Report 2: routy

Xiaoying Sun

September 17, 2020

1 Introduction

This report will describe the structure of a link-state routing protocol, OSPF, and how to implement it in Erlang.

2 Main problems and solutions

The main problems I had in the exercise included :

- How do routing table, network map, and interface work in OSPF?
- How to update the network map synchronously?
- What kind of link-state messages may appear and how to deal with it?

2.1 Data structures

Noun.	Data structure	Parameters
Node	automatic value	Below uses city's name
Map	$\{Node, [Node1, Node2...]\}$	Node directly accesses to Node1, Node2...
Table	$[\{des\_node, next\_jump\}]$	des_node, final destination $next\_jump$,forward to the next node
Intf	$\{Name, Ref, Pid\}$	$Name$, the symbolic name (london), Ref,a process reference Pid, a process identifier
Hist	$\{Node, old\}$ | $\{Node, N_{update\_times}\}$	old, message from $Name$ always old N_{update_times}, N times updated

Table 1: Data structures of the parameters

2.2 Differences between Network Map and Routing Table

- A network map provides an overview of all avaliable networks in a particular domain.
- A routing table is used to make the decision to which gateway a particular message must be forward to reach its target network.

3 Evaluation

3.1 Scenario 1

Scenario 1 consists of three routers: *beijing*, *henan* and *shanghai*. Among them, *beijing* and *shanghai* are directly connected to *henan*. Our aim was to verify if *beijing* could send messages to *shanghai* via *henan* and vice versa.

The setup commands can be found in the appendix of this report.

```
(china@192.168.1.0)26> beijing!{send, shanghai, "hello, shanghai"}.
beijing: Routing message ('hello, shanghai') from beijing to shanghai
{send,shanghai,"hello, shanghai"}
henan: Routing message ('hello, shanghai') from beijing to shanghai
shanghai: Received message 'hello, shanghai' from beijing
```

Figure 1: from henan to beijing

Figure 1 and 2 respectively show the terminal running interfaces of the router *beijing* and *shanghai* sending messages to each other via router *henan*.

```
(china@192.168.1.0)27> shanghai!{send, beijing, "hello, beijing"}.
shanghai: Routing message ('hello, beijing') from shanghai to beijing
{send,beijing,"hello, beijing"}
henan: Routing message ('hello, beijing') from shanghai to beijing
beijing: Received message 'hello, beijing' from shanghai
```

Figure 2: from henan to beijing

Scenario 1 implemented as expected. Figure 3 records all the information about the three routers in this local area network.

