Report 2: Routy, a small routing protocol

Simon Cathébras

September 13, 2012

1 Introduction

During this assignment, we developed a router in Erlang. Routers are used in interconnecting several networks. Thus, routers are part of one very common distributed system: the Internet.

For this seminar an implementation of dijkstra has been realised. To do so, `lists` API from Erlang has been used.

In addition, if the `routy.erl` file is excepted, all required modules have been implemented without using the skeleton from last year. Thus, we could learn a lot about Erlang programming. We will discuss about this later, in the second section of this document.

Basically, the Dijkstra's algorithm is used to build the routing table. This table is then used to choose the best route a given packet has to take to reach a given destination. Basically, the map is seen as a graph which Nodes are routers and edges are the link between nodes. In this model, each edge has the weight 1. To be noted that, we could give dynamic weight to the edges, depending on the occupation of the link (the more the link is occupied, the higher is the weight). This way, the Dijkstra's algorithm computes the shortest way in this graph to reach a given node.

Finally, after implementing the whole router given in the paper, here is a small test showing how it works.

1. In the same country (Erlang machine) Sweden, create three routers: Lund, Stockholm and Goteborg.
2. The map is the following:
   `{{lund, [stockholm]},{stockholm, [lund, goteborg]}, {goteborg, [stockholm]}}.`
3. Send a message from Goteborg to Lund. The only way to reach Lund is to go threw Stockholm.

Let's see what happened now, after all nodes have been created and the map initialized.

```
(sweden@127.0.0.1)21> goteborg ! {send, lund, "hello"}.
goteborg: routing message ([104,101,108,108,111])
{send,lund,"hello"}
stockholm: routing message ([104,101,108,108,111])
lund: received message [104,101,108,108,111] from goteborg
```

This is, fortunately, the result we were expecting.

2 Main problems and solutions

Coding in Erlang During this assignment, we encountered some problems during Dijkstra's implementation, especially when we implemented the `iterate` function. Our first way to do it was writing a big **case** inside the function. It was not especially readable, nor efficient. We swept to a more common, in an Erlang way, implementation, where we are matching the different case directly in the parameters of the function.

3 Conclusions

During this seminar, we actually implemented for the first time a Dijkstra Algorithm in a concrete problem. During this implementation, we have also discovered the existence of an useful API in Erlang for lists manipulation.