Report 2: Routy

Yanghang Zeng

October 13, 2022

1 Introduction

This assignemnt is about the implementation of router using Erlang, running link state routing protocol OSPF. It also involves the procedure of using Erlang's distributed programming feature. The main work is to build several modules that support all basic functions of a router. The main topic related to distributed system is the routing among network nodes. It is important to enable communication within a huge network and to excahnge link state inforamtion and then build a optimal routing table.

2 Main problems and solutions

To implement the router, several parts is needed.

The map module is used to represent the connection between nodes, i.e., which node is connected to which. It is a abstraction of the global topology, which will be used to calculate the shorted path. The intf module is used to store the interfaces of a router. The hist module is used to keep track of messages that a router have seen, to avoid cyclic path. The dijkstra module is the heart algorithm of OSPF protocol. Each node uses its knowledge of global topology to calculate the shorted path to other nodes, results in a routing table which indicates which gateway to go. The router module combines all other modules and implement the functions of a router, including link state message advertising.

3 Evaluation

To test the implementation, firstly some routers are created. And then assign interfaces to each router to form a topology. Next, routers broadcast their link state to the network and the other routers update their knowledge about the global topology. Finally, each router populates its routing table. The topology is as in figure 1.

The result of the test is as follows. In figure 2, three routers are created and interfaces are assigned. Three router forms a closed ring. In figure

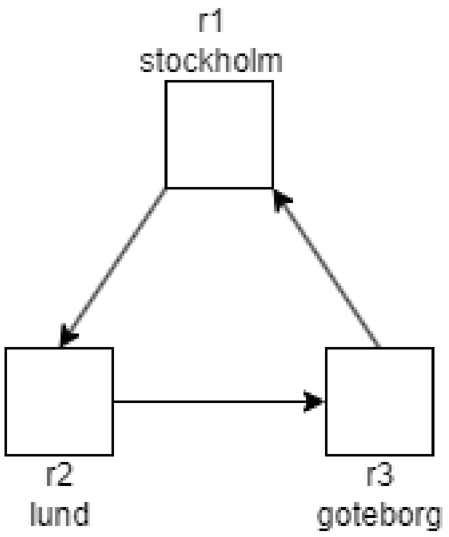
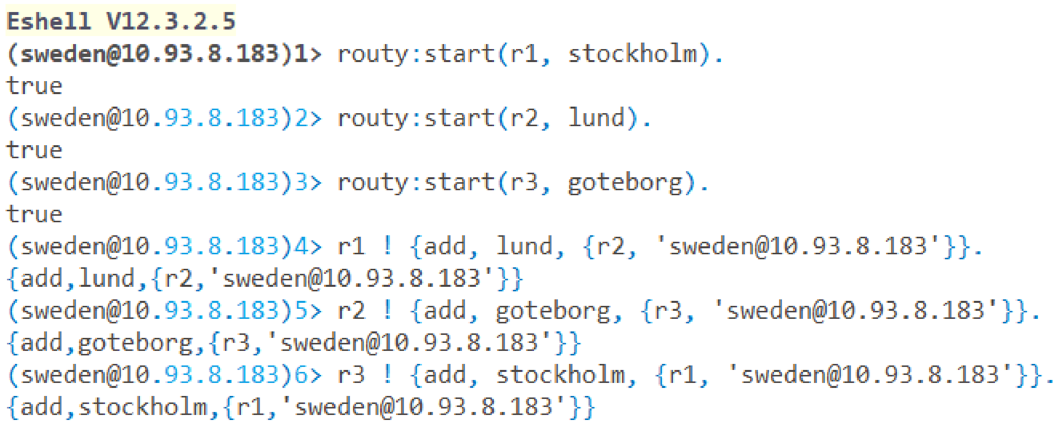


