# Lab 4: Link-state Routing Simulator

Deadline: December 8

## Goal

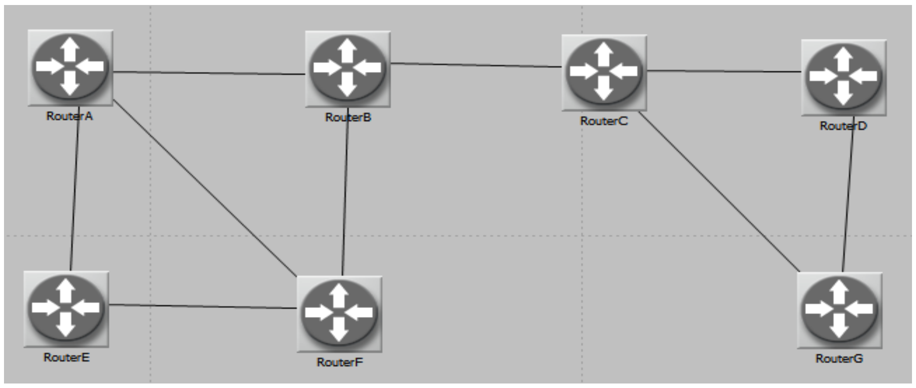
The goal of this lab is to develop a link-state routing algorithm to run over several routers. You will learn how to broadcast messages between the routers and implement the controlled flooding procedure. Moreover, you will also learn how to perform the Dijkstra short path algorithm to calculate the optimal paths. Remember, in a link-state, all routers know all other routers and recognize the cost of each link between them. In this lab, you are allowed to use any programming language of your choice. However, performing this lab with **Python** programming will gain five additional points.

**Broadcast (Controlled Flooding)**: The key to avoiding a broadcast storm is for a router to choose when to flood a packet judiciously. One way to control flooding is by using the sequence-number-controlled flooding method that goes as follows. When router A sends its link-state packet to all of its neighbors, it will increment a flooding sequence number for that packet. Next, when the neighbor router receives the packet, it will check if the sequence number is higher than the last one it received before, and then it will forward the packet; otherwise, it will drop it. One way to implement the control flooding approach is to create a data structure where you store packets information and do a fast lookup and decide whether you forward or drop them.

**Dijkstra Algorithm**: The Dijkstra algorithm makes use of the priority queue, and by using this algorithm, the packets will be broadcasted from the source router to the destination router using the shortest path. **Note:** You should print the routing table log for each router in a separate file after the algorithm finishes.

## Topology and Information

Below is the network topology that will be used for this lab:



There are in total 7 routers and they are named alphabetically RouterA to RouterG.

The below table represents the assigned IP address for the routers:

| **Routers** | **IP Addresses** |
|---|---|
| RouterA | 10.10.10.10 |
| RouterB | 10.10.10.20 |
| RouterC | 10.10.10.30 |
| RouterD | 10.10.10.40 |
| RouterE | 10.10.10.50 |
| RouterF | 10.10.10.60 |
| RouterG | 10.10.10.70 |

The below table represents the cost between each connection:

| **Links between Routers** | **Costs** |
|---|---|
| RouterA - RouterB | 10 |
| RouterA - RouterF | 20 |
| RouterA - RouterE | 50 |
| RouterB - RouterC | 30 |
| RouterB - RouterF | 30 |
| RouterC - RouterG | 10 |
| RouterC - RouterD | 50 |
| RouterD - RouterG | 10 |
| RouterE - RouterF | 30 |

