CS620 Assignment 1

Due date: September **11**, 2019

Objective: Build a Gossip protocol over a peer-to-peer network to broadcast simple messages.
Overview: Each peer in the network must be connected to a randomly chosen subset of other peers. The network must be connected, that is the graph of peers should be a connected graph. In order to bootstrap the whole process, the network should have at least one *Seed* node which has information (such as IP address) about all other peers in the network. Any new node first connects to a seed node to get information about other peers and then connects to a subset of these.

Once the network is formed, the peers broadcast messages in the network. This setup will be used in future assignments to form the base to build blockchain systems.

Network setup

Your network will consist of the following type of nodes:

1. *Seed Node*: A *Seed* maintains a list of client nodes in the form of IP address and port number pairs called *Client List (CL)*. A seed node is assumed to have a well-known IP address and port number. On startup, any new client registers itself with a seed node. Registration request triggers an event at the seed node to add an entry containing the client's IP and port number to the CL.

2. *Client Nodes*: On launching the *Client* at any node, it first registers its identity (IP address: port) with the seed nodes and requests the list of nodes from one or more *seed nodes*. You can hard code the IP:port for the seed node or read them from a file to make your life easier.

   After registration, a client requests the list of clients available at the seed node. In response, the seed node sends the list, out of which a client randomly selects a maximum of **4** nodes and establishes a TCP connection with each of them. While doing this, you should make sure that the resulting network is a connected one. You are free to use any method to do this.

   Along with this, every node maintains a data structure called *Message List(ML)* to efficiently broadcast a message so that every message travels through a link (pair of connected peers) at most once. ML consists of the hash of the message and a true/false bit, indicating the broadcast status of the message to its neighboring nodes.

3. *Link delays*: To make the network setup more realistic, you have to simulate the delay between two adjacent nodes to some random value ranged from A to B. Use the following command to simulate the delay:

   tcset <interface-id> –add –delay $x$ms –dst-network <IP Address>

   Related commands and their description is given below:

   tcset - can be used to set delays

   tcshow - can be used to see the currently set delays

tcdel - can be used to remove all delays. tc qdisc del dev <interface-id> root

Message format
A client can generate a message of the form

<self.timestamp>:<self.IP>:<self.Msg#>

Such messages will be generated by each client node every **5** seconds. The first message can be generated by a client after it connects to selected neighbors after registration. A client will stop message generation after it generates 10 messages.

Gossip protocol
After a node generates a message M, it transmits M to all its adjacent nodes. On receipt of a message the node checks for its entry in its ML. The presence of the entry results in no action at the receiving node, else it forwards the message M to all its neighbors and makes an entry in the ML(<hash(M),$true$>).

Program Output
Each node writes to the screen and also to an output file the list of peers it got from seed nodes. Every message received the first time should be displayed on the console as well as written to the output file along with a local timestamp and the IP address of the node from which it received the message.

Useful Links You can follow the links below to brush up your knowledge about socket programming.

1. Beej's Guide to Network Programming.
2. Python Network Programming.

Submission Instructions:

1. The assignment should be done in a group consisting of a maximum of two students.
2. The code should follow programming etiquette with appropriate comments.
3. Add a **README** file which includes a description of the code and gives detailed steps to compile and run the code.
4. Name your files as **client**.extension, **seed**.extension and **outputfile.txt**, where "extension" is the standard extension for the programming language you use.
5. Zip all your code as a single file with the name **rollno1-rollno2.tar.gz** and upload it to Moodle. Only one group member should upload the file.
6. You should be able to **demonstrate** the code on a minimum of three machines.