Project 2

CS 484/504 Computer Network I (Fall 2012)

Assigned: October 25, 2012

Due: December 4, 2012

Note: The program should be implemented in C or C++ and compile in an openSuse linux distribution.
– You will have to "tar" your project directory and submit it by email in the blackboard ONLY.
– Command to tar (combine) a set of files in a directory: tar cvf filename.tar directory_name
– Command to untar a directory: tar xvf filename.tar (check the unix/linux man pages to learn more).
– The name of the tarred file should be GroupMember1FullName_GroupMember2FullName_Project2.(tar).
– *Use modularized programming with code spread over several files and use the .h files for class (or structure) definitions. High quality documentation is a must.*
– Having a Makefile to compile your code is necessary.

This project requires you to design and implement a *discrete event network simulator*, to simulate the functioning of a computer network. A discrete event network simulator runs using a global virtual clock. The simulation is propagated in time by handling the pending events at their scheduled times, with the scheduled events giving rise to other events which are scheduled in the future and so on.

To simulate the network, you will need to:

(a) *randomly* create 150 nodes in the network, with each node capable of acting as a source/destination and a router that can forward packets.

(b) In addition, the nodes are connected together by links (generated randomly as well), in a way that every node can reach every other node in the network, i.e., the network of nodes is *connected*. The connectivity of the network can be checked by implementing a Depth First Search (*DFS*) on the network graph. There are other methods to check for connectivity, you are free to explore them.

(c) The bandwidth of the links and the propagation delay on the links follow a uniform distribution. The bandwidth $b(u, v)$ of a link (u, v) is uniformly distributed in the range $0 - 1$ mbps, i.e., $b(u, v) \sim \mathcal{U}(0, 1\text{mbps})$. The delay $d(u, v)$ of a link (u, v) is uniformly distributed in the range of $1 - 10$ seconds, i.e., $d(u, v) \sim \mathcal{U}(1\text{sec}, 10\text{secs})$.

(d) Data is generated at a source in the network using the Poisson distribution with a parameter $\lambda = 0.5$, hence the inter-arrival times are exponentially distributed, with parameter $\lambda = 0.5$. Assume that the time taken to generate a packet at the source is negligible.

(e) The size $S(i)$ of a packet i follows a uniform distribution, with $S(i) \sim \mathcal{U}(0, 1\text{mb})$.

(f) The packets are queued at the input queues of the routers in the network and the routers service the packet using an exponential distribution given by parameter $\mu = 1.0$. Assume that the queues are based on the First-Come-First-Serve (FCFS) discipline.

(g) Each router has output queues on which the packet's are placed when they are switched by the switching fabric. The packet stays in its output queue until it can be transmitted on the forward link to the next router

(or the destination). Here also, the transmission is FCFS. Note that the number of input or output queues in the router equals the number of interfaces of the router. The maximum queue size of the input and the output queues of all the routers is 30.

(h) Each router maintains a routing table. The routing table can be created by running the Floyd-Warshall algorithm to find all pairs shortest path and storing the next hop to reach the destination. You can also run the single source shortest paths Dijkstra's algorithm to solve this problem and run it for all nodes in the network. Remember, this is a simulation, the routing table can be created at the start of the simulation and stored in each router's data structure. Assume that no node goes down in the network and the network is static, so that you do not need to exchange HELLO messages at regular intervals to re-configure the routing table.

Operations:

Your coding for this project will be in two parts. **In the first part**, you will write a program to generate a network graph consisting of 150 nodes and certain number of edges (generated randomly) and check if the graph is connected. If the graph is connected, you write it to a file. The format of the graph will be: First Line: Number of nodes <space> number of edges. Second Line onwards each line represents the edge that connects two nodes. One example with five nodes and seven edges may be:

5 <space> 7
0 <space> 1
0 <space> 3
0 <space> 4
1 <space> 3
1 <space> 4
2 <space> 3
3 <space> 4
** (represents the delimiter implying end of file)

Generate more than one such graphs to test your code. Later, you will be provided with test graphs to run your code on.

In the second part:

1. To run the program you need to use the following at the command line:
   <prompt> executable_name seed graph_filename
   The seed value is used to seed the random number generator, which you will use to generate random variables for events in the network. The other argument to the program is the name of the file from which to read the graph.

2. You will randomly select 20 source – destination pairs in the network, with the source generating and transmitting traffic to the destination.

3. You will run the simulator for 1000 seconds and gather the following statistics:

   - Total number of packets generated.
   - Total number of packets that reached the destination successfully.
   - Percentage of successfully received packets.
   - Average packet transmission time for each transmission.

- Maximum, minimum, and average time for completion for the transmissions.
- Maximum, minimum, and average number of packets dropped at a router.

Some useful pointers:

- A standard uniform random variable, u, i.e., $u \sim \mathcal{U}(0,1)$, can be generated as $u =$ (float) rand()/RAND_MAX.
- A non-standard uniform random variable v, such that $v \sim \mathcal{U}(a,b)$ can be generated as, $a + u \cdot (b-a)$, where u is a standard uniform random variable.
- An exponential random variable x, with parameter λ, i.e., $x \sim Exp(1/\lambda)$, can be generated as, $x = -\frac{1}{\lambda} \cdot \log(u)$, where u is a standard uniform random variable.
- In a discrete event simulator, a global virtual clock is used to simulate the progress of time. This clock is represented by a for loop, with a counter t (say), that wraps the whole code. For each instance of time (some value of t), your program would perform all the events that need to be performed in that time instance. For example, at time instance $t = 20$, a certain number of routers will move the packet(s) in their input queue(s) to the respective output queue(s), some of the sources will generate and send a packet, some destinations will receive packets, etc.
- Think of each node as an object or a structure. Each node has 2 two-dimensional arrays representing the input and the output queues. It also has a routing table where it stores the forwarding information for each packet. You need to think of the other elements that need to be a part of the node.

  Note: To simulate the transmission of the packet on the wire you can store the packet in a separate global data structure, which stores all the packets that are currently on the wire, until they are supposed to reach the next node. Or, you could store it in a bigger size output queue until the packet needs to be in the next node's input queue.