Report 2: Routy

Gabriele Morello

September 21, 2022

1 Introduction

In this homework I have learnt how to implement a link state routing protocol in Erlang. In this protocol each node knows where to route a message given its final destination, to do so it uses a routing table to find the gateway. I have not implemented a control mechanism for lost messages.

2 Main problems and solutions

For this homework we didn't have an Erlang skeleton so I had to study more the language syntax and I had to explore more functional programming, but the instructions where exhaustive therefore I didn't have many problem to understand the general structure, the hardest part was to implement functions because data structure and workflows were given. I will focus first on the Dijkstra module because it was the hardest and because it is the core of the program: the routing table is created and updated here. This module implements the Dijkstra algorithm, as suggested in the instructions I used foldl to operate on lists. The main components of this module are a sorted list of tuples, where each entry contains the name of a node, the length of the path to the node and the gateway to reach the node, and the routing table: a list of tuple where the first element is the destination node and the second is the gateway to reach it. I also use the map from the map module: a list of tuples that contains in each entry a node and every link from it. we have the following functions:

- entry(Node, Sorted): return the length of the path from the current node to the Node in the arguments, 0 if there's no such node
- replace(Node, N, Gateway, Sorted): operates in the sorted list replacing old entries with the new arguments, the new list gets sorted before it gets returned
- update(Node, N, Gateway, Sorted): calls entry() and replace() to update the list with a new set of value in case the new length is shorter

than the one in the list, it means we found a shorter path to the destination.

- iterate(Sorted, Map, Table): I implemented it as a recursive function where for every iteration it consumes an element of the sorted list and it builds a step of the route table, iterating through the nodes that are reachable from the element we got in the sorted list it also updates the sorted list.

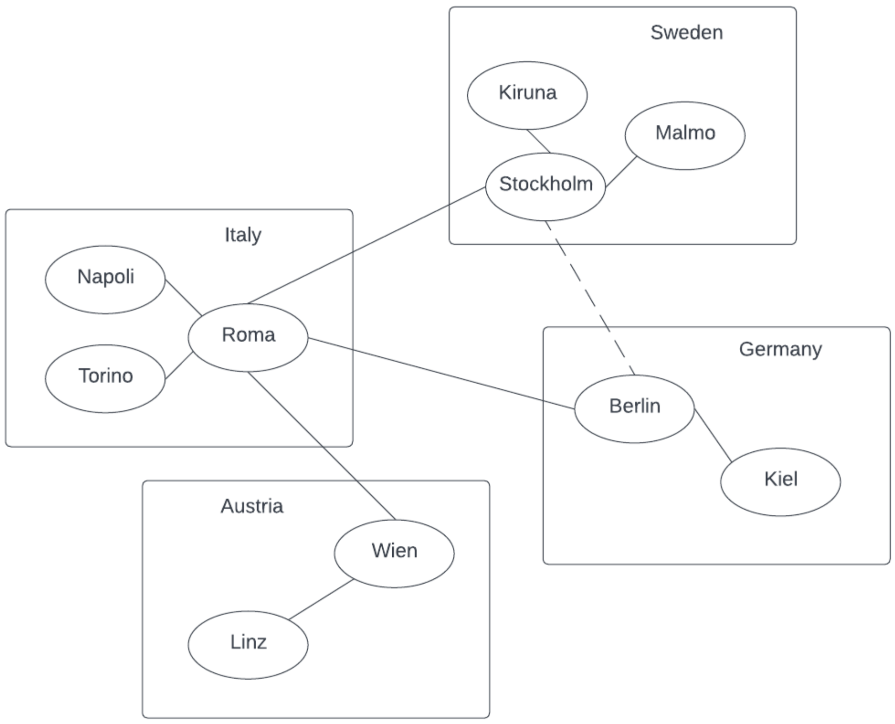
- table(Gateways, Map): constructs the routing table, the first two steps are to add dummy entries with infinite length for all the nodes and entries with 0 length for gateways, we will update this initial table calling the iterate function.

- route(Node, Table): looks for the node in the table to retrieve to which gateway we need to route the message.

As suggested in the instructions I often used lists:foldl to operate through lists

3 Evaluation

I tested my implementation locally first and later I built a network with Perttu Jääskeläinen, Jonathan Arns and Fabian Zeiher to simulate connection around Europe.



During personal tests I found errors in my hist:update function that causes an infinite loop and some typos, after fixing it I tried a ring configuration, a chain with up to five nodes and a star graph, I always had the message delivered and always with the optimal path. As suggested in the extra part of the assignment I connected my nodes to some of my classmates nodes and we carried out some tests, in the Figure 1 I reported the configuration, note that the dashed line is a connection we opened later to test loops. Every nation is a different machine and every city is a node running in a machine, each of us started his own nodes, then we connected nations through the capital city node, after the connection we sent the broadcast command to every node and then the update command, at this point each node had a full Map, routing table and list of interfaces, we randomly tested messages from many nodes and everything worked. To test loops we added a new connection: the dashed line in the diagram above. We then broadcast and updated every node and we tried to send messages and we observed their path to see if they take the shortest path and every time we observed the right scenario. We then killed the roma node and, after broadcasting and updating, we sent messages, we also tried to contact unreachable nodes and obviously messages weren't delivered but we didn't received any exceptions. This test was executed in front of a teaching assistant during an help session. I added a europe module that contains most of the functions used in these tests as well as the network setup

4 Conclusions

It was interesting to see how different implementation of the same protocol worked together when connected in a network, without any major issues. I didn't know how to use foldl before this lab but after some initial confusion, I figured it out and it was very helpful to operate in lists. I tried to use directives to the preprocessor during tests, I used them before in C but never in Erlang and they sped up the debugging phase. I also added more printing statements in given functions because I wanted to see what is happening under the hood during broadcasts and updates.