CSci4211: Introduction to Computer Networks
Programming Project
Due Date: April 21st, 2019 at 11:55 p.m

March 27, 2019

Phase 1 Due: **March 25th, 2019 at 11:55pm**
Phase 2 Due: **April 21st, 2019 at 11:55pm**

1 Description

Peer-to-peer (P2P) networks refer to computer networks that use a distributed architecture. Unlike in a client-server model, a P2P network has no central server, everyone becomes a server and acts as a client.

The primary purpose of a P2P network is to share resources and help computers work collaboratively, to deliver a specific service or to perform a particular task. P2P networks are used for sharing all types of resources including files, network bandwidth or disk storage space. In the scenario which a P2P network is used for file sharing: If a file is downloaded online from a P2P network (Think of BitTorrent websites), the file is downloaded in parts that come from many other computers (peers) in the P2P network that already have the file. At the same time, the file is also sent (uploaded) from your computer to others who ask for it. Some benefits of a P2P network are: P2P networks are extremely scalable, they enable faster file/data downloads. Some examples of real world applications of P2P networks are Skype, BitTorrent, VoIP, Spotify, SMTP (server-server).

2 Your Task

In this assignment, you are asked to create a P2P file sharing network in a rather non-genuine way. First you will implement the P2P network and next add support for file sharing using socket programming.

3 Details

Phase 1: A server S_1 will request to join the P2P network managed and maintained by a Metadata Sever M. The Metadata server will add S_1 to the P2P network recording its IP address. Next say server S_2 seeks to join the P2P network. M adds S_2 to the P2P network by referring it to S_1(the first server in its cache) and a connection is made from S_1 to S_2. M logs S_2's IP address. Now say server S_3 requests to join the P2P network as well, M goes through its logs in a top-to-bottom order and refers S_3 to the first server in its cache,S_1, since S_1 has not established its maximum number of connections per server in the P2P network(which is 2 for this programming assignment). A connection is made between S_1 and S_3. Next, S_4 requests to join the P2P network, M will refer S_4 to S_1 (The first server in its cache), S_4 will attempt to make a connection to S_1 but S_1 will reject the connection and refer S_4 to one of its neighbors, S_2. S_4 will then establish a connection with S_2 since S_2 has not yet established 2 connections. Lastly, S_5 will ask M to join the P2P network, M will refer S_5 to the first element in its cache, S_1, S_1 will reject S_5's connection (because S_1 already has 2 connections) and refer S_5 to S_2. S_2 will also reject S_5's connection and refer him to its neighbor S_4 and a connection will be made from S_5 to S_4. At this point, the P2P network will look as shown in Figure 1.

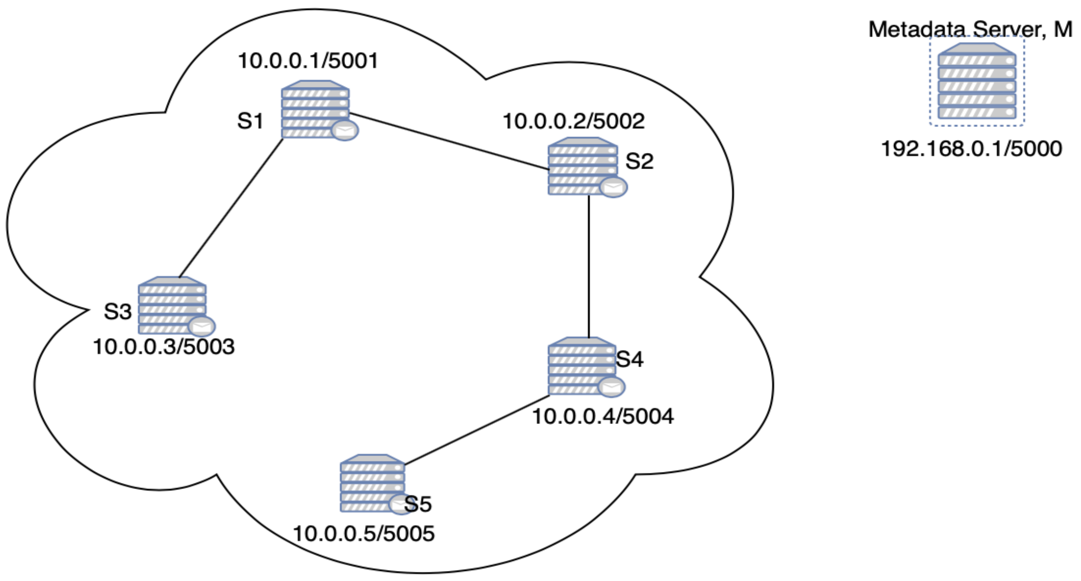

Figure 1: P2P Network Topology

Phase 2: The second phase of this programming project will be to implement file sharing. This is done as follows: Initially, each server has a set of files (these files will be provided to you). If S_4 needs to download file f_1 located on S_1, S_4 will reach out to its neighbor in the P2P network, S_2. S_2 will search its file directory for f_1, since S_2 does not have the file, it will reach out to S_1 (its neighbor. Note that S_2 does not reaches out to S_4 since it accepted the request from S_4). S_1 will search its file directory and when it finds f_1, it will establish a temporary socket connection to S_4 and send the file and then close the connection.

