Peer-to-Peer File System

The Pennsylvania State University
Fall 2019
Deven Patel
dbp4@psu.edu

11 October 2019

1 Introduction

In this project, I have designed a Torrent like peer-to-peer (P2P) file sharing system. I choose C programming language to implement the system. I used socket programming and multi-threading to implement the system.

2 Detailed Description of the System

This system include multiple clients (also called peers) and one central server. A client can join the P2P file sharing system by connecting to the server and providing a list of files it wants to share. The server keeps the list of all the files, shared by clients, on the network. The file being distributed is divided into chunks. For each file, the server keeps track of the list of chunks each client has. When a client intends to download a file, it will initiate a direct connection to the relevant clients to download the file. While downloading a file from the network, as a client receives a new chunk of the file it also becomes a source, of that chunk, for other clients. A client will be able to download different chunks of the file simultaneously from many clients (clients who have the file).

3 Protocol Specification

I use a messaging system amongst clients and between clients and the server. Below is the list of messages I use to design the system.

3.1 Chunk Registration

Client Register Request: Tells the server what files the client wants to share with the network. Takes the list of files from client. Send IP address (uint32) and listening port

(uint16), for other clients to accept client connections for download, the total number of files to register (int), and for every file, a filename (string) and its size (int).

Server Register Reply: For each file, it saves all the information in the metadata of server. When all the files have successfully registered it sends the message to client for success.

3.2 File List/Location Request

Client File List Request: Asks the server for the list of files. Once the metadata is received I search for the file and if the file is found in the network it prints the file name, chunk number of the file, file size, IP address of the client who has the file and that chunk and the listening port of the client.

Server File List Reply: It sends the most recent metadata to the client.

3.3 Chunk Register Request

Client Chunk Register Request: Tells the server when it receives a new chunk of the file and becomes a source (of that chunk) for other peers.

Server Chunk Register Reply: Advises if the chunk registration was a success.

3.4 Refresh metadata Request

Client Refresh Metadata Request: Asks the server for most recent metadata, which may include new clients with new files.

Server Refresh Metadata Reply: It sends the most recent metadata to the client.

3.5 File Chunk Request

Client File Chunk Request: Asks the client to return the file chunk. Sends file name (string), file size (int), chunk number (int).

Client File Chunk Reply: A stream of bytes representing the requested chunk.

4 Program Structure

When a client joins the network, they are asked to specify their listening port number as an argument. A listening port number is the port number that other clients will use to connect to this client for downloading. Listening port number will serve as a server on client side. I ask clients to update global definition "MY_IP" to write their IP address.

Once the client has established connection with the server, you will get an option to choose from the following option:

- UPLOAD FILE/S TO THE NETWORK
- SEARCH FOR A FILE IN THE NETWORK
- DOWNLOAD A FILE FROM THE NETWORK
- EXIT

4.1 UPLOAD

If you choose to upload files to the network, the interface will ask you to type in the name of the file/s (with path if not in current directory) you want to share with the network. You can enter multiple file names you wish to share with the network by adding space between file names. Once you write the file names, the code will compute file size and chunk numbers and submit them to the server. The server will than save this data and return the success message.

4.2 SEARCH

If you choose to search for a file in the network, the interface will ask you to type in the name of the file you want search for in the network. It will then send your request to server and server will return most recent metadata of clients and files they have in the network. The client will then search for the file and return the file name, chunk number, file size, IP address of client who has that file and chunk and the listening port of client. However, if the file is not found in the network it will return message saying "SORRY! THE FILE YOU ARE LOOKING FOR IS NOT IN THE NETWORK YET.".

4.3 DOWNLOAD

If you choose to download file from the network, the interface will ask you to type in the name of the file you want to download from the network. It will then look for the file in most recent metadata and compute the file size. Now, you will connect to other clients who have the file you are looking for to download. Thereafter, it will get the rarest chunks and start downloading simultaneously from clients who have the rarest chunks. Once, you receive these chunks it will then send an update to server that you have received this chunks and server will update its metadata accordingly. Before finding next rarest chunks, it will get the most recent metadata from server and connect to new client/s who might have downloaded chunks of the same file from network in the meantime. Now, you will also include these new clients in computing next rarest chunks. It will follow the same procedure until you receive all the chunks of the file you requested for download. While, you are downloading these chunks you will see the status of your download in the form which for example says "DOWNLOADING CHUNK 1 FROM CLIENT 2" and once you receive that chunk the interface will show "CHUNK FROM CLIENT 2 DOWNLOADED SUCCESSFULLY!!!". Once you receive these chunks and after registering the chunks the interface will show "CHUNK

REGISTRATION SUCCESSFUL!!!". After you have received all the chunks the interface will show "DOWNLOAD SUCCESSFUL!!!".

