Report 1: Rudy

Alexandru Matcov

September 18, 2024

1 Introduction

The following seminar required the implementation of a link-state routing protocol in Erlang.

In the implementation, a router broadcasts link-state messages to all its neighboring routers, informing them of its connected gateways. These neighboring routers process the message and then broadcast it to their own neighbors. As a result, the link-state message of a single router propagates throughout the entire network, allowing every router to learn the gateways of the router that originally sent the message. If all routers follow this procedure, each will possess a complete map of the network.

With this information, each router can then compute its own routing table, determining where to send a message destined for a specific location. To calculate the shortest paths in the network, the Dijkstra algorithm is used, ensuring efficient message routing.

2 Main problems and solutions

The main challenges faced in this assignment were related to the algorithm implementation of the link-state routing protocol and the data structure of keeping a history of the seen messages.

The former challenge, namely the implementation of the *iterate/3*, part of the Dijkstra algorithm, was challenging due to the complexity of the *foldl/3* function which is part of the **lists** library. The function iterates through a container, in the described implementation - a list, applying a defined function (fun) for each of the elements of the list. At the same time, *foldl/3* has an accumulator which helps keeping the results from the previous function call and pass it to the next. Below the implementation of the "brains" Dijkstra algorithm is represented:

```
NewList = lists:foldl(fun(Element, AccSorted) ->
    update(Element, N+1, Gateway, AccSorted)
end, Tail, ReachableNodes)
```

The second faced challenge in this assignment was the implementation of a dictionary for keeping track of the history. This data structure is contained in each router of the network and has N entry, where N is the number of all routers in the network. The reason for choosing a *dict* data structure over *lists* was for performance and scalability.

3 Evaluation

In order to evaluate the implementation of the *routy* a automated test was created setting up 3 different Erlang nodes. A test was written called **test.erl** which starts several routers in each country, makes connections between router, then broadcasts and updates each router.

In Figure 1 is depicted the graph of the tested network.

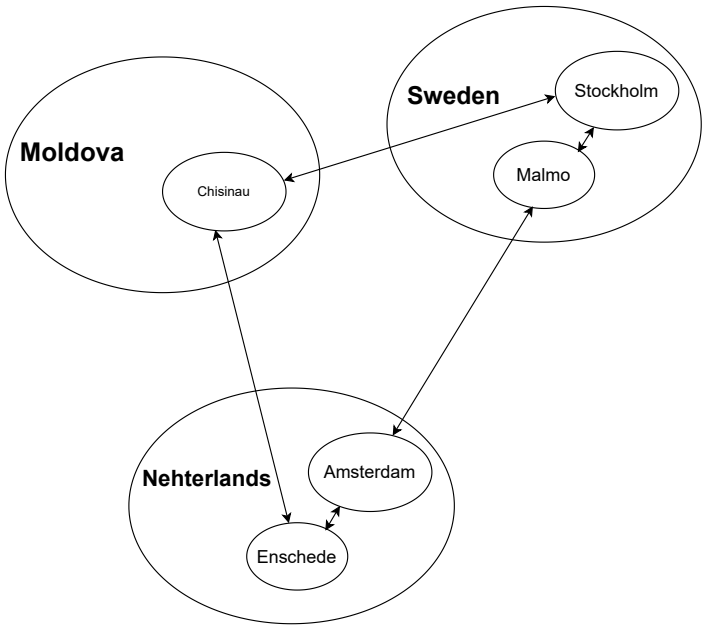

Figure 1: Network Graph

The furthest away cities are Stockholm and Enschede. Thus, exactly these 2 were taken as parameters to test the implementation. The objective of the test is to:

- Check the routing of the message through the shortest path (Enschede → Chisinau → Stockholm).
- Handle correctly the failure of a router by updating the routing table.

These objectives have been satisfied in the following test cases represented in the pictures below.

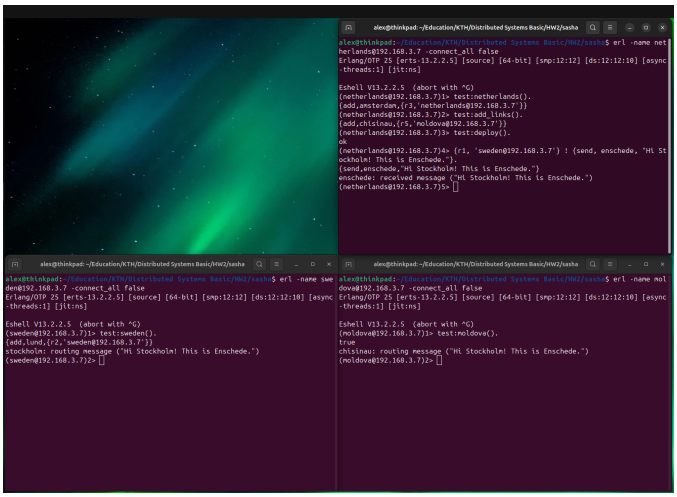

Figure 2: Routing through Moldova

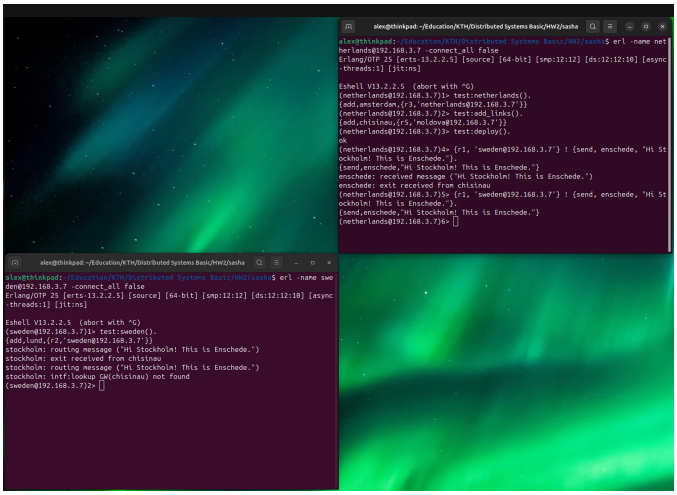

Figure 3: Router down and routing table updated

4 Conclusions

In conclusion, implementing the link-state routing protocol is an interesting solution to build a network of nodes on. However, there are some things that could be improved, like for example scalability as it's not efficient to keep track of the entire network by constantly building and updating routing tables.