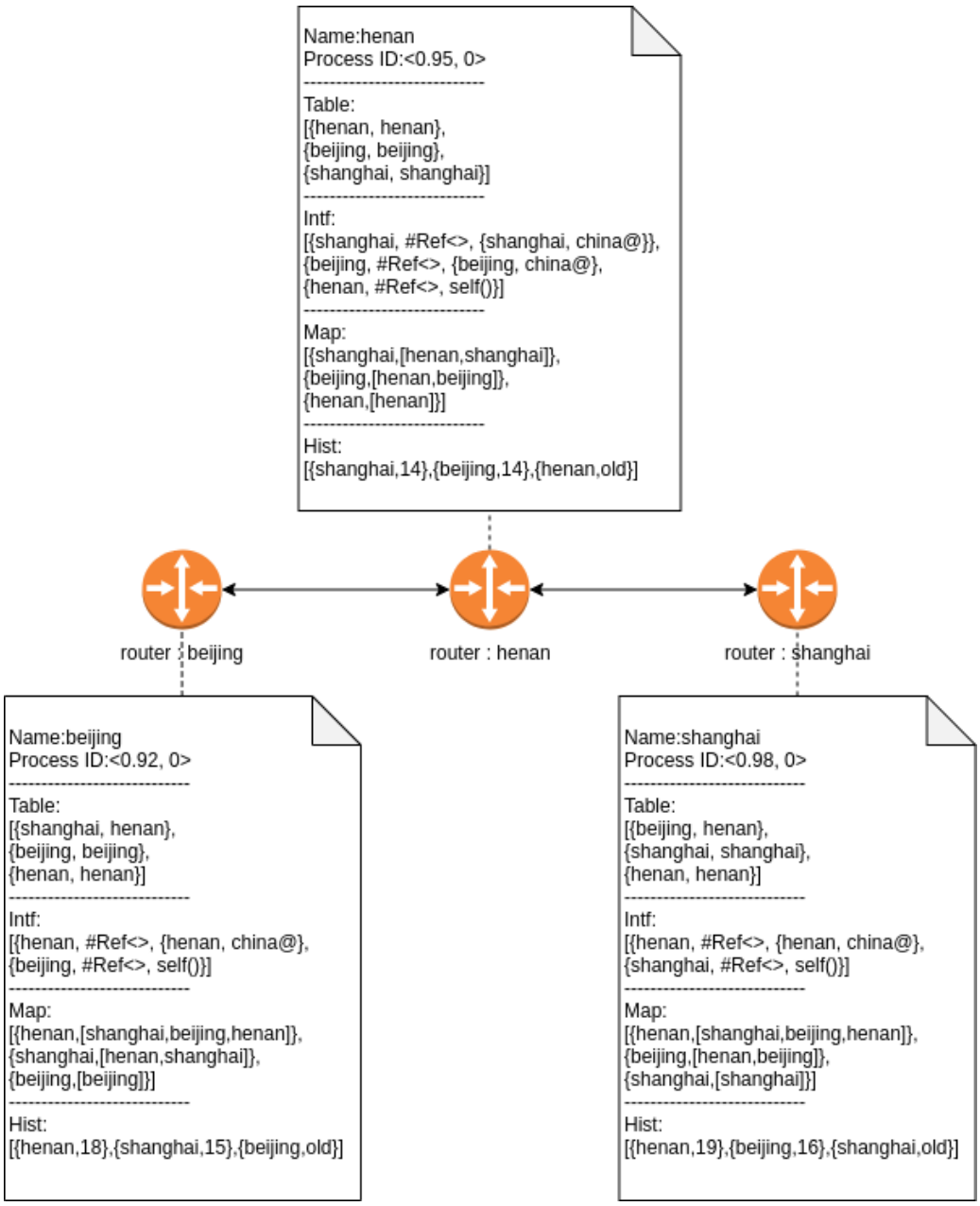

Figure 3: Beijing-Henan-Shanghai

3.2 Scenario 2

Scenario 2 is based on scenario 1. Our plan is to set up five routers on two different computers. On both computers, one router will play the role of a border gateway to its peer on the other computer. Our expectation in this

experiment was that *henan* could send messages to *goteborg* and *lund* by forwarding the message three times and vice versa.

It's very important to underline that two computers or two terminals must be on the same subnet to communicate directly. The simplest verification is to use their own IP address and subnet mask to perform an AND operation. If the results are the same, they belong to the same subnet, otherwise they are not.

The setup commands can be found in the appendix of this report.

As we can seen from figure 4, *henan* first sent the message whose destination was *goteborg* to the border router *beijing*, then *bejing* went to check out its own routing table and decided to forward it to the other border router *stcokholm*.

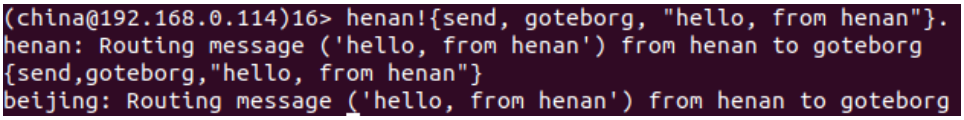

Figure 4: from henan to beijing

Figure 5 illustrates how *stockholm* received the message and directly sent it to its final destination *goteborg*.

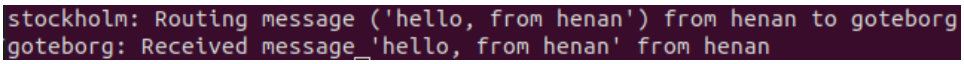

Figure 5: from stockholm to goteborg

As you might noticed, the link between router *goteborg* and *stockholm* was not a right two-way link. It was designed in this way deliberately to test the consequences if the final destination didn't exist in its routing table.

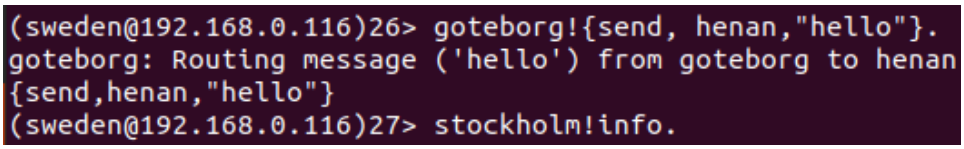

Figure 6: from goteborg to henan

The result turned out to be nothing happened. The node *sweden*@192.168.0.116 just ignored the wrong message and excuted the next command.

The following figure 7 records the details of all routers in both subnets.