4.4 EXIT

If you choose the exit option, the interface will show "YOU HAVE BEEN DISCONNECTED FROM THE SERVER!!!". From this point onward you will no longer be able to upload, search and download any file/s from the network, but you will still serve to other active clients download requests. It means now you are only running as a server. So, other clients can download data you have already uploaded to the network from you.

5 Limitations of Current System

The following are the limitations of the network in most recent build:

- Client can only search for one file at a time. However, this is easy to implement because if I want to allow clients to search for multiple files. I will use the same logic I used for allowing multiple file uploads here in searching and print results for each file.
- Similarly, I only allow clients to download one file at a time. This is also easy to implement because I can use same logic I used for upload here.
- For now only ".txt" are allowed to be uploaded and downloaded in the network.
- The system is not fault tolerant as of now. So, if the server crash all information will be lost and if the client crash while downloading or uploading, in this case not all the information will be lost but the system will crash and the server metadata won't be up to date.

One way to make system fault tolerant would be to write log files on both server and client side.

Lets first talk about the server side, whenever we receive an update from any client and if it changes the metadata of server we log that information in a "log.txt" file. Later, if the server crashes and when we re-run the server and if there is data in the server log file we load that data into the metadata of the server and start serving clients.

Now, for the clients, we write the log of files the client uploaded to the network, most recent service the client asked for, the clients IP address, clients listening port. So, the next time client with the same IP address joins the network the server will make sure that they have same listening port and they still have the files they uploaded to the network. Once the server verify all this information only then the client will continue to perform the service they were using when they crashed.

6 Sample Output

```
[Devens-MacBook-Pro:server deven$ ./server
 SERVER RUNNING!!!
```

Figure 1: Right after you start running the server (No clients have connected yet)

```
[Devens-MacBook-Pro:client deven$ ./client 8081
 Connected to server

 PLEASE ENTER YOUR CHOICE:
 1. UPLOAD FILE/S
 2. SEARCH FOR FILE/S
 3. DOWNLOAD FILE/S
 4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
 Enter your choice:
```

Figure 2: Right after you start running the client

As you can see in figure 2, the client will provide a listening port as an argument and then he will get to choose what kind of service does he want from server or other clients.

In figure 3 the client is uploading 2 files to the network and their names are separated by a single space and they receive message from server that their files have been uploaded successfully.

Figure 4 shows that another client has connected to the network and he is searching for file with the name "file3.txt", which was uploaded by first client as we can see in figure 3. As you can see in figure 4, the search results show that first client have all the chunks of "file3.txt". Please note that the output is not aligned with the text headings because of difference in text size. (FILE_SIZE is 449).

Figure 5 shows the output of Client 2 downloading file from Client 1. Since client 1 was the only client who has "file3.txt", each and every chunk will be downloaded from client 1 as seen in the figure. Also, after every chunk has been downloaded the client will register its chunk and server will approve and hence we see "CHUNK REGISTRATION SUCCESSFUL!!!". Also, after the download has completed we can see "DOWNLOAD SUCCESSFUL!!!" message in the end.

