Homework 2 Routy - a small routing protocol

1 Introduction

The task is to implement a link-state routing protocol in Erlang. In a link state protocol every node constructs a map of the connectivity to the network which shows which nodes are connected to each other. Each node calculates the best possible path from it to every destination in the network. The collection of best paths will form each node's routing table. It uses Dijkstra's algorithm to calculate the lowest cost paths.

Routers can be connected to each other with directional one-way links and they can only communicate with the routers that they are directly connected to. Routing processes should consult its routing table and determine which gateway (a routing process that it has direct connection to) is best suited to deliver the message. If a message arrives at it's destination it is printed on the screen.

2 Main problems and solutions

A good representation of a directional map was needed where updating and finding directly connected nodes should be done easily. For this a map module was implemented where each entry consisted of a city with a list of directly connected cities: [{berlin,[london,paris]}].

A Dijkstra algorithm was implemented for computation of the routing table in a module called dijkstra. The table is represented by a list with one entry per node where the entry describes which gateway and city should be used to reach the node: [{berlin,madrid},{rome,paris},{madrid,madrid},{paris,paris}]. It is a collection of the best paths to a node. In the algorithm we will use a sorted list (based on length) when we calculate a new routing table where each entry will hold the name of a node, the length of the path to the node and the gateway that we should use to reach the node: {berlin, 2, paris}.

In the dijkstra module there is an iteration function which takes the first entry in a sorted list (which is the name of the node), finds the nodes in a map reachable from this entry and updates the sorted list. The entry is then added to the routing table. When there are no more entries in the sorted the table is then complete.

A router will also need to keep track of a set of interfaces, for this an interface module was created. A interface is described by the symbolic name (london), a process reference and a process identifier.

A history module was implemented to keep track of what messages has been sent. It contains two procedures:

- new(Name) Return a new history, where messages from Name will always be seen as old.

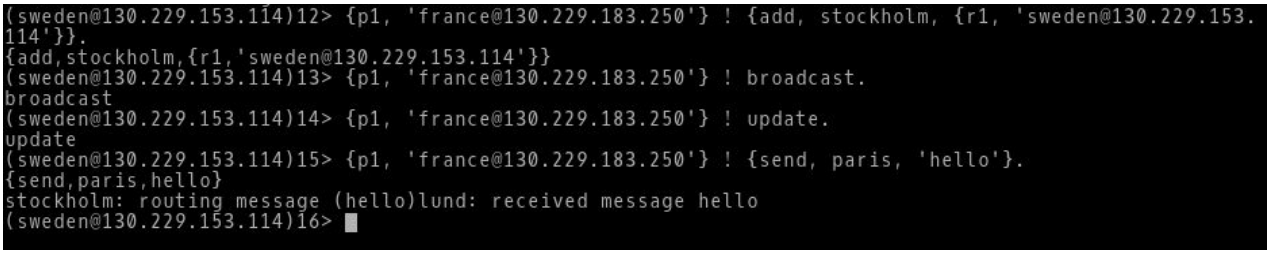
- update(Node, N, History) Check if message number N from the Node is old or new. If it is old then return old but if it new return {new, Updated} where Updated is the updated history

Last but not least a router module was implemented to route the messages through a network of connected nodes, maintain a view of the network and construct optimal routing tables.

3 Evaluation

I did a simple benchmark test program to run the router program and see if it is possible to route messages between the routers. I started three routers and connected them all to each other: stockholm registered under r1, lund registered under r2 and malmo registered under r3. They were all able to send and receive messages to and from each other. The update function is not executed automatically and must be called after each broadcast to maintain an updated routing table.

To test the program from different machines we set up two nodes: france and sweden. The images below shows how the communication went.



4 Conclusions

In this seminar I've learned how link-state protocols work and I've also got a deeper knowledge of how routers work and how they can route messages between each other. It was interesting to see how the messages were routed through the network and how two machines were able to communicate with each other.