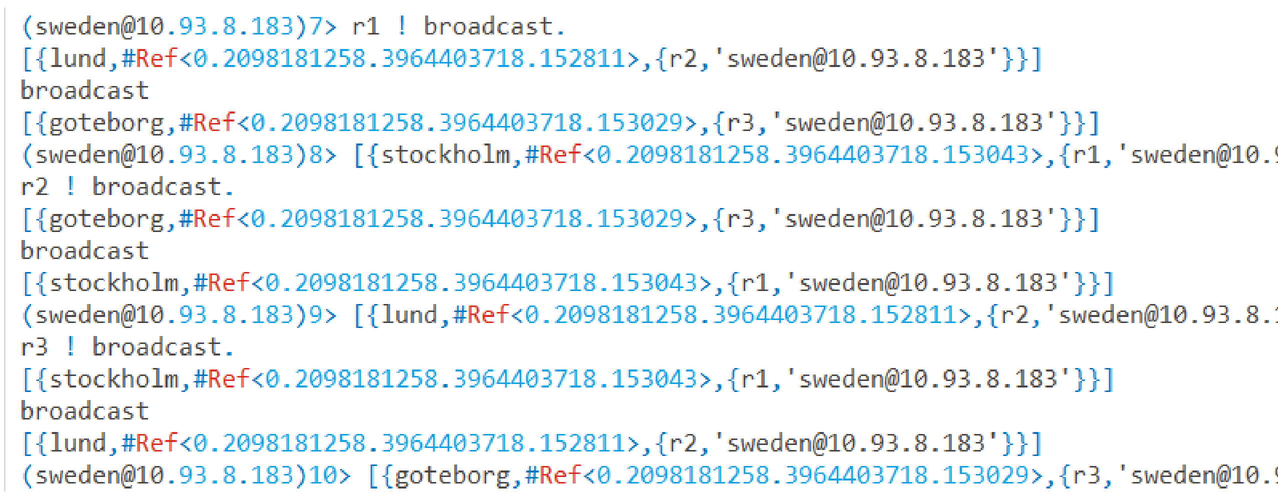
Figure 1: Figure 1

3every router broadcast its link state information to its neighbors. By doing this, each router can update its map of the network, thus being able to calculate the routing table. In figure 4, after the routing table is completed, information on all routers is printed to verify the correctness. From the result, we can easily verify every router has the correct table. In figure 5, stockholm sent a message to goteborg, the message was forwarded to lund then reach the goteborg, as it is in figure 1. So the implementation is successful in this case.



```
Eshell V12.3.2.5
(sweden@10.93.8.183)1> routy:start(r1, stockholm).
true
(sweden@10.93.8.183)2> routy:start(r2, lund).
true
(sweden@10.93.8.183)3> routy:start(r3, goteborg).
true
(sweden@10.93.8.183)4> r1 ! {add, lund, {r2, 'sweden@10.93.8.183'}}.
{add,lund,{r2,'sweden@10.93.8.183'}}
(sweden@10.93.8.183)5> r2 ! {add, goteborg, {r3, 'sweden@10.93.8.183'}}.
{add,goteborg,{r3,'sweden@10.93.8.183'}}
(sweden@10.93.8.183)6> r3 ! {add, stockholm, {r1, 'sweden@10.93.8.183'}}.
{add,stockholm,{r1,'sweden@10.93.8.183'}}
```

Figure 2: Figure 2

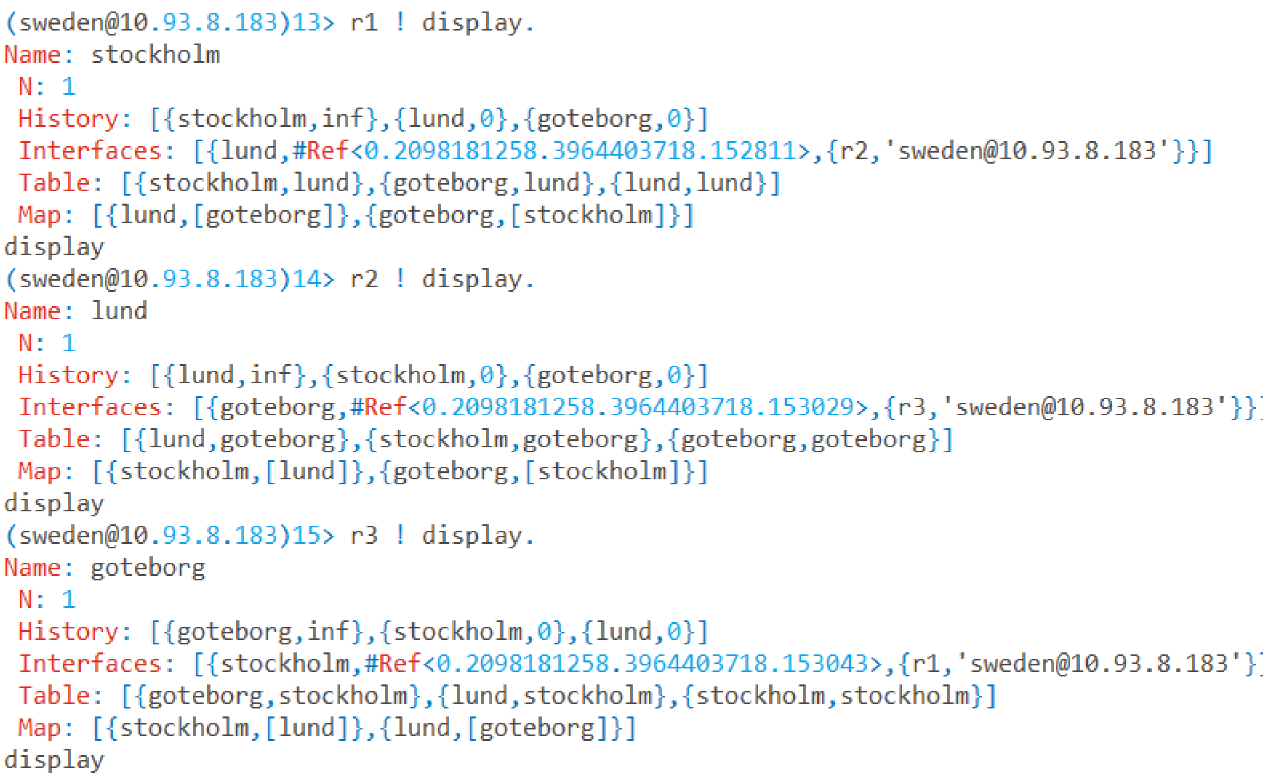


```
(sweden@10.93.8.183)7> r1 ! broadcast.
[{lund,#Ref<0.2098181258.3964403718.152811>,{r2,'sweden@10.93.8.183'}}]
broadcast
[{goteborg,#Ref<0.2098181258.3964403718.153029>,{r3,'sweden@10.93.8.183'}}]
(sweden@10.93.8.183)8> [{stockholm,#Ref<0.2098181258.3964403718.153043>,{r1,'sweden@10.9
r2 ! broadcast.
[{goteborg,#Ref<0.2098181258.3964403718.153029>,{r3,'sweden@10.93.8.183'}}]
broadcast
[{stockholm,#Ref<0.2098181258.3964403718.153043>,{r1,'sweden@10.93.8.183'}}]
(sweden@10.93.8.183)9> [{lund,#Ref<0.2098181258.3964403718.152811>,{r2,'sweden@10.93.8.1
r3 ! broadcast.
[{stockholm,#Ref<0.2098181258.3964403718.153043>,{r1,'sweden@10.93.8.183'}}]
broadcast
[{lund,#Ref<0.2098181258.3964403718.152811>,{r2,'sweden@10.93.8.183'}}]
(sweden@10.93.8.183)10> [{goteborg,#Ref<0.2098181258.3964403718.153029>,{r3,'sweden@10.9
```

