OSPF: Routing Protocol

Implementation and testing on the GENI network testbed

Lab 6: CS3210 - Computer Networks Lab

Instructor: Krishna M. Sivalingam
Due Date: 19th April, 6 PM, 2015
Jan – May 2015, Indian Institute of Technology Madras

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: (i) A single computer, with one process per OSPF router; and (ii) the GENI testbed (`www.geni.net`) that is available from the U.S. Detailed instructions to get it working on GENI will be given on Moodle.

There will only be a small difference between the local implementation and the GENI implementation, which will be explained in a separate document on moodle. This document will focus only on the local implementation.

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:

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

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 edges.

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

OSPF Processing: Each OSPF router (running as a 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 $(20000+i)$ 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 $MaxC_{ij}$. 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 second.

  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 `Lab6-CS1xBabc`, the main directory.

- Source code files (that implements the routing protocol)
- One sample input file used for your testing, with at least 7 nodes and 15 edges
- Corresponding Output files, showing the routing table entries for all routers.
- Report showing two different input topologies (each with at least 7 nodes and 15 edges) and corresponding routing tables.
- README and COMMENTS files

4 Grading

- Implementation working on Single Machine: 75 points
- Implementation working on GENI: 25 points (Details of implementation will be given in a separate document on moodle)

Note: This is an INDIVIDUAL assignment. If you are having difficulty with the assignment, please talk to the TAs/the instructor. Downloaded Code from the Web (not even for Dijkstra's algorithm) will NOT be considered for grading and such action will lead to academic penalties.