Routy - a routing network

Nihad Ibrahimli, nihadi@kth.se

September 21, 2017

Introduction

The task is implementing a link-state routing protocol. The protocol is used for OSPF, which is the most used routing protocol. The goal for this task is to be able to describe the structure of a link-state routing protocol, describe how a consistent view is maintained and reflect on the problems related to network failures.

A routing process should consult its routing table and should be able to find the best route for sending the message. If the message arrives at the destination it is printed on the screen, if doesn't it is thrown away.

In order to implement this routing protocol, many list function need to be used for searching, finding and replacing nodes in the list.

Main problems and solutions

The first thing is to write the map module. It should be functional for updating the map and also finding the nodes connected to a given node. Four functions needed to be implemented in this module.

- Update(Node, Links, Map) function is to update the directional links of the node. To update the links for the given Node, we need first to find the Node in the map. For this I used keysearch function which searches for the map, looks at only first index. Because the node is the first element for the Tuples inside Map.

  FoundTuple = lists:keysearch(Node, 1, Map)

  The second important function used in this function is keyreplace which replace the Links for the node in the index 1 in Map.

- Reachable(Node, Map) is for finding the links for which the given Node has directional link to them.
  For this function keysearch is used as described above.
- All_nodes(Map) returns the list of all nodes, the ones also without outgoing links. The foldl function of list library is used here. The next thing to do is to make the list unique list by converting it to set and back to list.

The next algorithm to be implemented is Dijkstra which computes a routing table. In the algorithm a sorted list is used when we calculate a new routing table. The list is sorted based on the length of the path.

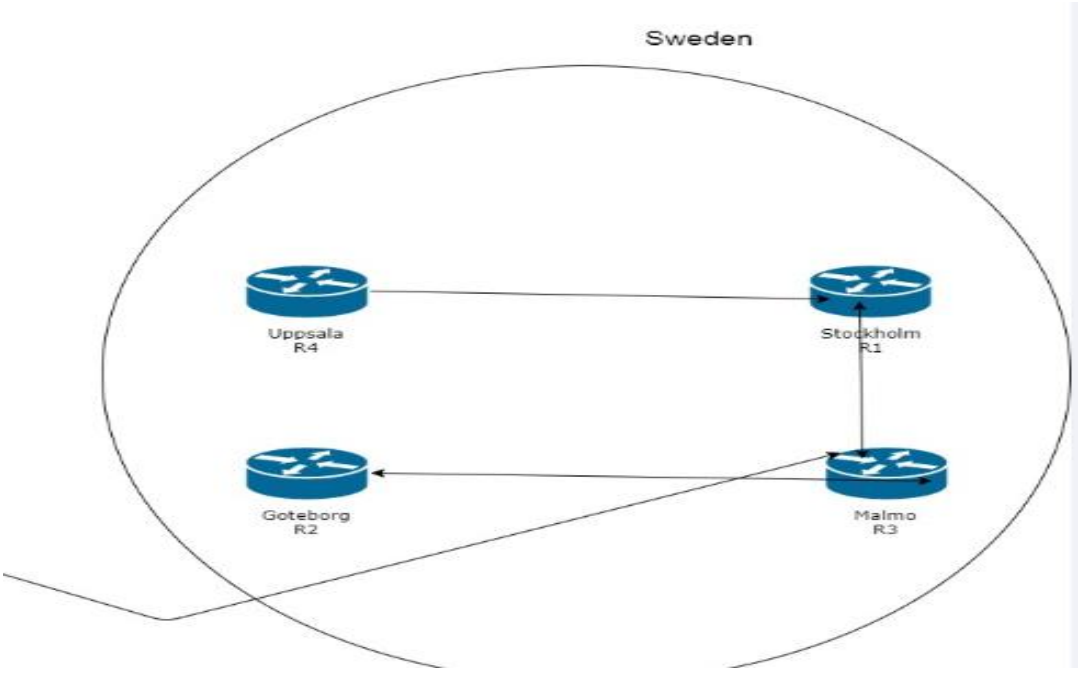
- Entry(Node, Sorted) return the length of the shortest path to the node.
  As the other functions above the keysearch used here to find the entry.
- Update(Node, N, Gateway, Sorted) update the list Sorted given the information that Node can be reached in N hops using Gateway.
  For which we need to find the cost for entry. It is important to decide update or not.
- Iteration part is the most important one. As stated in the assignment description, 3 conditions of iteration function do exist. The first iteration function will be run if there is not any entry in sorted list. The second case is the first entry has infinite path. Because the list is sorted it means that the entries after that entry are also infinite. Otherwise the sorted list is separated to head and tail parts and iterated through it.

A router will also need to keep track of a set of interfaces and history. History is for throwing away the messages is already sent. For that counter is used. After implementing history I tested it as below.

History=[{a,10}, {b,5}, {c,4}].
hist:update(a, 11, History).

Testing

For the testing purposes I built this topology. And I made sure that my routing is working properly.



Conclusion

Despite the fact that I learnt much from it, this assignment was very hard. The most difficult thing was to understand how to use list functions, especially the foldI function. The task was very interesting, and especially at the end to see the result of the work was interesting. I think that it would be better to increase the difficulty of the task steadily. T first assignment was actually copy and paste and just editing some parts. Moving from the assignment with that level to this Routy was a little bit shocking for me. In conclusion, I want to point out that I am excited about next assignments.