Report 2: Routy

Zidi Chen

September 17, 2020

1 Introduction

In this assignment, I have implemented a very basic Erlang program to manipulate a link-state routing protocol. The core of the algorithm in the protocol is the Dijkstra algorithm which is implemented in an iterative way in this program.

2 Main problems and solutions

The first difficulty for me is to get familiar with lists in erlang. I actually felt a little bit confused about tuples and lists. But I managed to solve the tasks and implemented the functions as required. I found that the best way to learn is to keep trying the lists function in the command prompt instead of reading, which really works.

The second difficulty is to understand Dijkstra algorithm and the iteration function. I found the best way to understand is to just draw it down with a few input examples, and then follow the process in the functions. Especially when the functions are iterative, trying to follow with a few rounds. This gives me a better understanding.

3 Evaluation

Test 1: I had three routers in the same terminal and connected them as a ring.

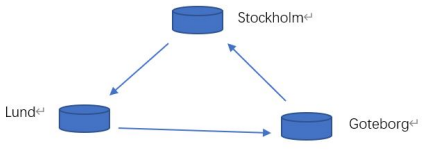

Figure 1: The structure of test 1

It basically works, the status of each router are shown below.

```
r1 status:
Name: stockholm
N: 1
History: [{goteborg,0},{lund,0},{stockholm,inf}]
Interfaces: [{lund,#Ref<0.274652017.3652976644.79778>
,{r2,'sweden@192.168.3.2'}}]
Table: [{stockholm,lund},{goteborg,lund},{lund,lund}]

Map: [{lund,[goteborg]},{goteborg,[stockholm]}]
ok
(sweden@192.168.3.2)14> routy:status(r2).
r2 status:
Name: lund
N: 1
History: [{goteborg,0},{stockholm,0},{lund,inf}]
Interfaces: [{goteborg,#Ref<0.274652017.3652976644.79
783>,{r3,'sweden@192.168.3.2'}}]
Table: [{lund,goteborg},{stockholm,goteborg},{gotebor
g,goteborg}]
Map: [{stockholm,[lund]},{goteborg,[stockholm]}]
ok
(sweden@192.168.3.2)15> routy:status(r3).
r3 status:
Name: goteborg
N: 1
History: [{lund,0},{stockholm,0},{goteborg,inf}]
Interfaces: [{stockholm,#Ref<0.274652017.3652976644.7
9788>,{r1,'sweden@192.168.3.2'}}]
Table: [{goteborg,stockholm},{lund,stockholm},{stockh
olm,stockholm}]
Map: [{stockholm,[lund]},{lund,[goteborg]}]
ok
```

Figure 2: The status of routers in test 1

A message can be sent from Stockholm to Goteborg.

```
(sweden@192.168.3.2)13> r1!{send,goteborg,'hi'}.
stockholm: routing message (hi)
lund: routing message (hi)
{send,goteborg,hi}
goteborg: received message hi
```

Figure 3: Sending message in test 1

Test 2: In this test, I extended the first test. I added the connection between Stockholm and Goteborg. Now the structure looks likes figure 4.

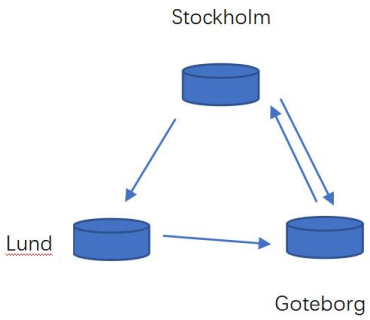

Figure 4: The structure of test 2

In this way, when Stockholm sent message to Goteborg, it directly went to Goteborg, instead of went to Lund for routing.

```
(sweden@192.168.3.2)22> r1!{send,goteborg,'hi'}.
stockholm: routing message (hi)
goteborg: received message hi
{send,goteborg,hi}
```

Figure 5: The result of test 2

This proves that Dijkstra algorithm ensures to route to the destination in the shortest way in the example.

In addition, when I removed the link of Stockholm and Lund, and the link from Stockholm to Goteborg, the message failed to send to Goteborg.

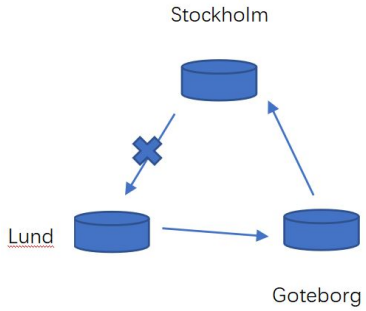

Figure 6: The structure of connection broken

This proves that the broken connection causes error in message routing in this example. Just do not forget to broadcast the update manually after remove the node.

Test 3: Here I created three routers which were connected in a ring in my computer as in China, and my colleague created three routers in his computer also in a ring as in Sweden. We tried to connect two rings with each other as the figure 7 shows.

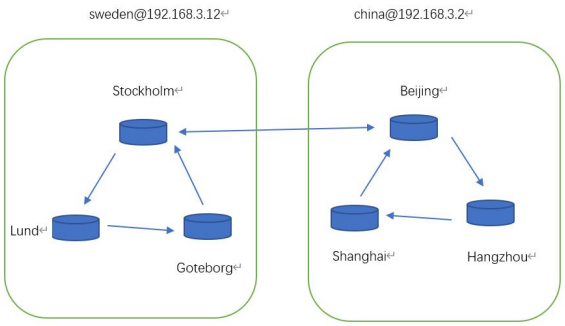

Figure 7: The structure of test 3

Then, we tried to send message from Stockholm to Shanghai. In this situation, Beijing and Hangzhou routed the message to the next destination.

```
(china@192.168.3.2)30> beijing: routing message (hello)
(china@192.168.3.2)30> hangzhou: routing message (hello)
(china@192.168.3.2)30> shanghai: received message hello
```

Figure 8: The result in test 3

4 Conclusions

In this assignment, I learned the basic structure of a routing protocol. In addition, I learned to know about lists and how to handle with tuples in Erlang. I also reviewed Dijkstra algorithm and got to know about link-state message.