HW Report 2: Routy – A Small Routing Protocol

Ayushman Khazanchi

September 22nd, 2021

1 Introduction

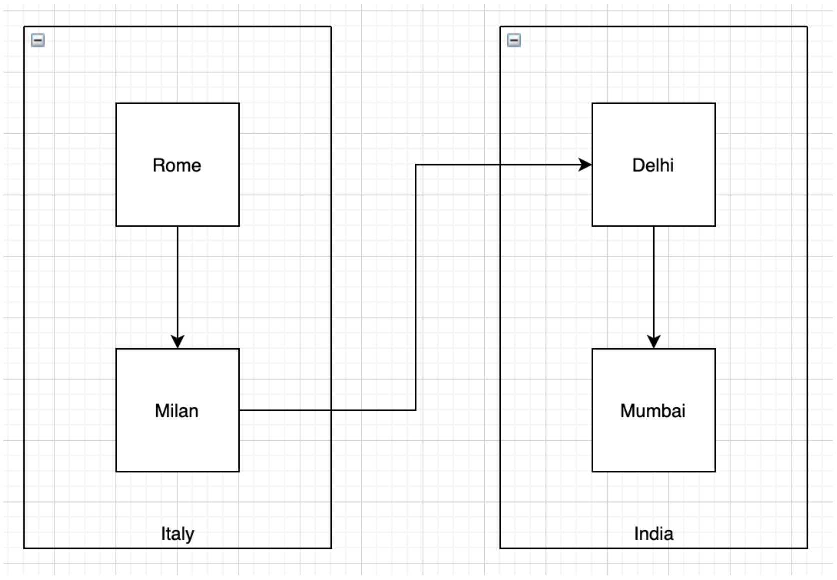
For this homework assignment the goal was to implement and understand a simple routing system based on the link-state routing protocol. As part of the exercise, we are able to describe an implementation of a link-state routing protocol, describe how we can attempt to maintain consistent views, and understand what happens when network failure occurs. The routing paths are implemented using Dijkstra's algorithm which ensures that a message always takes the shortest path to its destination.

2 Main Problems and Solutions

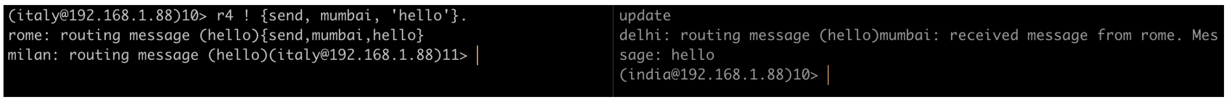
The main problems that I faced were mostly during the implementation. It was quite difficult to do this assignment initially without much boilerplate code. Implementing the algorithm from scratch was a lot of effort but it was difficult also to understand the exact requirements in the PDF. However, the main challenges with the coding aspect were in getting the routers to work together with the link-state protocol.

3 Evaluation

Initially, to make sure my code was working I tried passing messages between two routers in the same node (country). Once this was working, I tested by two different nodes (two countries) and two routers per node (four cities). I had two routers in *Milan* and *Rome* in *Italy* and *Mumbai* and *Delhi* in *India*. The *Milan* and *Delhi* routers acted as the gateway routers as they were linked together.



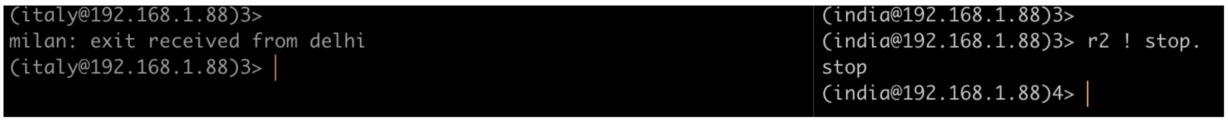
And, we can see the message from Rome (r4) to Mumbai (r1) traversing our route above. The image below shows two terminals side-by-side with the message being sent from Rome (r4) to Mumbai (r1). The message gets routed via Milan and Delhi and eventually is received by Mumbai.



```
(italy@192.168.1.88)10> r4 ! {send, mumbai, 'hello'}.
rome: routing message (hello){send,mumbai,hello}
milan: routing message (hello)(italy@192.168.1.88)11>
```

```
update
delhi: routing message (hello)mumbai: received message from rome. Mes
sage: hello
(india@192.168.1.88)10>
```

In the same scenario above I also implemented a sort of network failure to understand what would happen. I stopped the Delhi router and almost instantly I received an alert on my Milan router stating that it had lost connection with Delhi. The image below depicts this test. You can see on the left side the Milan router has received an exit from Delhi.

