Link-State Routing Assignment

# CSCI 4760

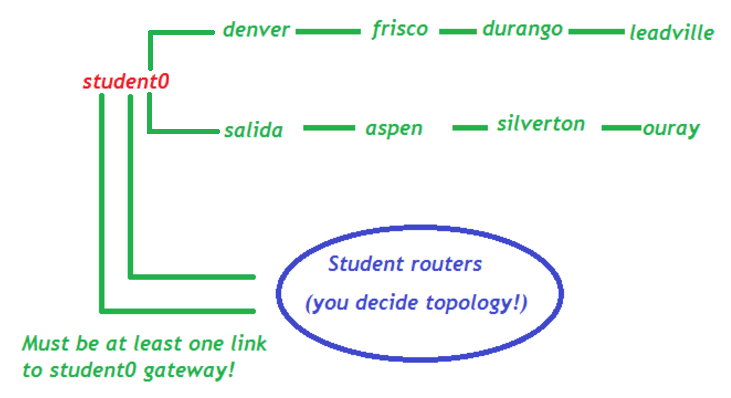
**What is this?** In this exercise we will implement an overlay network using link state routing and forwarding. For any student to get full credit, at least 20 students must create a connected overlay network and demonstrate passing echo messages along it.

### Academic Honesty Guidelines for this Assignment

This is an individual assignment, so you may not copy code from any source, help any other student with code, or get help with your code from any person except the instructor and TA. Aside from that, I encourage you to discuss ideas with each other. You will definitely need to discuss the overlay network topology, as described below.

### The Network Topology

Here is a map of the overlay network as it will exist when we get done. The machines on the top two rows are simulations inside my routing program, so you can’t connect directly to them. The names are places in Colorado, where I hope to hike to my 50th high school reunion this coming summer.



### Components of your project

As usual, we will simulate our underlying Layer 3 using UDP packets. Use these ports:

* xxx20: receive Link-State messages
* xxx21: receive incoming echo messages

As usual, these ports should be in your assigned range. My project will run on ports 20020 and 20021 -- you can use it for test purposes. I will provide a CSV file with all the port numbers -- I suggest that you upload it and process it in your router constructor:

Here are the software components you must implement:

* ***Master LSRouter thread:*** initializes your node’s link table and the lists of routing ports (xxx20) and forwarding ports (xxx21) and starts all the subsidiary threads listed below. Every 30 seconds thereafter, runs Djikstra’s algorithm to recompute its routing table.
* ***Link message broadcast thread***: every 30 seconds, send Link-State messages to each of the other students (including student0), describing the overlay links involving your machine. The format of these messages will be defined below
* ***Link message receive thread***: listens on xxx20. When it receives a Link-State message, adds the link to the link table if it does not already exist, and updates the expiration time for the link.
* ***Forwarding/echo thread***: listens on xxx21 for incoming Echo message. When it receives and echo message, either sends an Echo reply message (if this message is addressed to you), forwards it using your routing table if it is for another user to whom you can route, or replies with an Echo reply message containing an error message if you cannot route it to the intended destination.
* ***Echo Client:*** send and receive Echo messages. Each message will be initially sent to port xxx21, where your forwarding/echo thread will deliver it to the final destination.

### Link State Messages:

I have provided a LinkMsg class a for converting link messages to JSON and reconstituting the objects from a JSON representation. It’s important that we all use the same format for our link-state messages:

.

Note that this example just reconstitutes a dictionary. To reconstitute an actual object, see this link: http://stackoverflow.com/questions/3768895/python-how-to-make-a-class-json-serializable

When you create a link state message, the expiration time should be 120 seconds in the future (except for my fake links, which have expiration times far in the future). Use **time.time()** to find the current time in seconds since the beginning of the “Unix epoch,” Jan 1 1970.

Routers should discard any link messages that are past their expiration date, so your routing advertiser must run constantly for this assignment to work. Use

**nohup python LSRouter.py &**

to start your router; it will then continue when you log off the system. When you log back on, you can kill your router with the **ps** and **kill** commands.

Echo and Response Messages

Echo and response messages have a from\_node, to\_node, and msg fields. Transmit them in JSON format: {“msg”: “Launch all vessels”, “from\_node”:”Starfleet”, “to\_node”:Spacedock}

From <from-address> to <to-address>:message

-- for example, **from fred to denver:Howdy, pardner!**

The message text for echo messages can be anything you want. The response message should be

Received your message <original message>

For example: from denver to fred: Received your message Howdy, partner!

Since the echo and reply messages are both received on port xxx21, be sure NOT to reply to messages that start with "Received!"

### Test Your Receiver Thread:

### I have created a program to send out link state messages on demand. Just type

### **ls\_send <port-number>**,

### where <port-number> is your receiving port number (xxx20).

### Network Topology:

You, the students, should create an overlay network that includes each student account and student0, the instructor account. (You can’t link to the Colorado towns, because I just simulate them. You can and should route packets to the Colorado towns.) Please, no more than four links out of any one student node. You may create up to four links from student nodes to student0. Please send me a message with your network topology, and updates as they occur.

### Log files:

I encourage you to use log files, both to debug subtle issues and to verify your accomplishments, so you can get partial credit in case not everything works perfectly. Please mention the log files in your README.

### Grading Rubric:

|  |  |
| --- | --- |
| Link Message Broadcast | 20 pts |
| Link Message Receive Thread | 20 pts |
| Correctly runs LS routing algorithm | 20 pts |
| Forwarding | 20 pts |
| Echo client | 20 pts |
| Overlay network with >= 20 students (all credit or none!) | 20 pts |
| Total, including extra credit | 120 pts |