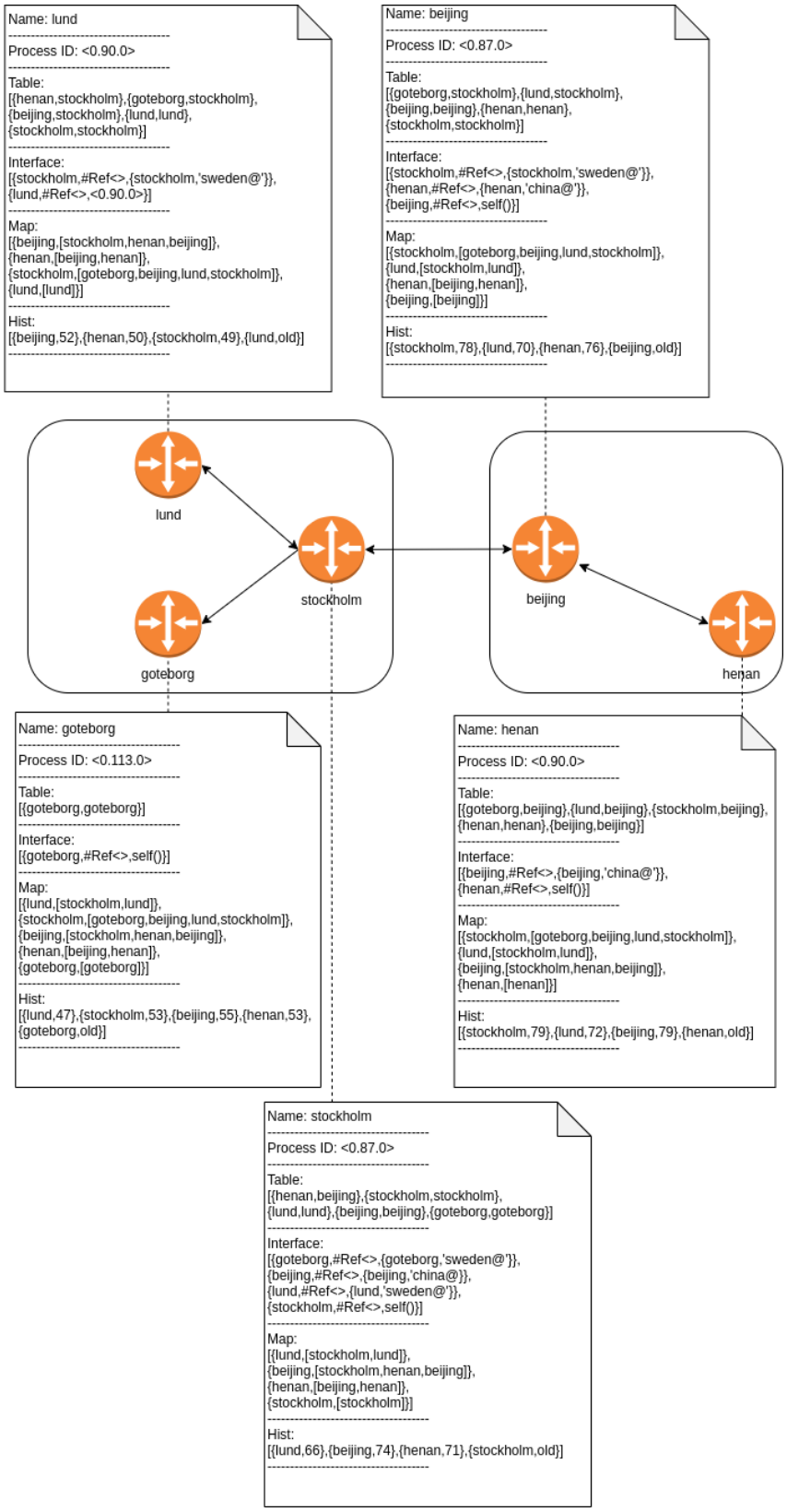

Figure 7: from henan to goteborg

4 Conclusions

From the above, we successfully had several routers up, running, and connected them together.

The further work direction should be,

- Figure out the differences between OSPF and other dynamic routing protocol and implement them in Erlang.
- Set a timer to report the error when not receiving response within the specified time.
- Change the manual addition of routers to automatic detection

5 APPENDIX

Scenario 1

```
% Start a  new command terminal on the second computer
%% and call it "sweden@<your-IP>"
erl -name china@192.168.0.114 -setcookie routy -connect_all false

routy:start(beijing).
routy:start(henan).
routy:start(shanghai).

beijing!{add, henan, {henan,'china@192.168.0.114'}}.
henan!{add, beijing, {beijing,'china@192.168.0.114'}}.
henan!{add, shanghai, {shanghai, 'china@192.168.0.114'}}.
shanghai!{add, henan, {henan, 'china@192.168.0.114'}}.


beijing!broadcast.
henan!broadcast.
shanghai!broadcast.

beijing!update.
henan!update.
shanghai!update.


beijing!{send, shanghai, "hello, shanghai"}.
shanghai!{send, beijing, "hello, beijing"}.
```

Scenario 2

```
% Start a  new command terminal on the second computer
%%  and call it "sweden@<your-IP>"
erl -name sweden@192.168.0.116 -setcookie routy -connect_all false

% The process of building a subnet is  similar to Scenario 1
% But pay attention to add the following interfaces to connect
%% to the other computer

beijing!{add, stockholm ,{stockholm, 'sweden@192.168.0.116'}}.
stockholm!{add, beijing ,{beijing, 'china@192.168.0.114'}}.
```