Peer to Peer File Sharing Network

Abstract

The project is an attempt to implement the peer to peer sharing model. A central node stores the connection information and performs distributed file search among the connected peers. Peers get notified if the file requested exists in any of the other peers, and then get an option to download that file directly from the peer. New peers can be added to the network and can unsubscribe itself from the network.

Introduction

Peer to peer networking has recently emerged as a new paradigm for building distributed networked applications.A peer is both a producer and a consumer of the implemented service. you can say a peer generates workload for the P2P application, while also providing the capacity to process the workload requests of others. Taking these concepts in mind we have implemented the peer to peer sharing model. In our project we have a central node which stores all connection information as well as it performs the file search among the connected peers. Any node or peers can join or leave the network any time without disturbing the entire network. When a peer requests a file, using the central node it will get to know about the file it requested, whether the file exists in any of other peers or not. If it is, it gets an option to download the file directly from the peer.

Literature Review

The key points the P2P network should follow and types of P2P network system.

P2P network is mainly done to cooperatively exchange the resources kept at the different edges over the internet to be shared with users cooperatively. These systems are more robust, fault tolerant and give better performance than client-server model.

Peer-to-Peer can be of following 2 types:

- Centralized System: These systems can be considered as hybrid systems, where the social meta information about the data and peers are stored at the multiple centralized server entities. The data distribution is still organized in distributed fashion, directly from peer-to-peer.

- Decentralized Systems: In these systems, both the data discovery and data download are organized in a distributed fashion. The systems can do their resource sharing by different mechanisms such as handshaking. Decentralized Systems can be further classified as Structured and Unstructured networks. In Structured networks, there are some restrictions on network topologies and the placement of contents. In Unstructured networks, placement of data or contents are not dependent upon the network topologies and structure of the network. Unstructured networks perform better than Structured networks in the view of performance. But there are some disadvantages such using Unstructured networks such as fake content distribution, free riding and more.

There are the key points that the Peer-to-Peer network should follow:

- **Mutual Exclusion**: When two Peers in a connected network are performing any operation, then any other peer in the network should not interfere or disturb the connection or the operation which the two Peers are performing. Also other peers in the network should not know about the type of the operation or about the type of resources which the two peers are using. This comes under the security measurement.

- **Load Balancing**: There might be the situation in the network when one type of file is asked many times or it is more in demand. Then in that case a similar file is to be transferred by the owner of the file to all the peers who are demanding. So to balance the load on a single peer, it will be a great idea to replicate this file to more than one peer, so that if any of the peers asked for the file then except the owner other peers could also transfer the file when asked.

- **FIFO Communication Channel**: The communication channel between the two peers can be FIFO or non-FIFO. Both the methods have their own advantages and disadvantages. But in FIFO channels, it is guaranteed that the message passed first will be delivered before the other message. It will be helpful in making the decision to take place and peers could perform the task rather than thinking about the order.

- **Multi-Threading in Sending resources**: If more than one peer has the file which was demanded by any of the peers rather than sending the whole file by one of the peers, it could be sent by other peers who have the file in chunks means the whole file could be divided into chunks and then one part could be sent by one peer and other chunk could be sent by some other peers. Through this approach, load on the single peer is also reduced as well as operation could also be done fast.

- **Avoiding Single Point of Failure**: By replicating the files to multiple peers, the concept of Single point Failure could also be avoided as when the file was asked, if the owner was down than another peer who has the replicated version of that file could become active and can perform the operation.

- **Synchronous System**: There are two types of systems: Synchronous Systems and Asynchronous Systems. In Synchronous systems, operations are coordinated by one or more centralized clock signals. In Asynchronous systems, there are no centralized clock signals. These systems do not depend upon strict arrival times of signals or messages for reliable operation. Both the systems have their own advantages and disadvantages, synchronous systems could be preferred over others because in these types of systems, if the message is lost between the channels then after a certain threshold time period, the message for the operation could be sent again.

Proposed Methodology

- Initialization
- File Searching
- File Downloading
- Client Existing Algorithm
- File Replication

Initialization:

