UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

UNDERGRADUATE UNIVERSITY

Course: Distributed System

Final Group Project

Peer to Peer File Transfering System

Submitted by Group 9

Group Member

Do Thi Minh Ngoc	USTHBI8-129
Nguyen Trong Son	USTHBI8-153
Do Anh Tu	USTHBI8-157
Nguyen Trung Dung	USTHBI8-029
Nguyen Khanh Nam	USTHBI8-122
Pham Viet Minh Duc	USTHBI8-045

Lecturer: Dr.TRAN Giang Son

Hanoi, January 2020

Table of content

1.Introduction

1.1 What is Peer-to-Peer file sharing?

Peer to peer File Sharing systems are no longer just a new fad technology. They have become ingrained in our Internet culture. It is a process of sharing and transferring digital files from one computer to another. In the p2p network, each ‘peer’ is an end-user’s computer connected to the other ‘peer’ via the Internet – without going through an intermediary server.

1.2 Why use Peer-to-Peer file sharing ?

P2P programs can be an efficient way to share large files with others, such as personal video recordings or large sets of photos. P2P is also used to facilitate direct communications between computer or device users. Chances are you’re already using some form of P2P technology: for example, Skype built its communications systems on P2P technology.
P2P file sharing could play an even larger role in your computer activities in the future.

2.Objectives

We created this project for the purpose of learning how to build a file transfer system based on peer to peer networks. From this construction we gain more knowledge regarding distributed systems, connections between devices and communication networks.

In this project, our goal is to be able to transfer files directly between two different users via localhost without the need for any third party. These files can be messages, or files such as images, videoclips, text documents, etc.These messages can also be performed in parallel from one user to many other users. connect with them.

3.Existing Systems

There are many peer-to-peer file sharing applications that are able to share files between devices over the network. Some of these applications work with computers and some on mobiles. However, peer-to-peer applications cause heavy traffic on the internet. BitTorrent, Gnutella, Napster are some of the most popular applications in peer-to-peer systems.

a. BitTorrent

BitTorrent is a peer-to-peer file sharing protocol designed by Bram Cohen and is one of the most popular file sharing systems in the past few years used to distribute data content. BitTorrent shares the same file between nodes over the internet instead of different files. BitTorrent is a peer-to-peer distribution network and this type of network uses the bandwidth of the user.

b. Napster

The architecture of Napster is based on the Centralized Model of P2P file-sharing. It has a Server-Client structure where there is a central server system which directs traffic between individual registered peers. The advantage of the Napster network is that the query is efficient and finding the files guaranteed. On the other hand, the peers depend on a central server, if the server goes down, the network will become useless.

c. Gnutella

Gnutella is one of the most popular peer-to-peer file sharing systems to date used to exchange and share data between the peers. There is no central point or server in the Gnutella network as opposed to centralized systems such as Napster. The main advantage in Gnutella is that there is no central point in the network. It depends on flooding in routing queries.

4. Method

a. System Architecture

The system is based on peer to peer networks. In a peer-to-peer network, the "**peers**" are computer systems (or an unique address) which are connected to each other via the Internet. Files can be shared directly between systems on the **network** without the need of a central server. In other words, each computer on a Peer-to-Peer network becomes a file server as well as a client.

