Report: Routy - a small routing protocol

Muhammad Azeem (azee@kth.se)

September 17, 2020

1. Introduction

A small router network using Erlang processes was implemented by making multiple small routers inside Erlang nodes. The routers communicated using link-state protocol which is normally part of the OSPF; i.e. the most used protocol in the router communication.

The routers represented individual cities while the nodes represented the countries.

As mentioned, the routers communicated by link-state routers and they basically had a map of routers in the network. Furthermore, a Dijkstra module was implemented which was used to keep track on the hops required to reach routers as well as to select the shortest path to a router.

In case of a node or router failure, the remaining nodes (as well as their routers) will remain active and communicate. However, it's possible for some messages to be lost because of network failure. Any part of the network failing is however communicated though to other routers.

A history log is also kept to make sure that old messages are not being propagated continuously.

2. Main problems and solutions

- Firewall issues: Firewall were blocking access to communication to other nodes on external pcs, which can be disabled in order to test.
- Short names were given to nodes in order to avoid the nodes reaching namespace limit.
- Non-public IP address. Unlike production routers I couldn't make using a public IP address of course. However, it's possible to communicate within the same network.
- The router was rather rudimentary with basic features and avoided using complicated algorithms to shortest path calculation.

- Error handling and message failing. No error handling has been implemented however message failure can occur. And old messaged get dropped in order to avoid them continuously circulating.

3. Evaluation

At first a single node was run with a couple of routers. Control messages were sent to start and setup the nodes. DOWN messages could also be communicated when a node goes down in a multiple node scenario.

There was not much delay in message transfer due to the network being fairly small and nodes being in the same machine or using the same Wi-Fi network.

The history was useful to keep track to the messages and it is a rather useful implementation to drop old messages.

The network used single-directional connections in which there is a possibility of some router being only able to receive messages and not transmit; this can happen if there network hasn't been setup properly so a connection is lost between a router. A bidirectional link would solve the issue of a router only receiving message and not being able to transmit.

Network is fault-tolerant; the whole network doesn't go down because a single router is lost. However, a single router can be of course be disconnected (as mentioned above). Furthermore, when a router goes down, the DOWN control message is transmitted to other routers so that they can update their tables.

4. Conclusion

I have understood routers and specially the link-state protocol in more detail. As well as learned some fault-tolerance techniques as well as routing and finding the shortest path. It is also easy to scale up the network by simply adding more nodes/routers and sending control messages to setup and inform other routers on the new routers in the network.