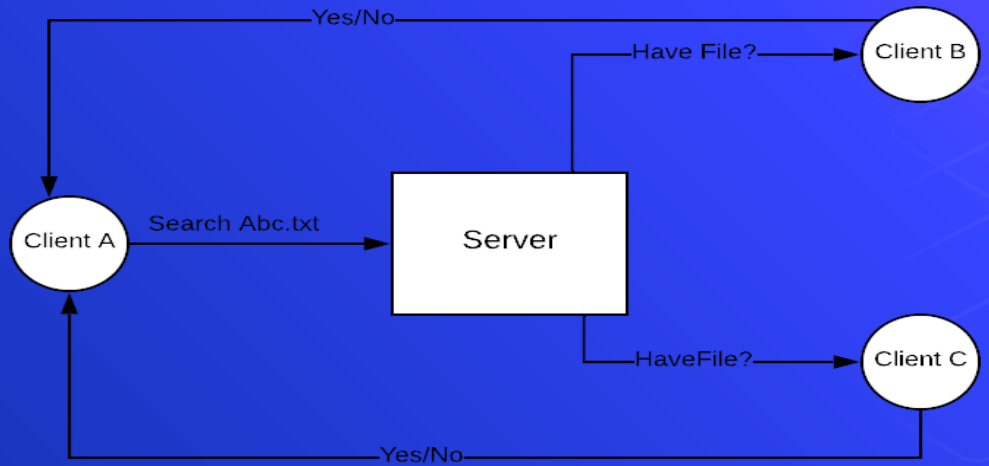
- Server is started first. It has a serverClient handler which spawns different threads for different clients and these serverClient handlers communicate between server and its client.
- Server has information regarding inputStreams, OutputStreams, locking status, for each client.
- Client when started execution , it attempts to connect to the server, this leads to creation of a new serverClient handler thread by the Server.
- Server asks the MAC address of the client and then it records that client in the database where using MAC address acts as the primary key in the 'peers' table, which stores information regarding the client.
- On successful connection, client gets a menu from which it can select the operation it wants to perform.
- The menu is as follows , "**Press: (1)** to Enter file to search | **(2)** to de-register your pc | **(3)** to download files". So these options can be selected by the client.

File Searching:

- When client presses 1 on the menu, the server asks the filename from the client, which is then sent by the client to the server.
- Server then writes “ForSearching <count>” to its output stream, which is the input stream for the client. This notifies the client that the search process for the requested file has initiated. Here <count> is the number of unlocked online clients,except the requesting one.
- Requesting client then waits for a <count> number of replies of “yes” or “no” .
- Meanwhile, the server asks other clients to search for the requested file, it sends “HaveFile? <filename> <requesting client’s ip address>” to all other clients .
- The server locks the requesting client, so that it doesn’t request other operations, while one is going on.

- The clients who receive the “HaveFile?” request, send the reply directly to the client who is requesting for the file, without server as middle-man.

- After <count> number of replies is received by the requesting client, client sends “UNLOCKME” to the server to unlock itself.



File Searching Flow Diagram

File Downloading:

- When client presses 3 on the menu, the server puts the lock on the client to preserve the mutual exclusion, and asks the client the filename to download..

- Server then writes “Connect <chunk flag>” to its outputStream which is the inputStream for the client. Here the chunk flag is the type in which the client wants to download the file. If it is 1 means the client wants the file to be downloaded in chunks else the whole file at once.

- Now the server writes to the client , the file name & the “Replication status” . Here replication status would be “Replication yes” if the client requested the same file more than 4 times. To avoid the single point failure we replicate the file to multiple peers else it would be “Replication No”.

- Server then writes “<count>” to the client. Here <count> is the number of unlocked online clients,except the requesting one. Count is basically for the requesting client so that it waits for replies from <Count> number of other clients.

File Downloading Contd:

- Server asks each client except the requesting one , whether they have the file? If they have it, then they should start miniserver listening on port N1 & N2 (number of ports depending on chunking flag) waiting for the requesting client to connect to them to download the file.

- Each Client maintains three lists, one to store ip addresses of clients which don't have the file , and other to store ip addresses & ports of clients which have the file respectively.

- If more than one client has the requested file and the chunking flag is 0, A new thread is spawned which connects to the client having the file through socket on port N1 and receives the whole file. If the chunking flag is 1. Two new threads are created, one for the first half of the file and one for the last half which listen through different port numbers (N1 & N2) . After getting both parts of the file the requested client merges both parts of the file to get the file.

Client Exiting Algorithm:

- When the client requests to exit the network, the user presses ‘2’ in the menu.
- On receiving ‘2’ from the client , the server sends an “Exit” message to the Client, and after sending the message, the server removes the entry of that client from all its client data structures as well as the database.
- On receiving the “Exit” messages, the client terminates.