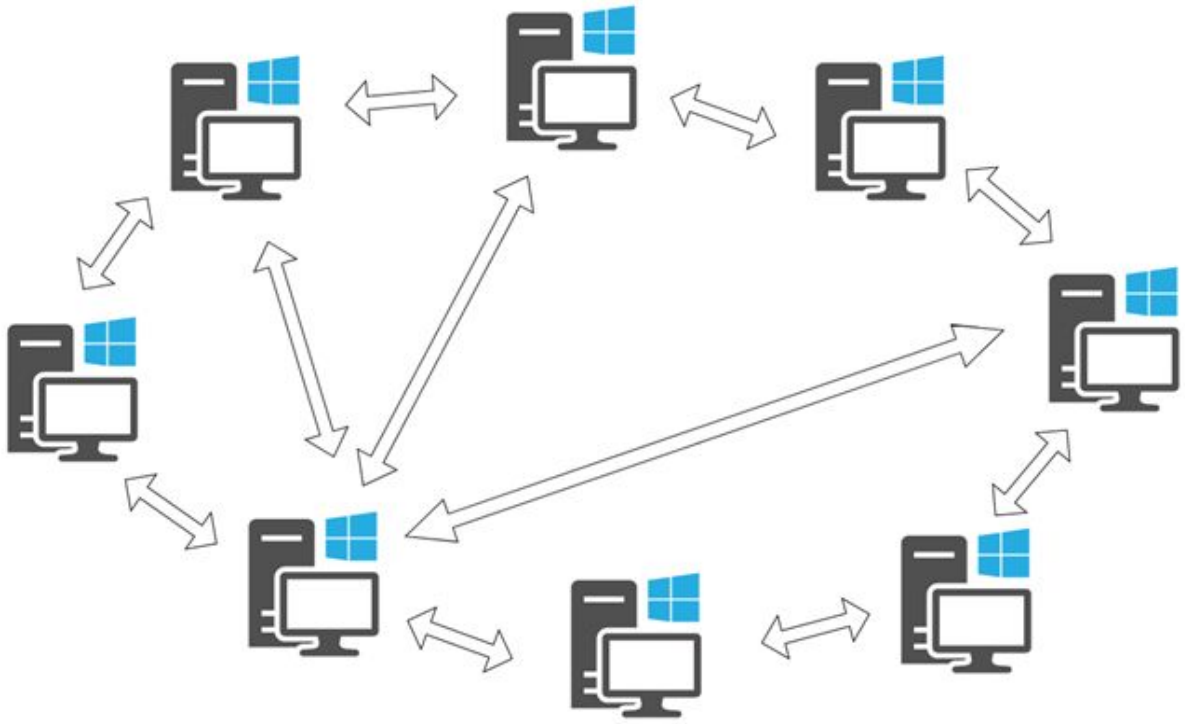

Figure 4.1 Peer to Peer Network architecture

Our system allows the users to represent themselves as peers and connect to other peers directly through the local network.

To make it more clear, let's take a look at an example of connection between 3 peers: Peer1, Peer2, Peer3. In this example:

- Peer1 is the client of Peer2 - Peer2 is the server of Peer1
- Peer1 is the client of Peer3 - Peer3 is the server of Peer1
- Peer2 is the client of Peer3 - Peer3 is the server of Peer2