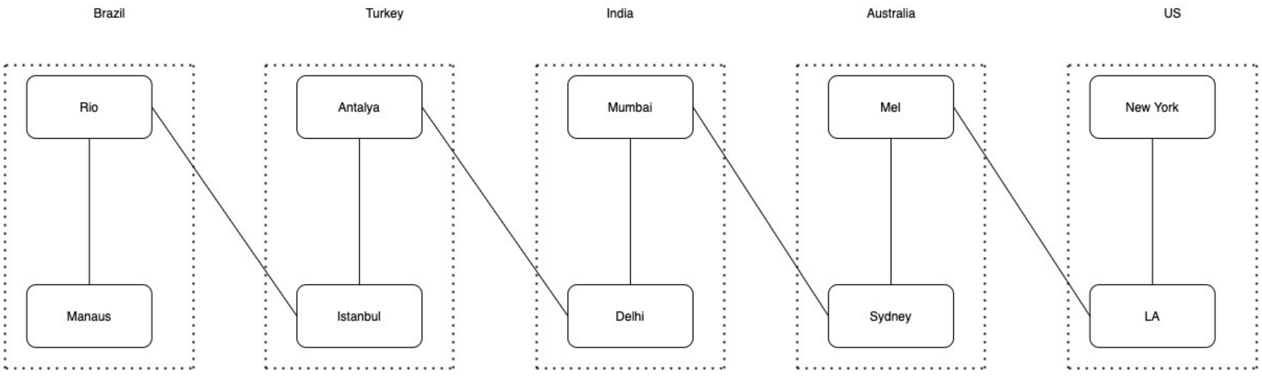


```
(italy@192.168.1.88)3>
milan: exit received from delhi
(italy@192.168.1.88)3>
```

```
(india@192.168.1.88)3>
(india@192.168.1.88)3> r2 ! stop.
stop
(india@192.168.1.88)4>
```

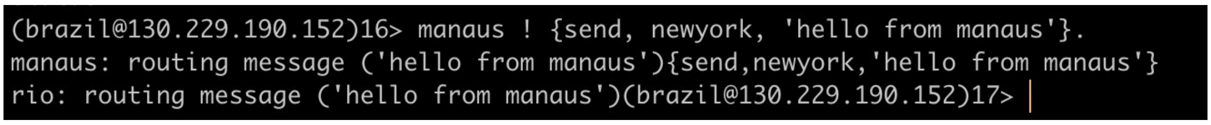
The World

The following implementation was performed along with my peers: Abdullah Abdullah, Chrysoula Dikonimaki, Vasigaran Senthilkumar, and Ilias Merentitis. We setup multiple routers based in different countries and different regions. The diagram below represents our network.



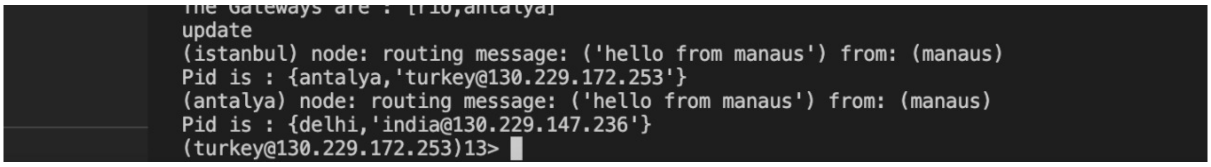
Once the network was setup and the routers were broadcast and updated, we ran a test to pass the longest message on the network originating from Manaus and going to New York. This was pretty cool to see played out in real-time as all our routers showed some output of routing the message onwards. The following screenshots from each machine describe the path of the message.

