Homework 2 : Routy

Reethika Ambatipudi,
September 21, 2022.

1 Introduction –

I have implemented a link-state routing protocol in Erlang. Started several routers in a network, made connections between them and sent messages from one node to another. Dijkstra algorithm automatically finds the shortest possible path between the two nodes and the messages are sent. Our ultimate aim is to send the message between 2 nodes in the shortest possible path possible.

The main concepts involved in this are link state protocol which includes flooding, routing table, dijkstra algorithm. The main topic related to this in Distributed systems is Networking.

2 Main problems and solutions -

The main problem I faced was understanding how link state protocol works. It was very fascinating to know the Dijkstra algorithm. In this process, I also understood the whole process of link state protocol is divided into 2 phases where the flooding of link state messages is done in the first phase and a routing table is built using Dijkstra algorithm in the second phase. Every time we make changes in the network, it should be broadcasted and updated. Figuring out the history module took sometime.
I had to know how keyfind/3, keydelete/2, map/2 and foldl/3 work.

```
keyfind(Key, N, TupleList) -> Tuple | false
keydelete(Key, N, TupleList1) -> TupleList2
foldl(Fun, Acc0, List) -> Acc1
> lists:foldl(fun(X, Prod) -> X * Prod end, 1, [1,2,3,4,5]).
120
```