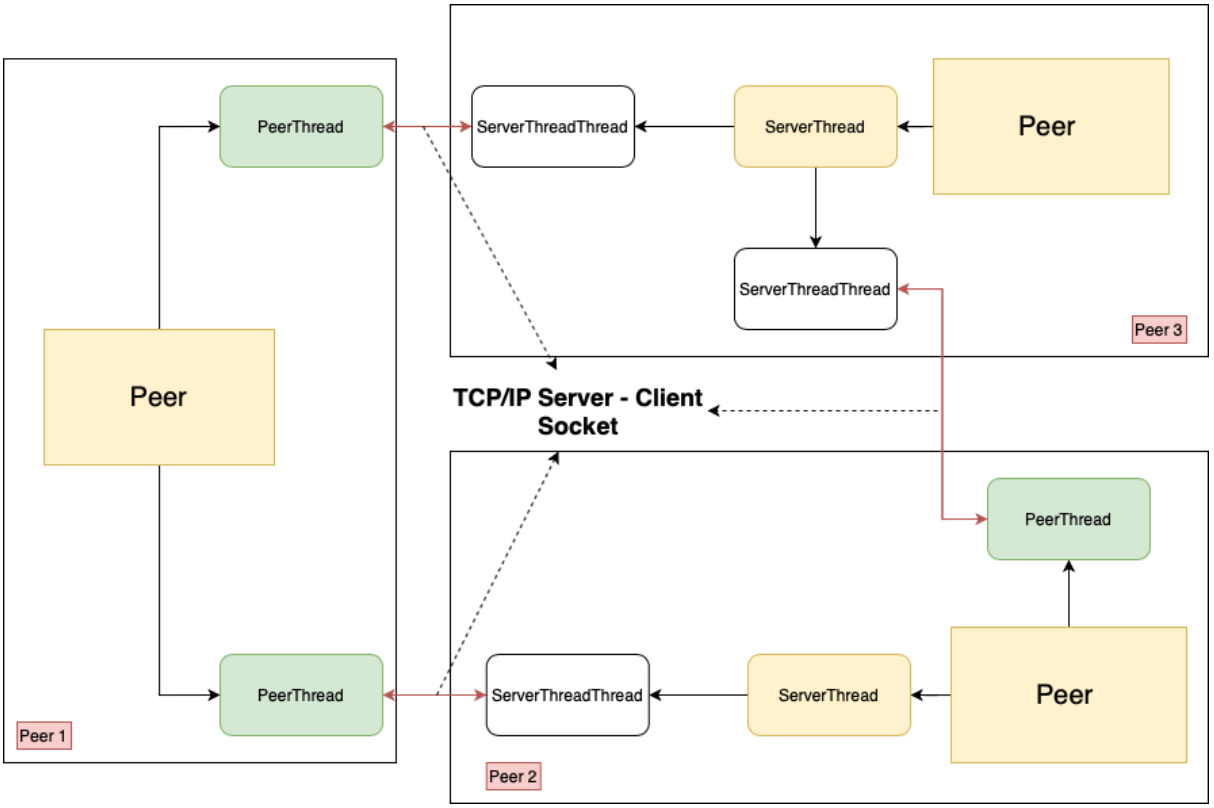

Figure 4.2 Connection example

As you can see from the figure, a peer acts like a client when it contains a PeerThread and communicate with a ServerThreadThread of another peer (Peer1, Peer2), it act like a server when it contains the ServerThreadThread communicate with a PeerThread of another peer (Peer3, Peer2).

So, a peer can be both client and server when it contains both of these threads (Peer2).

b. System implementation

The application is built using **Eclipse IDE for Java Developer** and an external library JSON.simple .

i. Peer.java

This Java class contains the main method for the program. It allows the user to create a peer with its port. This class will interact with users, each peer representing an individual user. From this peer, users can interact with other peers by receiving messages or files and also send files or allow other users to read their message. After the user initiates their name and PORT, this class calls ServerThread class to create a ServerSocket with the provided PORT.

ii. PeerThread.java

This class is called by the Peer class each time a user chooses to connect to other peers. Each Peerthread contains a client socket which has the same PORT with the Server Socket of the peer that the user wants to get data from. This class handles data from the Server Socket through InputStream, processes it and returns to the user's screen.

iii. ServerThread.java

This class is also called by the Peer class when the user successfully initiates their PORT. This class responds when a client connects to the ServerSocket and handles the data that the user wants to transfer through the ServerSocket(message or file). Each time a peer connects to the ServerSocket, the ServerThread creates a new thread (ServerThreadThreads) and puts it in a HashSet. By doing this, the ServerSocket can serve multiple peers simultaneously in individual threads.

This class provides functions for Peer class to parse data (message or file) to ServerThreadThreads for OutputStream.

iv. ServerThreadThreads.java

This final class is the thread created by the ServerThread class in order to handle data for OutputStream on ServerSocket for individual peer connection.

- PrintWriter printWriter: During the running state of the thread, if the thread don't have to handle file sending, it will handle message with this printWriter under the form of JSON object with the format:

```
[ "username" : sender , "message" : message , "code" : tokenizer ]
```

- String filePath: set to null at initial stage. During the running state of the thread, if this variable changes, the thread will run the algorithm to send the file. The thread will end after the file is delivered.

5. Evaluation

Response Time

The response time calculation is executed in 1 computer only due to lack of devices. Peers create threads to handle connections, the ability to run threads smoothly depends partly on the computer's hardware and normally each computer represents just 1 peer.

The tests for response time are performed under 3 scenarios:

- Case 1: 11 peers listen to 1 peer: 22 threads
- Case 2: 3 peers listen to 1 peer: 6 threads
- Case 3: 1 peer listen to 1 peer. 2 thread

	Avg. 1st Msg Response Time	Avg. 2st Msg Response Time	Avg. 3st Msg Response Time	Avg. 4st Msg Response Time	Avg. 5st Msg Response Time
Case 1	467.1 ms	5.5 ms	4.6 ms	6.55 ms	4.64 ms
Case 2	108.3 ms	2 ms	3.67 ms	2.67 ms	2.33 ms
Case 3	35 ms	3 ms	1 ms	1 ms	1 ms

The more connections a peer has, the slower it is. The first message takes the longest to respond but all the messages after that will have stable response time and it can't not be considered as slow even in the hardest scenario test.

6. Conclusion

The system has achieved the initial goals of the project, including:

1. The system is based on peer-to-peer network architecture.
2. Transfer files directly between peers, without the need for third parties.
3. Files can be transferred from one peer to many peers simultaneously.
4. Accept multiple file types

Also,we have gained a lot of knowledge about computational systems as well as connection protocols. In the future, this project may be further developed to further improve its functionality and user interface.