Report : Routy, an Erlang implementation of Link State Routing

Avneesh Vyas

September 21, 2016

1 Introduction

The report describes the various steps and procedures followed during the implementation of a link state router in Erlang mainly using its lists library and distributed programming capabilities.

2 Main problems

Link state routing one of the two main classes of routing protocols. The other being distance vector routing. The basic premise of link state routing is that each node (or router) broadcasts its knowledge of its neighbors to all other nodes in the network. This is done via periodic 'link-state' message broadcasted message. On receipt of each broadcast message, each node prepares a map representing the whole network. And then the node runs the Djikstra algorithm on this map, to determine the shortest path to each node in the network. To route an incoming packet to its destination, all that the router needs to know is the next hop to forward the packet. So, once the shortest path to each node is available, router can generate a 'routing table' containing the best next hop gateway for each destination node.

3 Solution

To implement such a router, the following data structures were identified:

1. Map: This data structure is used by the router to represents its knowledge of the network. It is structured as a key-value store with node as the key and a list of immediate neighbors as its value.
2. List of destination nodes: This data structure is the sorted list of 3-member tuple {destination_node, length, next_hop_gw}. So a node not reachable shall be stored in sorted list with length field set to 'inf' and next_hop_gw set to 'unknown.
3. Routing table: This is the output data structure that we will generate by running an algorithm on the above 2 data structures. This is

structured as a key value store with destination node as key and the next hop as its value.

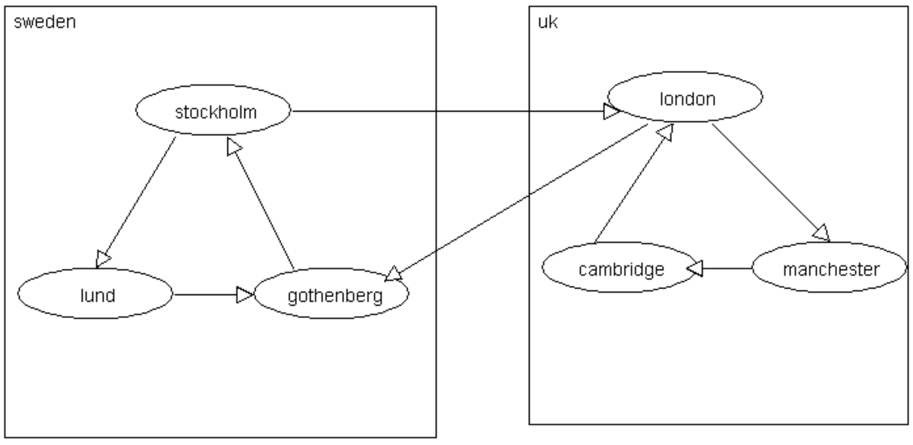
<u>Algorithm</u>

1. In our case, the immediate neighbor lists are manually provided to a particular router (maps to a single routing process running in an erlang engine). In real world, router usually automatically detects its immediate neighbors.
2. Then each router instance periodically broadcasts its list of neighbors across the network.
3. Each router by receiving each other's broadcast messages builds/updates the network map (data structure 1 in section 2).
4. Using the information from the map above, each router prepares a sorted list of nodes (data structure 2 in section 2).
5. As a final step, each router instance iterates over the sorted list to create a routing table with node mapped to best next hop gw.
6. Step 3,4,5 are repeated with every broadcast message received.

In our implementation, link-state broadcast and routing table update is manually performed.

4 Test Network

Our router implementation was tested by spawning several router instances simulating the following test network.


sweden
stockholm
lund
gothenberg
uk
london
cambridge
manchester