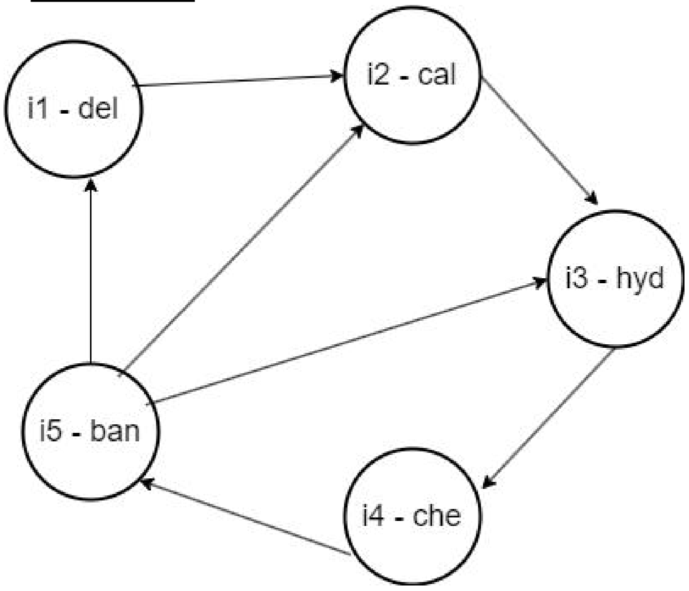
3 Evaluation -

```
C:\Users\reeth\IdeaProjects\LearnErlang\routy>erl -name india@192.168.0.10
Eshell V13.0.4  (abort with ^G)
(india@192.168.0.10)1> routy:start(i1,del).
true
(india@192.168.0.10)2> routy:start(i2,cal).
true
(india@192.168.0.10)3> routy:start(i3,hyd).
true
(india@192.168.0.10)4> routy:start(i4,che).
true
(india@192.168.0.10)5> routy:start(i5,ban).
true
(india@192.168.0.10)6> i1 ! {add, cal, {i2, 'india@192.168.0.10'}}.
del Received add signal Node:cal
  Pid:{i2,'india@192.168.0.10'}
{add,cal,{i2,'india@192.168.0.10'}}
(india@192.168.0.10)7> i2 ! {add, hyd, {i3, 'india@192.168.0.10'}}.
cal Received add signal Node:hyd
  Pid:{i3,'india@192.168.0.10'}
{add,hyd,{i3,'india@192.168.0.10'}}
(india@192.168.0.10)8> i3 ! {add, che, {i4, 'india@192.168.0.10'}}.
hyd Received add signal Node:che
  Pid:{i4,'india@192.168.0.10'}
{add,che,{i4,'india@192.168.0.10'}}
(india@192.168.0.10)9> i4 ! {add, ban, {i5, 'india@192.168.0.10'}}.
```

```
che Received add signal Node:ban
  Pid:{i5,'india@192.168.0.10'}
{add,ban,{i5,'india@192.168.0.10'}}
(india@192.168.0.10)10> i5 ! {add, del, {i1, 'india@192.168.0.10'}}.
ban Received add signal Node:del
  Pid:{i1,'india@192.168.0.10'}
{add,del,{i1,'india@192.168.0.10'}}
(india@192.168.0.10)11> i5 ! {add, cal, {i2, 'india@192.168.0.10'}}.
ban Received add signal Node:cal
  Pid:{i2,'india@192.168.0.10'}
{add,cal,{i2,'india@192.168.0.10'}}
(india@192.168.0.10)12> i5 ! {add, hyd, {i3, 'india@192.168.0.10'}}.
ban Received add signal Node:hyd
  Pid:{i3,'india@192.168.0.10'}
{add,hyd,{i3,'india@192.168.0.10'}}
(india@192.168.0.10)13> i1 ! broadcast.
broadcast
(india@192.168.0.10)14> i2 ! broadcast.
broadcast
(india@192.168.0.10)15> i3 ! broadcast.
broadcast
(india@192.168.0.10)16> i4 ! broadcast.
broadcast
(india@192.168.0.10)17> i5 ! broadcast.
broadcast
(india@192.168.0.10)18> i1 ! update.
update
(india@192.168.0.10)19> i2 ! update.
update
(india@192.168.0.10)20> i3 ! update.
update
(india@192.168.0.10)21> i4 ! update.
update
(india@192.168.0.10)22> i5 ! update.
update
(india@192.168.0.10)23> routy:get_status(i5).
Name:ban
N:1
Hist:{dict,5,16,16,8,80,48,{[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]},{{[],[],[],[],[[hyd,0]],[[che,0]],[],[],[],[],[],[],[[del,0],[ca
l,0]],[],[[ban,inf]],[]}}}
Intf:[{hyd,#Ref<0.2315035310.3116630019.26059>,{i3,'india@192.168.0.10'}},{cal,#Ref<0.2315035310.31166300
19.26052>,{i2,'india@192.168.0.10'}},{del,#Ref<0.2315035310.3116630019.26045>,{i1,'india@192.168.0.10'}}]
 Table:[{ban,hyd},{che,hyd},{hyd,hyd},{del,del},{cal,cal}]
 Map:[{che,[ban]},{hyd,[che]},{cal,[hyd]},{del,[cal]}]
  ok
(india@192.168.0.10)24> i5 ! {send, hyd, 'hello'}.
ban routing hello message from ban to hyd
ban Found pid {i3,'india@192.168.0.10'}
{send,hyd,hello}
hyd received hello message from ban
(india@192.168.0.10)25> i5 ! {send, cal, 'hello'}.
ban routing hello message from ban to cal
ban Found pid {i2,'india@192.168.0.10'}
{send,cal,hello}
cal received hello message from ban
(india@192.168.0.10)26> i5 ! {send, del, 'hello'}.
ban routing hello message from ban to del
ban Found pid {i1,'india@192.168.0.10'}
{send,del,hello}
del received hello message from ban
(india@192.168.0.10)27> i5 ! {send, che, 'hello'}.
ban routing hello message from ban to che
ban Found pid {i3,'india@192.168.0.10'}
```

```
{send,che,hello}
hyd routing hello message from ban to che
hyd Found pid {i4,'india@192.168.0.10'}
(india@192.168.0.10)28> che received hello message from ban
(india@192.168.0.10)28> i1 ! {send, ban, 'hello'}.
del routing hello message from del to ban
del Found pid {i2,'india@192.168.0.10'}
{send,ban,hello}
cal routing hello message from del to ban
cal Found pid {i3,'india@192.168.0.10'}
(india@192.168.0.10)29> hyd routing hello message from del to ban
(india@192.168.0.10)29> hyd Found pid {i4,'india@192.168.0.10'}
(india@192.168.0.10)29> che routing hello message from del to ban
(india@192.168.0.10)29> che Found pid {i5,'india@192.168.0.10'}
(india@192.168.0.10)29> ban received hello message from del
(india@192.168.0.10)29> i4 ! {send, hyd, 'hello'}.
che routing hello message from che to hyd
che Found pid {i5,'india@192.168.0.10'}
{send,hyd,hello}
ban routing hello message from che to hyd
ban Found pid {i3,'india@192.168.0.10'}
(india@192.168.0.10)30> hyd received hello message from che
(india@192.168.0.10)30>
```

4 Conclusion -



If we have to send a message from i4 to i3, there are 3 possible ways.
1. i4-i5-i3
2. i4-i5-i2-i3
3. I4-i5-i1-i2-i3

In the above execution, I took the shortest path via i5(ban), which clearly shows that the link state protocol is working fine.

The assignment was very useful in understanding the link state protocol.
Thank you!!