CS 6543 – Spring 2015

HW-04 – Due Date: <u>check BB Learn</u>

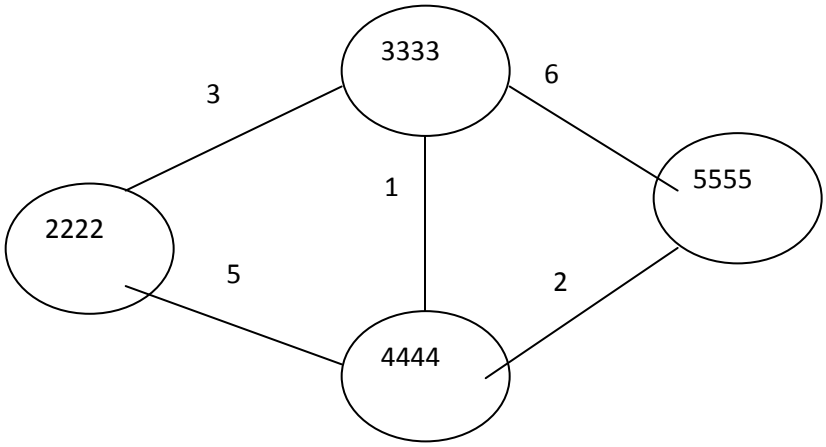
You must submit your source code and related files on BB Learn,

!!! no late HW will be accepted!!!

Total 100 points

In this HW, you are asked to implement a program (say node.c) to emulate a router running a simplified version of link-state routing (e.g., OSPF) as an application on top of UDP.

Specifically, you will implement one program node.c and execute it as many times as we want on the same computer to emulate routers/nodes in a network. Each router/node will be identified by a UDP port number while the hostname is the same for all, i.e., localhost. So your program should first get the UDP port number as a command line parameter and create a UDP socket to listen to. It then should wait for user to enter neighbors' port numbers and cost. But before entering neighbors' ports or costs, make sure all the nodes are started successfully. Then you can enter the neighbors' port numbers and cost for each node. For example, we can create the below network by running our node program as follows:




```
> node 2222
Enter neighbors:
3333    3
4444    5
-1  -1
```

```
>  node 3333
Enter neighbors:
2222    3
4444    1
5555    6
-1  -1
```

```
> node 4444
Enter neighbors:
2222    5
3333    1
 5555   2
-1  -1
```

```
>  node 5555
Enter neighbors:
3333    6
4444    2
-1 -1
```

OSPF does a lot of things but in this assignment you will ignore most of the details and only implement the following key functionalities.

Flooding (40%) without reliability part: Suppose that UDP links on the same computer are 100% reliable. So each node creates a Link State Advertisement (LSA) packet and sends it to its neighbors. A receiving node will check if it has seen this LSA before or not. If not, it will keep/save a copy in its Link State Database and forward it to its own neighbors except the one from which it received that LSA. Since we assume UDP links are reliable, you don't need to implement ACKs timers etc.

When a node learns a new link etc, print the current view of the network at each node, so that we can see how your program works.

Implement Dijktra's algorithm (40%) to compute the shortest path tree based on the current view of the network it should be a general implementation to work with any topology. But you don't need to implement the efficient ones, the basic one with O(n^2) is enough

Then **Determine/print the routing table** (20%) based on the shortest path tree you computed at each node... (you can do this whenever the current view of the network changes).

You don't need to use the exact packet format defined in OSPF. Just define your own LSA packet formats to carry the information needed for your simplified link-state (OSPF) protocol.

GRADING (Total grade will be 100 points.)

1. (30 points) Implementation (your source code(s) and makefile, Comments in your source program)

3. (70 points) Correct Execution (give the output of your program for at least one case)

What to turn in

- Please put all your documentation, source, output files under a directory called lastname-hw4, then ZIP this directory as a single file and submit it on BB Learn.
- Please after submission, DO NOT change or delete your soft copies, we might execute them later together for grading….