3.1 Metadata Server, M

Implement the Metadata server program as specified below:

1. M accepts a socket connection from a server via IP address and port number (For example 10.0.0.1 and port 5001) and waits to receive a flag from that server. Print out the string ******Connected to <Incoming Server ID>****** after the connection is established. Replace <Incoming Server IP address> with the server's IP address (—,—,—,—)

2. M receives the flag, ***"P2P"***(or any other message) from the connected server wishing to join the P2P network. Print out the string ******Valid Flag******. Print out ******Invalid Flag****** if the message received is different from <***P2P***> and asked the connected server to send a valid flag.

3. For every valid flag, M assigns an IP address (on the P2P network) to the server and refers it to the first server in its cache(already part of the P2P network). The message format send to the server will be of the form <**Port Number, Referred Port Number**>. A persistent socket connection is then established between both servers in the P2P network (provided the referred server does not have 2 socket connections already made) . Print out the string ******Connected to Referred Server ID******.

4. M closes its connection to the server and listens for other incoming connections.

3.1.1 Notes

- M is a multi-threaded socket server to handle multiple socket connections.
- Handle error checking when M receives a request to join the P2P network(See sample inputs and outputs provided).
- When M assigns an IP address to a server, the assigned IP address should be on a separate network (This is indicated by a port range).

3.2 Server

Implement the server program as specified below:

1. Ask the user if it wants to download a file or add a server to the P2P network. **1** : Add server to the P2P network and **2** : Download a file.
2. If the user enters **1**, ask him/her to input the ID of the server (for example S_1).
3. Ask the user to enter ***TOPO*** if he/her wants to view the current P2P network topology.
4. Use the server ID, and port number to open a socket connection to the Metadata server via IP address and port number.
5. Ask the user to input a flag (for example ***P2P*** or ***blablabla***).
6. Send the user input flag to M and wait for a response.
7. Base on the message received from M, the server will either repeat step 5 above or close its connection with M (if it receives <***Port Number, Referred Port Number***>).
8. If the server receives <***Port Number, Referred Port Number***> it will used its assigned port number and establishe a persistent socket connection using the port number of the server referred by M (***Port Number,Referred Port Number***).
9. Repeat step 1. If the user enters **2**, ask the user what file it wishes to download (for example f_1).
10. Parse through the P2P network and establish a non-persistent socket connection from the server that has the file to the server requesting the file and download the file. In your implementation, forward the file download request by passing the string ***"Request Server Port Number:File"*** so that when the server who has the file gets this request, it will know the file and port number of the server who made the request.
11. Whenever a server in the P2P network receives the string ***"Request Server Port Number:File"***, print out ******Received "Request Server Port Number:File" from Server****** where server is the ID of the server who forwarded the request.
12. Repeat the above instructions starting from 1 until the user enters **Stop** and gracefully exit and terminate all socket connections.

3.2.1 Notes

- No error handling on the ID of the server is required.
- No user flag error handling happens on the server side, only on the Metadata server side.
- For grading purposes, display message on the server terminal indicating what the server is doing. For example, when the server receives a valid query from M and needs to close it's connection with M, display something like "Closing socket connection with Metadata Server" and so on....This is because, I need to be able to follow the execution of your programs when it's running.
- Each server should first query its directory file before passing the connection. i.e When a server reaches out to a nearby server requesting a file, that server should first look through its file directory before forwarding the request to another nearby connected server.

4 Protocol and Program Messages

- <P2P> : Valid message send to M requesting to join the P2P network.
- <Port Number, Referred Port Number> : Message replied by M to the server requesting to join the P2P network."Port Number,Referred " are the referred info of the server already in the P2P network.

- <TOPO> : Program dumps the current P2P network topology in the format ”Server ID :, Established connection server IDs” (for example S_1 **:** S_2 S_3 indicates that server S_1 is connected to both S_2 and S_3 in the P2P network.
- <Server ID, Connections> : Format used when <TOPO> is evoked by the user to dump the current P2P network topology.
- <Request Server Port Number:File> : Where ”Request Server Port Number” is the port number of the server requesting to download file ”File”.
- <1> or <2> : User flag to ask the user to download a file.
- <Stop> : Terminates the server program closes all open socket connections exiting gracefully.

5 What to submit

You should upload the project files to the Canvas site under Programming Assignment Submissions section. Your submission (in a tar or zip format) should include the followings:

- The server and metadata server code (in any programming language of your choosing).
- A README file stating any compilation script you used, and any details that we should be aware of in order to compile and run your program. (Please try to run your program on CSE-Lab machines before submitting).
- Your README file should also include two or more paragraphs explaining what each segment of both the metadata server and server code do. This is not a line-by-line comments, but rather the logic of your algorithm and what each segment does, such as at what point does the metadata data send a respond to the server. Why the server needs multiple sockets? What is the point of threading and how we handle multiple threads? Also state any assumptions made.

6 Tips

- Start early to avoid last-minute issues.
- Save and compile your work frequently.
- Try to make your program simple, commented and easy to read.
- Do not try to learn a new language, utilize knowledge of languages you are familiar with.
- Some references to socket programming can be found on the course website.
- Good luck and most especially enjoy programming while learning socket programming.