Figure 6 shows the output of Client 3 downloading file from Client 1 and Client 2 because now both the clients have "file3.txt". As seen in the figure, Client 3 will download chunks from client 1 and client 2 simultaneously and than register the chunks.

```
Devens-MacBook-Pro:client deven$ ./client 8081
Connected to server

PLEASE ENTER YOUR CHOICE:
1. UPLOAD FILE/S
2. SEARCH FOR FILE/S
3. DOWNLOAD FILE/S
4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
Enter your choice: 1
ENTER THE FILE NAME (MULTIPLE FILES SPACE SEPARATED): file1.txt file3.txt
FILE/S UPLOADED SUCCESSFULLY!!!

PLEASE ENTER YOUR CHOICE:
1. UPLOAD FILE/S
2. SEARCH FOR FILE/S
3. DOWNLOAD FILE/S
4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
Enter your choice:
```

Figure 3: Client 1 is uploading 2 files to the network.

```
Devens-MacBook-Pro:client1 deven$ ./client 8082
Connected to server

PLEASE ENTER YOUR CHOICE:
1. UPLOAD FILE/S
2. SEARCH FOR FILE/S
3. DOWNLOAD FILE/S
4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
Enter your choice: 2
ENTER THE FILE NAME TO SEARCH: file3.txt
SEARCHING FOR FILES...
FILE_NAME       CHUNK_NO        FILE_SIZE       CLIENT_IP       CLIENT_PORT
file3.txt       0       449     16777343        37151
file3.txt       1       449     16777343        37151
file3.txt       2       449     16777343        37151
file3.txt       3       449     16777343        37151
file3.txt       4       449     16777343        37151
file3.txt       5       449     16777343        37151
file3.txt       6       449     16777343        37151
file3.txt       7       449     16777343        37151
SEARCH COMPLETED!!!
CHOOSE OPTION 3 TO DOWNLOAD FILE/S

PLEASE ENTER YOUR CHOICE:
1. UPLOAD FILE/S
2. SEARCH FOR FILE/S
3. DOWNLOAD FILE/S
4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
Enter your choice:
```

Figure 4: Client 2 is searching for file3.txt

```
PLEASE ENTER YOUR CHOICE:
1. UPLOAD FILE/S
2. SEARCH FOR FILE/S
3. DOWNLOAD FILE/S
4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
Enter your choice: 3
ENTER FILE TO BE DOWNLOADED: file3.txt
DOWNLOADING FILE file3.txt...
Connected to client 0 for downloading
DOWNLOADING CHUNK 0 FROM CLIENT 0
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 1 FROM CLIENT 0
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 2 FROM CLIENT 0
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 3 FROM CLIENT 0
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 4 FROM CLIENT 0
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 5 FROM CLIENT 0
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 6 FROM CLIENT 0
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 7 FROM CLIENT 0
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOAD SUCCESSFUL!!!

PLEASE ENTER YOUR CHOICE:
1. UPLOAD FILE/S
2. SEARCH FOR FILE/S
3. DOWNLOAD FILE/S
4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
Enter your choice:
```

Figure 5: Client 2 is downloading the file "file3.txt" from Client 1

```
Devens-MacBook-Pro:client2 deven$ ./client 8083
Connected to server

PLEASE ENTER YOUR CHOICE:
1. UPLOAD FILE/S
2. SEARCH FOR FILE/S
3. DOWNLOAD FILE/S
4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
Enter your choice: 3
ENTER FILE TO BE DOWNLOADED: file3.txt
DOWNLOADING FILE file3.txt...
Connected to client 0 for downloading
Connected to client 1 for downloading
DOWNLOADING CHUNK 0 FROM CLIENT 0
DOWNLOADING CHUNK 1 FROM CLIENT 1
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK FROM CLIENT 1 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 2 FROM CLIENT 0
DOWNLOADING CHUNK 3 FROM CLIENT 1
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK FROM CLIENT 1 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 4 FROM CLIENT 0
DOWNLOADING CHUNK 5 FROM CLIENT 1
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK FROM CLIENT 1 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOADING CHUNK 6 FROM CLIENT 0
DOWNLOADING CHUNK 7 FROM CLIENT 1
CHUNK FROM CLIENT 0 DOWNLOADED SUCCESSFULLY!!!
CHUNK FROM CLIENT 1 DOWNLOADED SUCCESSFULLY!!!
CHUNK REGISTRATION SUCCESSFUL!!!
DOWNLOAD SUCCESSFUL!!!

PLEASE ENTER YOUR CHOICE:
1. UPLOAD FILE/S
2. SEARCH FOR FILE/S
3. DOWNLOAD FILE/S
4. EXIT (YOU WILL NOT BE ABLE TO USE ANY OF THE SERVICES)
Enter your choice:
```

Figure 6: Client 3 wants to download file "file3.txt"