dict, a copy of the global routers from router object
- Graph: dict, with key as router name and value contains a list of edge objects

Edge Class contains

- Start: string, start node name
- End: string, end node name
- Weight: float, weight between start and end

3. Restrict excessive link-state broadcasts

As soon as the router starts it starts sending its message packet to the neighbours, and this process will be done by all the routers. Also, each router will listen to all the incoming packets. All messages checked if they have sent this packet via a timestamp of the message and **restrict delivery of the message**. Also, a check is done before forwarding this packet to its neighbours that if the last sender was one of my neighbours, I will not send this packet to that neighbour. This prevents excessive link-state broadcasts.

4. Node Failures

I maintain a timestamp for all the routers to check if they are dead or alive. I update the timestamp each time I receive a packet corresponding to the sender's name and the timestamp which was in the message. I check my neighbours after every 3 seconds that are they alive? And I check the global topology after every 12 seconds. I update the global router topology data structure accordingly.

5. Design trade-off

I should have saved port in integer and distance also in float, but for sake of simplicity I have kept them as string, they can be changed to number/decimal number when required.

Messages are sent via Pickle library of python. We could use JSON for better cross language compatibility. Pickle only works for python to python files.

6. Special about your implementation

I have used objects in the complete program and only used dictionaries when possible. I have used as descriptive names as possible and tried to follow the correct code conventions as much as I could. Also, I have declared all the constants timers on the top which will allow any user to test the program to change the values at one place and see the updates.

7. Possible extensions and improvements

I feel that there is little more than usual code in one file and hence it should be separated into modules. And I should import those modules into the main program to work more efficiently. This will make the code scalable and maintainable.

The values should be in their respective data types, for example port is being saved as string, it should be saved as integer.

Using JSON rather than Pickle for multi-programming language support, which will allow JAVA nodes to be compatible with python nodes, currently this will work with only python nodes.

8. Segment of code borrowed

I have not used any segment of code from the Web or other books. I have studies code provided by the professor in its UDP client and server. That code is standard.