Report 2: Routy: A Small Routing Protocole

Luxiang Lin

September 18, 2024

1 Introduction

The goal of this assignment is to gain a deep understanding of the link-state routing protocol by implementing a server based on it. In Homework 2, we will focus on both the implementation and detailed explanation of the link-state protocol. This includes maintaining a consistent network view and reflecting on potential routing issues that may arise. To ensure that each routing path is optimized for the shortest distance, we used utilize Dijkstra's algorithm for constructing the routing table

2 Main problems and solutions

The main challenge I encountered during the assignment was implementing Dijkstra's algorithm, particularly with the logic in the `iterate()` function. This function is central to the algorithm but can be quite complex. To address this, I first identified three possible situations that might occur within the function. Then, I decided to refactor some procedures into separate methods. This approach aimed to clarify the logic of `iterate()` and make it easier to debug.

3 Evaluation

The first experiment I conducted involved setting up two nodes, Sweden and China, in two different terminals. Within each node, I created two routers: Stockholm and Lund for Sweden, and Beijing and Shanghai for China. I then established a route starting from Stockholm to Lund, with Lund acting as a gateway to Beijing, and finally from Beijing to Shanghai.

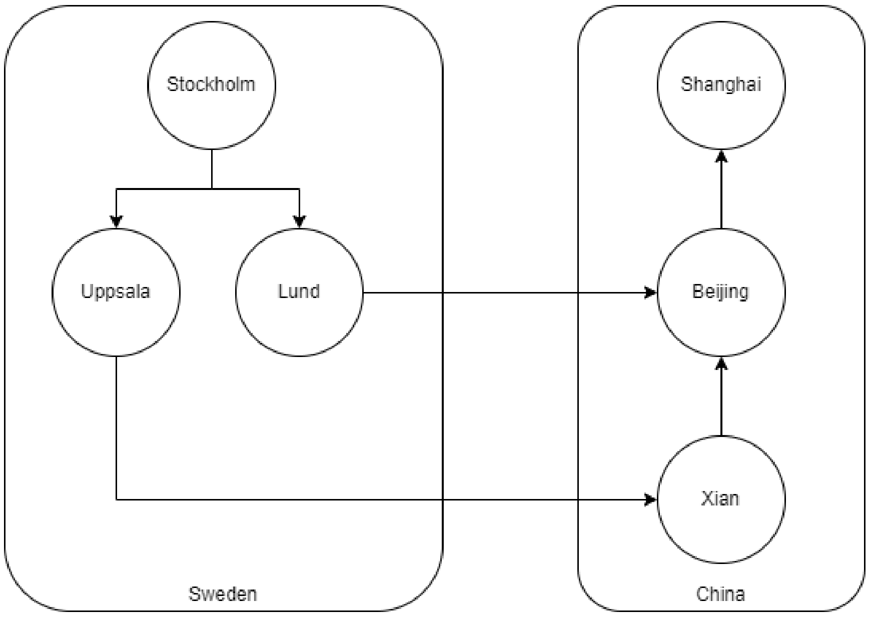
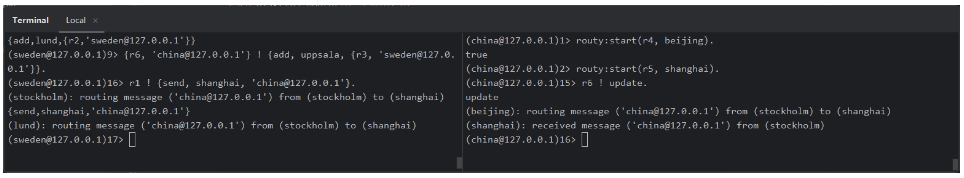


