Routy : a small routing protocol

Nikhil Dattatreya Nadig

September 20, 2017

1 Introduction

The objective of this exercise was to implement a link-state routing protocol in Erlang.

The basic concept of the routing protocol is to create a map of connectivity of the network showing the connections among the nodes. The link-state routing protocol widely used for OSPF protocol for Internet routers. Connection with several countries depicted that communicate with each other. Each router is defined with a unique name of a city that will be connected to other cities locally. The router can communicate directly with other nodes and with nodes that are not connected directly as well. The routing process constructs a table and determines which the proper gateway is for a node to communicate with the other. The system rejects messages for which there is no a logical path to reach the receiver.

2 Main problems and solutions

Following are the main problems that were encountered:

- Representation of a map of nodes and links.
- Implementation of Dijkstra's Algorithm in Erlang.

First, a function to represent all the direct links to a nodes is created. The map module constructs the map which represents the nodes in the network and it's direct links of every node. Here, the map constitutes a list of cities and it's corresponding links to other cities.

The implementation of Dijkstra's algorithm in Erlang was particularly difficult. This algorithm is key to the protocol's functioning so it was time consuming to check for all the edge cases and making sure that the algorithm did not fail for any case.

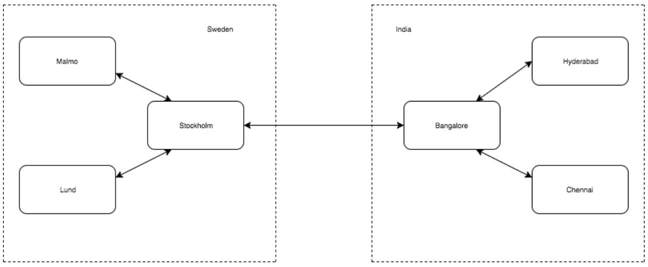

Figure 1: Topology of the Nodes.

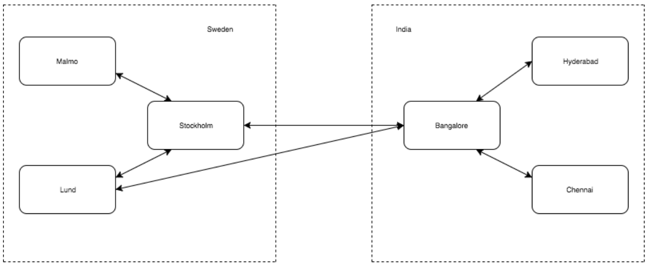

Figure 2: A new connection between Bangalore and Lund has been added.

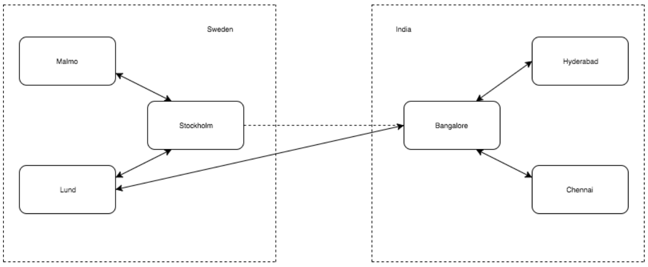

Figure 3: Connection between Bangalore and Stockholm has been removed. However, this does not affect the routing of messages.

3 Evaluation

Figure 1 represents the topology of the nodes. Here, there is a cluster of two countries each having 3 nodes in it's cluster. The two clusters are connected through Stockholm and Bangalore. All the nodes are connected to each other bidirectionally, i.e. for example, a message can be sent from Bangalore to Chennai and Chennai to Bangalore. All these nodes are created in the same computer in different shells. Messages here reach from one shell to the other mimicking the passing of message through a network
When a node in India wants to send a message to a node in Sweden, the message has to be routed through Bangalore and Stockholm. These are the only nodes connecting the two clusters. If this connection was somehow broken, the communication between these two cluster would not have been possible. In Figure 2, we have added a connection between Lund and Bangalore. Now, if a message had to sent from Lund to Bangalore, the message would directly reach Bangalore. Previously, if a message had be delivered from Lund to Bangalore, it had to route through Stockholm.
In Figure 3, the connection between Bangalore and Stockholm has been removed. This however does not affect the connection of the network. The messages that were routed through Stockholm to India, now are routed through Lund. Dijkstra's algorithm ensures the shortest path to the destination is taken.

Using the same topology, message passing through a network to a different computer can be done. This will be demonstrated in the seminar. When a node is closed, the network can still find a different route to send the message to it's destination.

4 Conclusions

Link-state protocols and implementation of Dijkstra's algorithm in Erlang was learned through this exercise.