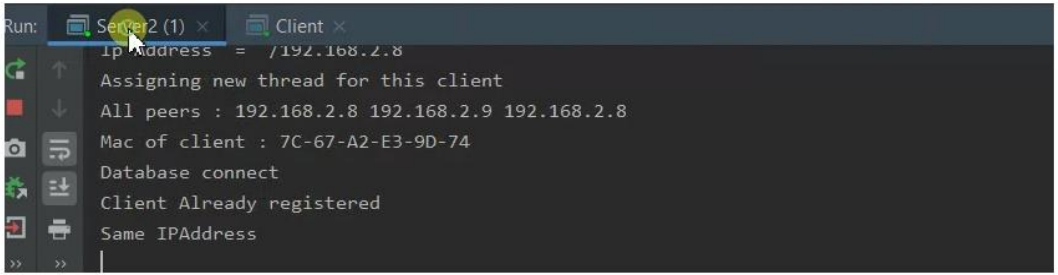
File Replication:

- When a client is asking to get a file transferred to it multiple times, (we used a threshold of 4), then it indicates that this particular file is in high demand.
- The ServerClient handler has this information regarding the number of times a file is requested to be downloaded by a client stored in a hash table “FileCount” for each client.
- The requesting client also sends “HaveFile?” request to all other clients to check if they have the file, based on the replies , it builds a list of clients who have the file and those who don’t.
- Now this client sends a request to the server to replicate the file on one another client (say with ip address repId) which doesn’t already have the file. This will eliminate single point of failure on some highly demanded file.
- The server asks the client with IP address repId to download that file with the file download methodology mentioned above and store it in its local system, with other flag values such that no further replication occurs.

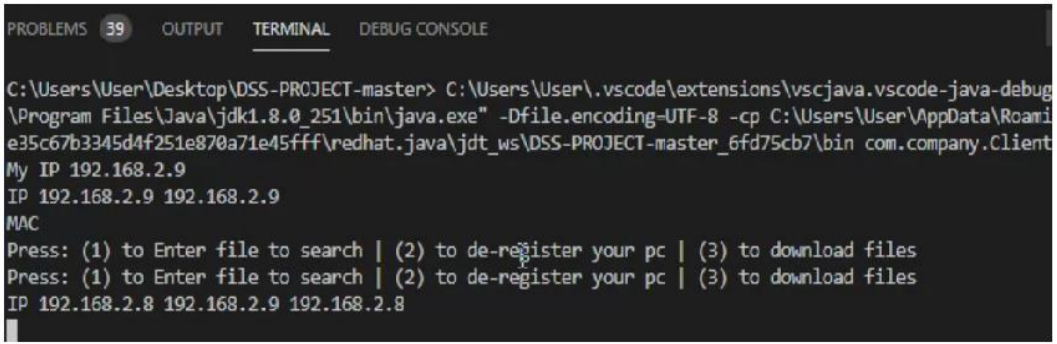
Results

output taking all the possibilities

❖ We start a server and 2 clients , so both are connected to the server and give the user the list of services.

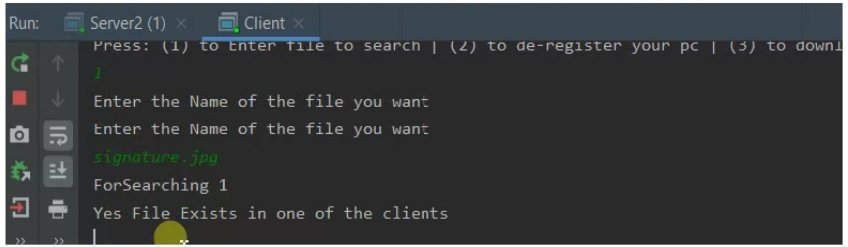


SERVER & CLIENT-1

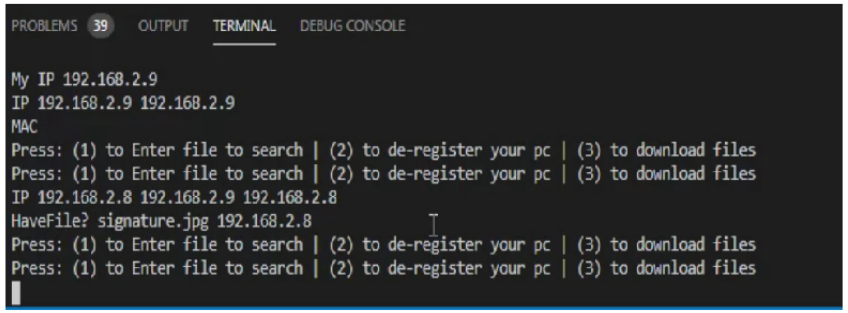


CLIENT-2

❖ Client-1 presses ‘1’ to search a file.



