CIS656 – Distributed Systems – Fall 2025
Programming Assignment 6 – P2P Architecture
Maximum Points: 100 (25% of the final grade)
Due: 8:30PM on Wednesday, December 3, 2025
(You may work on that individually, or as a group of two or three people)

Objective

The objective of this programming assignment is for students to build a small P2P distributed system based on what they learned about communication, concurrency and P2P structure. Upon completion, students will be able to construct an unstructured P2P architecture.

Description

You are asked to develop two programs, called "central-server" and "peer". "central-server" is used to store the information of "peer"s (e.g., IP address and/or hostname). "peer"s need to register themselves to "central-server" and "central-server" returns the connection information to "peer"s. Multiple hosts execute the program "peer" to form a P2P system. We call such a host with a running "peer" program as a peer host. Once a peer-to-peer system is set up, any peer host can use the command "neighbors" to show a list of its neighbor peers. There are multiple peer hosts and one central server in the network.

- **Forming a P2P System**
  When the "central-server" starts, it shows a message "The Server is on!" and waits for peers' connections.

  Then a peer can join the network by registering itself to the central server first. In other words, a new peer should first contact the central server. After that, the central server looks up its record, randomly picks up an existing peer host in the network and returns its IP address to the new peer host. Then this new peer host will try to join the P2P system by sending connection request to this existing peer host.

  If there is no peer host in the network (the new peer host is the first one to register), the central-server of course does not return the IP address, but instead, it returns a confirmation message to the new peer host to let it form a P2P network. (After that, there is only one peer host in the P2P network.)

  To better explain, we assume the name of the existing peer host picked up and returned by the central-server is "A". And the new peer host is called "B". After the new peer host B receives the IP address of the existing peer host A from the central server, B sends the connection request to A. The following rule applies.

1) If A has no more than 2 neighbor peers, A accepts the connection request from B. Then B successfully joins the P2P network via the connection with A. Then A and B become neighbors.

2) If A has already 3 neighbors, A will decline the connection request from B. Then, A randomly chooses one of its 3 neighbors, and return C's IP address to B, for example. When B gets this information, it will send the connection request to C. If C has no more than 2 neighbor peers, C will accept B's connection request and let B join the P2P network via the connection with C. Otherwise, C will decline the connection request, but return one of its neighbor peer D to B and let B connect to D. This process will continue until B successfully joins the P2P network.

This rule is to make sure each peer host has no more than 3 neighbors in the P2P network.

Note: "central-server" and "peer" are not neighbors.

- **Displaying Members/Neighbors**
  - Central Server:
    After the peer-to-peer system is set up, in the central server's terminal, a user can type the command "members" to show the hostnames/IP addresses of current existing peers.

  - Peers:
    A user can type the command "neighbors" in a peer's terminal to show this peer's direct (one-hop) neighbor peer(s)' hostnames/IP addresses.

- **Termination**
  A user can enter the command "quit" to terminate a peer. **It is required that the termination is handled gracefully (i.e., its direct (one-hop) neighbor peer(s) should display a message like ": Peer eos03.cis.gvsu.edu disconnected!" if "eos03.cis.gvsu.edu" terminates).** Please make sure that there is no crash on any neighbor peer(s) and the central server. The remaining peer hosts and central server must still work correctly.

  If a peer terminates by "quit" command, you do not need to consider re-connecting the two separated network partitions together. However, you need to make sure these two separated networks work well respectively. Also, the latest list of current peers and neighbors should be correctly shown for "members" and "neighbors" commands after a peer terminates.

  The central server terminates when the "quit" command is entered in its terminal. If this happens, peers should still work fine, although no new peer can join the network.

- **Commands you need to support**
  - Central Server:
    1) members: show a list of current peers' hostnames/IP addresses
    2) quit: to terminate the central server

  - Peer:
    1) neighbors: show the hostnames/IP addresses of its direct (one-hop) neighbor peer(s)
    2) quit: to terminate a peer host

- **Concurrency**
  Your programs should support concurrency to meet the above requirements. For example, your peer program should listen to the port for new potential peers' connections, monitoring existing connections for neighbor peer(s)' termination, while also receiving keyboard command input. The central server program should also support the similar concurrency.

Any programming language is acceptable. Please make sure that your programs work well in CIS lab machines. Your programs must be able to work on multiple machines, i.e., the central server and each peer host should run on different machines well.

Grading Criteria:

- **Program correctness: 75%**
- **Demo or video: 25% (please see the requirements below. If you only submit the source codes but do not demo your program or make a video, 25% of your grade will be deducted.)**

Depending on the format, please follow the following requirements.

(1) If you can demonstrate your program in person with the instructor, please do that during the office hours, or at the end of Wednesday, December 3's class. Then you do NOT need to submit anything to BlackBoard.

(2) If you want to demonstrate your program online synchronously with the instructor, you can do that during the office hours, or at the end of Wednesday, December 3's class online via Zoom. Then you do NOT need to submit anything to BlackBoard.

If you cannot demonstrate your program in person or online synchronously with the instructor, please make a video to show the program correctness and go through your source codes to explain how they work. Then please submit the source codes and video link to BlackBoard. If you work in a group, only ONE group member needs to submit the source file and the video link, but please include your group members' names in "Submission" field of the submission page.