The table below represents the port number for each of the connection between the router:

| **Links between Routers** | **Port Number** | **Links between Routers** | **Port Number** |
|---|---|---|---|
| RouterA - RouterB | 1 | RouterB - RouterA | 3 |
| RouterA - RouterF | 2 | RouterF - RouterA | 2 |
| RouterA - RouterE | 3 | RouterE - RouterA | 1 |
| RouterB - RouterC | 1 | RouterC - RouterB | 3 |
| RouterB - RouterF | 2 | RouterF - RouterB | 3 |
| RouterC - RouterG | 2 | RouterG - RouterC | 1 |
| RouterC - RouterD | 1 | RouterD - RouterC | 2 |
| RouterD - RouterG | 1 | RouterG - RouterD | 2 |
| RouterE - RouterF | 2 | RouterE - RouterF | 1 |

## Building your Client/Server Router

Each of the routers is acting both as a client and server that can accept packets from a designated port and also able to forward the packet to the next connected router. Moreover, the router should examine the control flooding sequence for the packets before forwarding. The router executable should read four command line arguments, the socket IP address, port number, router name, and a routing file, like follow:

```
$> ./Router <IP address> <port number> <router name>
<filename>
```

The routing file will consist of multiple lines, each expressing a neighbor router and the link cost as follows: (e.g. Router A):

| **IP Addresses** | **Port Number** | **Cost** |
|---|---|---|
| 10.10.10.20 | 1 | 10 |
| 10.10.10.60 | 2 | 20 |
| 10.10.10.50 | 3 | 50 |

The output file for each of the router will be a log file which will look something like this (e.g. RouterC):

| **Source IP** | **In Port** | **Destination IP** | **Out Port** | **Content** | **Forward/Drop** |
|---|---|---|---|---|---|
| 10.10.10.70 | 2 | 10.10.10.40 | 1 | Broadcast Message | Forward |
| 10.10.10.40 | 1 | 10.10.10.20 | NA | Broadcast Message | Drop |

## Tips and Guidelines

- At a fundamental level, the server and the client are programmed together as a single router. The router should accept packets over a UDP network socket and it should forward the packets to the destination.

- You should test your code by changing the cost of each of the paths.
- Run each router in a separate Linux/Windows terminal
- Each router executable, while running, should accept input from the user in the terminal to initiate a broadcast operation and type. Below are some examples:
  - broadcast,noDijkstra - Broadcast a message to all other routers natively
  - broadcast,withDijkstra - Broadcast a message to all other routers using the optimal path calculated by the Dijkstra's algorithm
  - **Graduate only:** p2p,Dijkstra,10.10.10.60 - Send a per to per message to router F using the optimal path calculated by the Dijkstra's algorithm
- You are free to design your own link-state packet structure, but it should have the following fields:
  - IP address to source
  - Port of source
  - Flood control sequence number

## Experiments

The following scenarios need to be implemented:

- Broadcast without Dijkstra: a source router should natively send a broadcast packet to all other routers on the network. Repeat this experiment 7 times for each router.
- Broadcast with Dijkstra: In this experiment, a source router should perform the Dijkstra algorithm and calculate the optimal path to all other routers on the network. Next, the router will send a broadcast packet to all other routers by following the optimal path. Repeat this experiment 7 times for each router.
- **Graduate Only:** Peer to Peer: In this experiment, you will perform the Dijkstra algorithm in a source router and calculate the optimal path to all other routers on the network. Next, you will send one packet from that source router to a designated router following the optimal path. Repeat this experiment 7 times for each router.

## Create the Project Repository

In LoboGit, create a new project in your group and name it **"Lab4"**. You should use this repository to maintain your code and collaborate with your group members.

## Submission

You have to solve this assignment in groups of two to four students. You must use a git repository (LoboGit), Lab 4 project, to maintain and submit your code. You can't change your group until you get approval from me, you should send me an email requesting such change with a good reason.

When you submit your work, the repository should contain the following:

- A wiki page, which includes a brief explanation of how your group implement the simulator, each experiment details, and output logs. Also add a walkthrough on how we can run your code. Be precise and brief in your writing.
- Your code files. Make sure your code is well documented and readable, I will deduct points if the code misses documentation.
- A makefile to build your code, if you are using C or Java.