CS3205 – Computer Networks
Even Sem. 2023, Prof. Krishna Sivalingam
Lab 5: OSPF Routing Algorithm
Due date: April 27, 2023, 11PM, On Moodle
Individual Assignment
Languages: C or C++ or Java or Python or combination of any two

April 10, 2023

1 Overview

The purpose of this lab is to implement a simplified version of the Open Shortest Path First (OSPF) routing protocol.

Given a set of N routers, the goal is for EACH router to: (a) exchange HELLO packets with neighbours, (b) create Link State Advertisement (LSA) packets based on neighboring nodes' info, (c) broadcast the LSA packets to all other routers in the network, (d) construct the network topology based on the LSA packets received from other routers, and (e) determining the routing table entries based on this topology, by using Dijkstra' algorithm (single source – all nodes shortest paths). If multiple equal-cost paths exist, any one of them can be reported.

The HELLO packets will be exchanged every x seconds; the LSA updates will be sent every y seconds; the routing table computation will be done every z seconds.

2 Details

The project will be implemented on a single computer, with one Linux process per OSPF router. Each process might have multiple threads to implement different functionality.

Input: The input to your program will be as follows:

```
./ospf -i id -f infile -o outfile -h hi -a lsai -s spfi
```

The values specified in the command line are:

- -i id: Node identifier value (i)
- -f infile: Input file

- -o outfile: Output file
- -h hi: HELLO_INTERVAL (in seconds)
- -a lsai: LSA_INTERVAL (in seconds)
- -s spfi: SPF_INTERVAL (in seconds)

Input File: The input file format is as follows:

The first entry on the first line specifies the number of routers (N). The node indices go from 0 to $(N-1)$. The second entry on the first line specifies the number of links.

Each subsequent row contains the tuple $(i, j, \mathrm{MinC}_{ij}, \mathrm{MaxC}_{ij})$. This implies a bidirectional link between nodes i and j. The use of minimum and maximum will be defined later.

14		35	
0	2	2	8
2	3	5	10
3	4	6	20
4	7	4	10
...			

OSPF Processing: Each OSPF router (running as a Linux process) will perform the following actions:

- Obtain necessary parameters including its node identifier (i).
- Read the input file and find out its neighboring node identifiers.
- Establish a UDP socket on port number $(10000 + i)$ for all OSPF communications.
- Send a HELLO message to its neighbors, once every HELLO_INTERVAL seconds. This value is specified in the command line; Default: 1 second. You can have one thread implement this part. Packet Format:

HELLO	srcid

  where srcid is replaced with i.

- When a router receives a HELLO message on an interface, it will reply with the HELLOREPLY message along with the cost. The cost reported for link_{ij} by node j for a packet received from node i is a RANDOM number between $MinC_{ij}$ and $MaxC_{ij}$; this cost is generated randomly each time. Node i, on receiving this message, will store this value as the cost for link_{ij}. The message format is:

HELLOREPLY	j	i	value for link ij

- Send a Link State Advertisement (LSA) message to its neighbors, once every LSA_INTERVAL seconds. This value is specified in the command line; Default: 5 seconds. You can have one thread implement this part. Packet Format:

LSA	srcid	Seq. Number	No. Entries	Neigh1	Cost1	Neigh2	Cost2	...

The sequence number is incremented by the sender for each LSA message that it sends.

- When a node receives an LSA message from a neighbor, it will store the LSA information and forward the LSA to all interfaces other the interface that the packet arrived on, *if and only if* the newly received LSA's sequence number is strictly greater than the last known sequence number from the sender.
- Determine the topology using all recent Link State Advertisement (LSA) messages received from all other routers; and then run shortest-cost path computation algorithm every SPF_INTERVAL seconds. This value is specified in the command line; Default: 20 seconds.

  You can have one thread implement this part. The output will be stored in the routing table output file along with the time stamp. The output file name for Node i will be *outfile*–i.txt, where outfile is specified in the command line.

  Output Format:

Routing Table for Node No. 1 at Time 30		
Destination	Path	Cost
2	1-3-2	5
3	1-3	2
4	1-3-2-4	10
...		

3 What to Submit

Create a folder `Lab5-CS22Babc`, the main directory.

- Source code files (that implement the routing protocol)
- One sample input file used for your testing, with at least 8 nodes and 20 links.
- Corresponding Output files, showing the routing table entries for all routers.
- Report showing two different input topologies (each with at least 8 nodes and 20 links) and corresponding routing tables.
- README and COMMENTS files

4 Grading

- Implementation working on Single Machine: 90 points
- Demo and Viva Voce Exam: 10 points

5 Extra Credit

Introduce dynamic link changes in the network – network links can be brought down; downed network links can be brought up later; new links can be added to the network. You must define the necessary functionality, user interface, implement and demonstrate the dynamic network operation. This is worth 20 points.

Modify the implementation to run each router on different machines or virtual machines and demonstrate the same. This is worth 10 points.

The extra credit demonstration will be presented to Prof. Krishna – using a video of program in execution and a brief report highlighting the details of changes made.

6 Miscellaneous

1. This is an INDIVIDUAL assignment. No sharing of code.
2. Ask questions EARLY and start your work NOW. Take advantage of the help of the TAs and the instructor.
3. Submissions PAST the extended deadline SHOULD NOT be mailed to the TAs. Only submissions approved by the instructor or uploaded to Moodle within the deadline will be graded.
4. Demonstration of code execution to the TAs MUST be done using the student's code uploaded on Moodle.
5. NO sharing of code between students, submission of downloaded code (from the Internet, Campus LAN, or anywhere else) is allowed. The first instance of code copying will result in ZERO marks for the Lab component of the Course Grade. The second instance of code copying will result in a 'U' Course Grade. Students may also be reported to the Campus Disciplinary Committee, which can impose additional penalties.
6. Please protect your Moodle account password. Do not share it with ANYONE, including your team member. Do not share your academic disk drive space on the Campus LAN.
7. Implement the solutions, step by step. Trying to write the entire program in one setting may lead to frustration and possibly failure.
8. Downloaded Code from the Web (except for possibly Dijkstra's algorithm) will NOT be considered for grading and such action will lead to academic penalties.