Figure 1

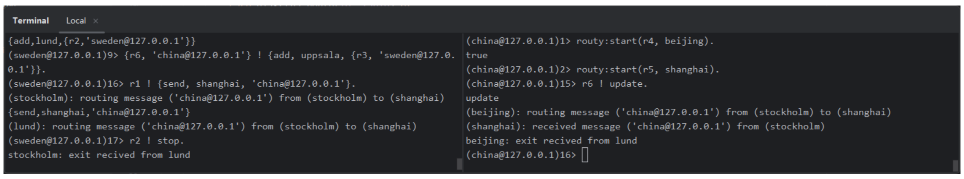


```
{add,lund,{r2,'sweden@127.0.0.1'}}
(sweden@127.0.0.1)9> {r6, 'china@127.0.0.1'} ! {add, uppsala, {r3, 'sweden@127.0.
0.1'}}.
(sweden@127.0.0.1)16> r1 ! {send, shanghai, 'china@127.0.0.1'}.
(stockholm): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
{send,shanghai,'china@127.0.0.1'}
(lund): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
(sweden@127.0.0.1)17>
```

```
(china@127.0.0.1)1> routy:start(r4, beijing).
true
(china@127.0.0.1)2> routy:start(r5, shanghai).
(china@127.0.0.1)15> r6 ! update.
update
(beijing): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
(shanghai): received message ('china@127.0.0.1') from (stockholm)
(china@127.0.0.1)16>
```

Figure 2

I have also carried out a test to see what would happen when one of the nodes in the route is shut down. In the test, I shut down the beijing router. The picture below shows the result: lund had received an exit message from beijing.

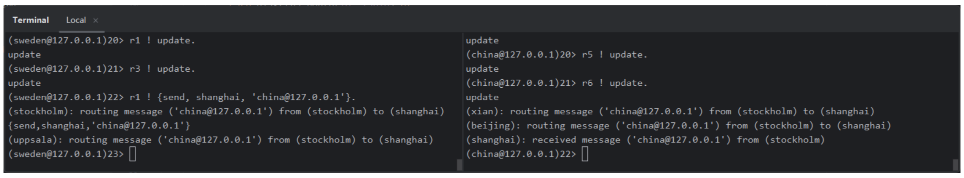


```
{add,lund,{r2,'sweden@127.0.0.1'}}
(sweden@127.0.0.1)9> {r6, 'china@127.0.0.1'} ! {add, uppsala, {r3, 'sweden@127.0.
0.1'}}.
(sweden@127.0.0.1)16> r1 ! {send, shanghai, 'china@127.0.0.1'}.
(stockholm): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
{send,shanghai,'china@127.0.0.1'}
(lund): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
(sweden@127.0.0.1)17> r2 ! stop.
stockholm: exit recived from lund
```

```
(china@127.0.0.1)1> routy:start(r4, beijing).
true
(china@127.0.0.1)2> routy:start(r5, shanghai).
(china@127.0.0.1)15> r6 ! update.
update
(beijing): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
(shanghai): received message ('china@127.0.0.1') from (stockholm)
beijing: exit recived from lund
(china@127.0.0.1)16>
```

Figure 3

After removing Lund from the network, the network was still running and was able to route messages from Stockholm to Shanghai.



```
(sweden@127.0.0.1)20> r1 ! update.
update
(sweden@127.0.0.1)21> r3 ! update.
update
(sweden@127.0.0.1)22> r1 ! {send, shanghai, 'china@127.0.0.1'}.
(stockholm): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
{send,shanghai,'china@127.0.0.1'}
(uppsala): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
(sweden@127.0.0.1)23>
```

```
update
(china@127.0.0.1)20> r5 ! update.
update
(china@127.0.0.1)21> r6 ! update.
update
(xian): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
(beijing): routing message ('china@127.0.0.1') from (stockholm) to (shanghai)
(shanghai): received message ('china@127.0.0.1') from (stockholm)
(china@127.0.0.1)22>
```

Figure 3

4 Conclusions

In this assignment, we implemented a link-state routing protocol using Dijkstra's algorithm to ensure optimal routing paths. I have strengthened my understanding of Dijkstra's algorithm and learned how to use it in a practical routing system.