Message starting from Manaus and routing through Rio -



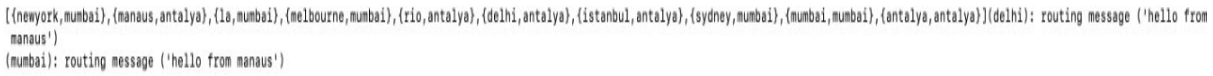
```
(brazil@130.229.190.152)16> manaus ! {send, newyork, 'hello from manaus'}.
manaus: routing message ('hello from manaus'){send,newyork,'hello from manaus'}
rio: routing message ('hello from manaus')(brazil@130.229.190.152)17>
```

Routing via Istanbul and Antalya –



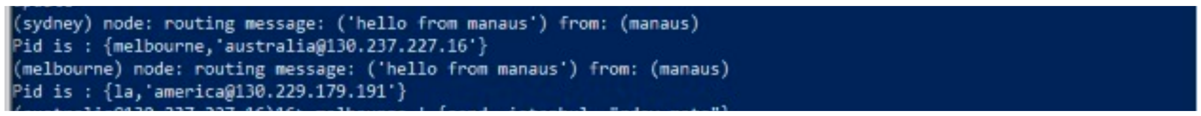
```
The Gateways are : [rio,antalya]
update
(istanbul) node: routing message: ('hello from manaus') from: (manaus)
Pid is : {antalya,'turkey@130.229.172.253'}
(antalya) node: routing message: ('hello from manaus') from: (manaus)
Pid is : {delhi,'india@130.229.147.236'}
(turkey@130.229.172.253)13>
```

Routing via Delhi and Mumbai –



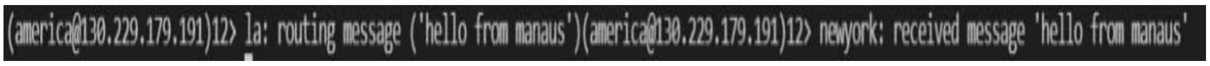
```
[{newyork,mumbai},{manaus,antalya},{la,mumbai},{melbourne,mumbai},{rio,antalya},{delhi,antalya},{istanbul,antalya},{sydney,mumbai},{mumbai,mumbai},{antalya,antalya}](delhi): routing message ('hello from
 manaus')
(mumbai): routing message ('hello from manaus')
```

Routing via Sydney and Melbourne –



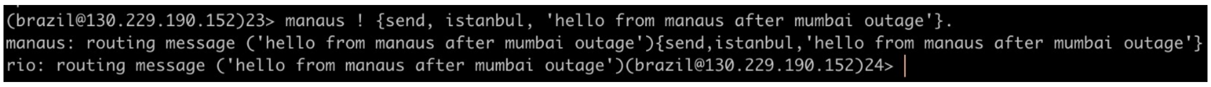
```
(sydney) node: routing message: ('hello from manaus') from: (manaus)
Pid is : {melbourne,'australia@130.237.227.16'}
(melbourne) node: routing message: ('hello from manaus') from: (manaus)
Pid is : {la,'america@130.229.179.191'}
```

Message arriving in New York via LA –



```
(america@130.229.179.191)12> la: routing message ('hello from manaus')(america@130.229.179.191)12> newyork: received message 'hello from manaus'
```

As you can see the network functions as intended when all routers are working. However, when we take out one router (Mumbai) the network still works in parts. We simulated an “outage” by removing Mumbai router from the picture and we were still able to use the network from Manaus to Istanbul. This is pictured below with the message arriving in Istanbul from Manaus.



```
(brazil@130.229.190.152)23> manaus ! {send, istanbul, 'hello from manaus after mumbai outage'}.
manaus: routing message ('hello from manaus after mumbai outage'){send,istanbul,'hello from manaus after mumbai outage'}
rio: routing message ('hello from manaus after mumbai outage')(brazil@130.229.190.152)24>
```

4 Conclusions

In this assignment I learned a lot of the basics of Erlang including how to use lists, tuples, and the list functions present in Erlang. I learned a bunch about routing protocols, routers, and I even got a more in-depth understanding of how real-world routers work to some extent. While the initial part of the assignment was quite tough to do without code once I got things running in the terminal by example it was much easier to understand. After that, simulating outages and watching messages travel across the network was actually quite fun.