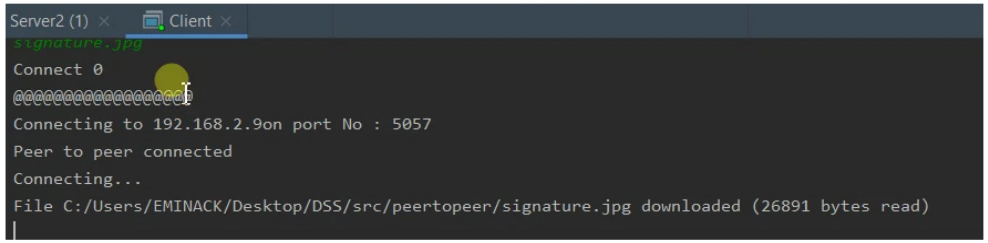
Client-1 broadcast search for *signature.jpg* and receives YES from Client-2



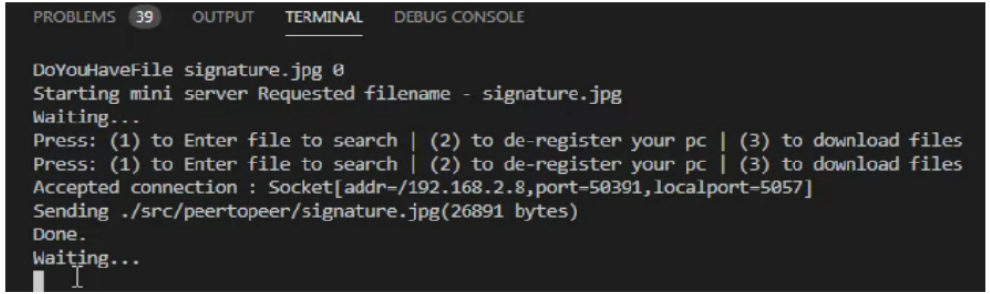
Client-2 receives HaveFile? *signature.jpg* , so it searches it and replies with Yes.

- Client-1 presses ‘3’ to download a file. The CHUNKING_FLAG = 0 so the file is downloaded in one piece.

CHUNKING_FLAG = 0



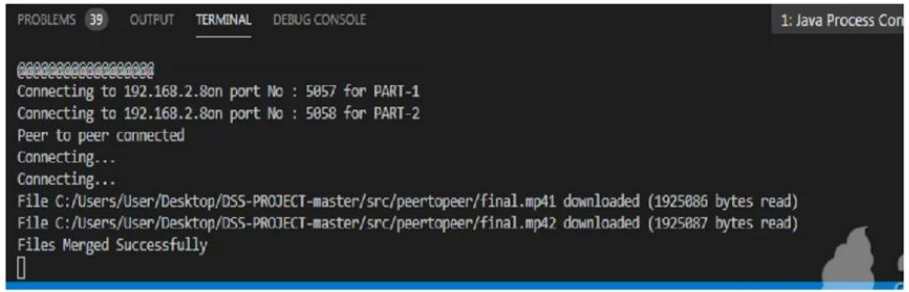
Client-1 wants *signature.jpg* , so it broadcasts transfer request to every client & downloads it from Client-2 by directly connecting to it.



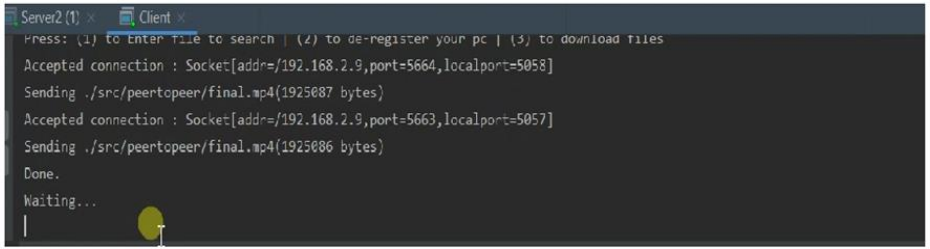
Client-2 receives transfer request from Client-1, so starts a miniserver & waits for Client-1 to connect, then transfers the file.

- Client-2 presses ‘3’ to download a file. The CHUNKING_FLAG = 1 so the file is downloaded in chunks over 2 different connection and then merged.

CHUNKING_FLAG = 1



Client-2 requests file *final.mp4* , connects to Client-1 using 2 connections on different ports & receives file in 2 parts, each part from a connection and then merges the received part to construct the original file.



Client-1 receives transfer request from Client-2 & starts 2 miniserver on different ports to transfer the file in 2 chunks,waits for connection from Client-2 , then transfers the chunks.

References:

[1] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," Proceedings First International Conference on Peer-to-Peer Computing,Linkoping,Sweden,2001,pp.

[2] J. Sen, "Peer-to-peer networks," 2012 3rd National Conference on Emerging Trends and Applications in Computer Science, Shillong, 2012, pp.

[3] A. Hac, “A distributed algorithm for performance improvement through file replication, file migration, and process migration” , “ IEEE Transactions on Software Engineering” , 2009

Thanks!

Aditya Raj
IIT2017005

Sameer Kathal
IIT2017010

Nishant Kumar
IIT2017023

Manish Kumar Jagnani
IIT2017503