Figure 3: Figure 3

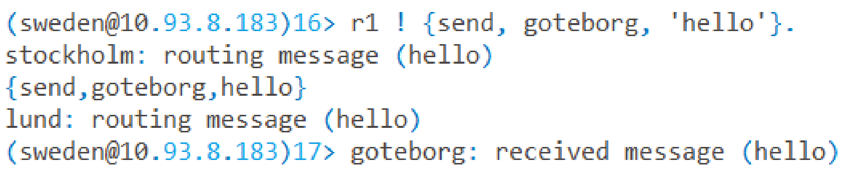
4 Conclusions

Through this assignment I understand the procedure to build a simple routing system, by implementing functions of a router and exchange link state information to calculate a optimal routing table. It is useful for communication within a large network.



```
(sweden@10.93.8.183)13> r1 ! display.
Name: stockholm
 N: 1
 History: [{stockholm,inf},{lund,0},{goteborg,0}]
 Interfaces: [{lund,#Ref<0.2098181258.3964403718.152811>,{r2,'sweden@10.93.8.183'}}]
 Table: [{stockholm,lund},{goteborg,lund},{lund,lund}]
 Map: [{lund,[goteborg]},{goteborg,[stockholm]}]
display
(sweden@10.93.8.183)14> r2 ! display.
Name: lund
 N: 1
 History: [{lund,inf},{stockholm,0},{goteborg,0}]
 Interfaces: [{goteborg,#Ref<0.2098181258.3964403718.153029>,{r3,'sweden@10.93.8.183'}}
 Table: [{lund,goteborg},{stockholm,goteborg},{goteborg,goteborg}]
 Map: [{stockholm,[lund]},{goteborg,[stockholm]}]
display
(sweden@10.93.8.183)15> r3 ! display.
Name: goteborg
 N: 1
 History: [{goteborg,inf},{stockholm,0},{lund,0}]
 Interfaces: [{stockholm,#Ref<0.2098181258.3964403718.153043>,{r1,'sweden@10.93.8.183'}
 Table: [{goteborg,stockholm},{lund,stockholm},{stockholm,stockholm}]
 Map: [{stockholm,[lund]},{lund,[goteborg]}]
display
```

Figure 4: Figure 4



```
(sweden@10.93.8.183)16> r1 ! {send, goteborg, 'hello'}.
stockholm: routing message (hello)
{send,goteborg,hello}
lund: routing message (hello)
(sweden@10.93.8.183)17> goteborg: received message (hello)
```

Figure 5: Figure 5