

YORK UNIVERSITY

Lassonde School of Engineering



By:- Vishal Malik



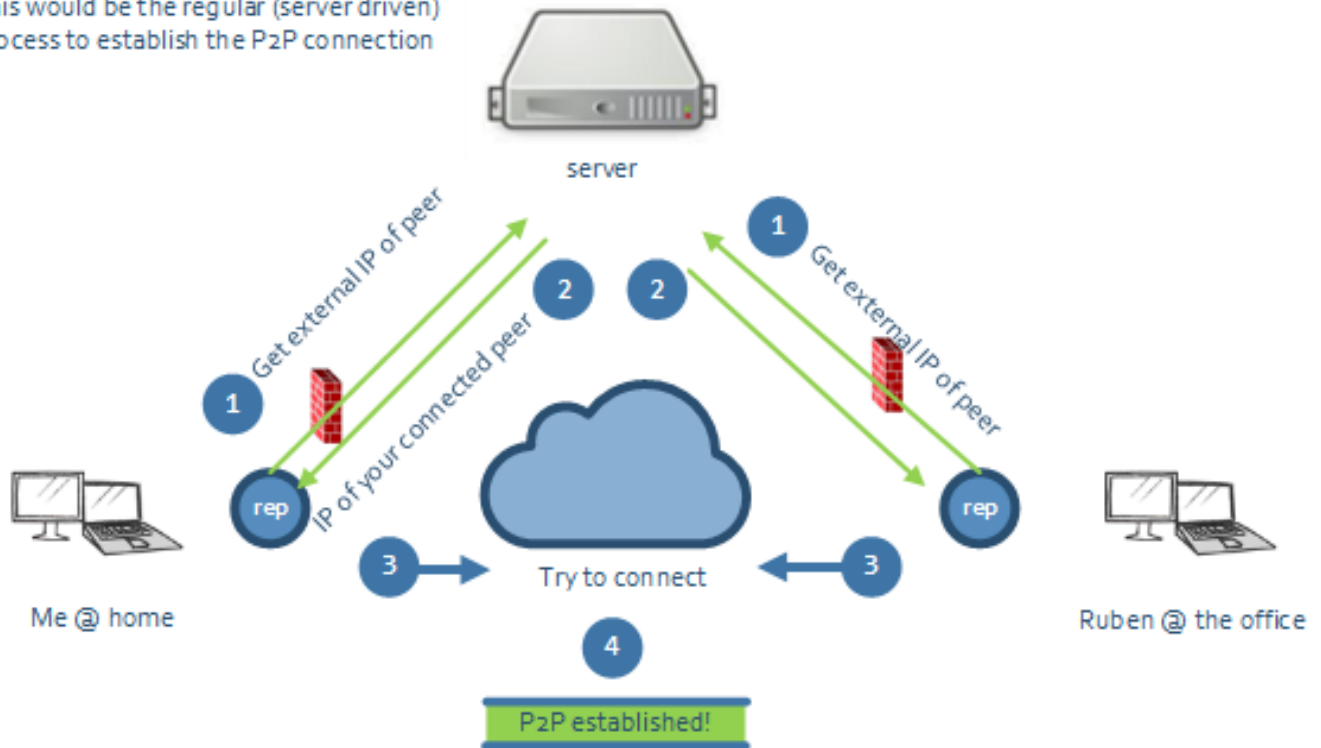


Peer to Peer

This application consists of two separate programs: Client.java and Server.java. In which Server.java is a Concurrent Server that uses multithreading to run multiples client at the same time. In this server runs indefinitely and accepts Client Connections, terminates clients and make a List of all online or say connected client whereas Client part of application is for end users in which it is runs of host machine and connect to server just to asks for the List of Client which are connected, after which it opens a direct connection which the Client it want to talk. Any Client can open a connection with other by putting their port Number and IP address. The basic outlay

design of this model is as below

This would be the regular (server driven)
process to establish the P2P connection

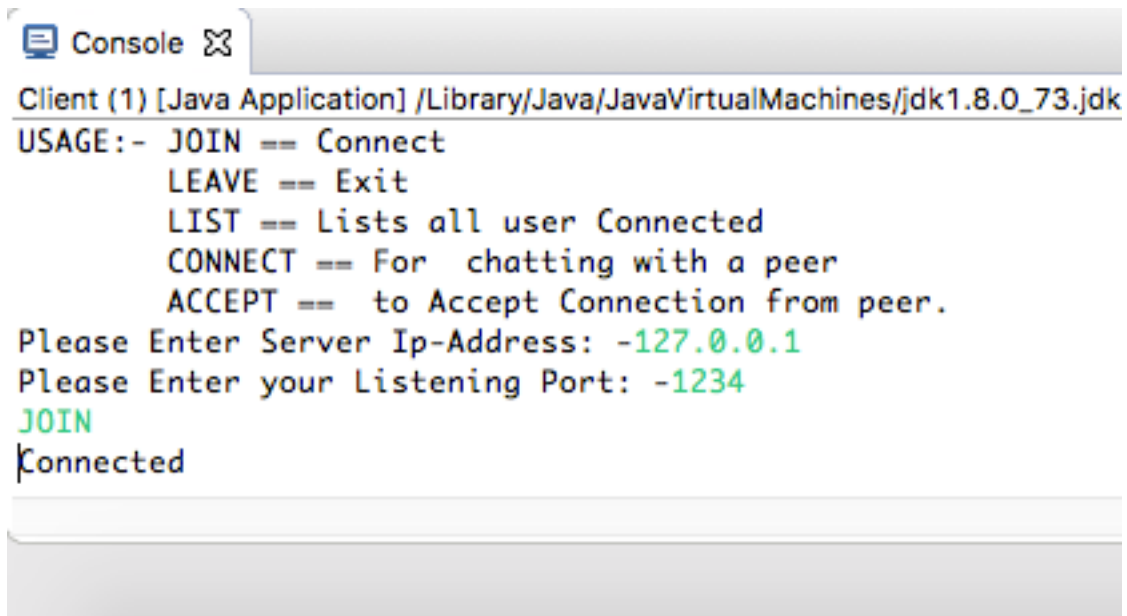


Usage

Initially the Server class is must be launched before clients and waits for clients to connect. The main information that a server need is port number, IP- Address. Which can be easily passed from command prompt

```
Console
Server (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/Contents
Server up and running waiting for connections....
```

This indicates that the clients can be run now to connect to the server. Each client needs to know the port number on which they will listen and the IP address of the server. The commands must be entered exactly as the prompts suggest:



```
Client (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk
USAGE:- JOIN == Connect
        LEAVE == Exit
        LIST == Lists all user Connected
        CONNECT == For chatting with a peer
        ACCEPT == to Accept Connection from peer.
Please Enter Server Ip-Address: -127.0.0.1
Please Enter your Listening Port: -1234
JOIN
Connected
```

When the client has connection to server, it can use ‘JOIN’ command to connect to server the “LEAVE” command initiates an attempt to disconnect from the server using the previously entered data User name. If the attempt successful, the following prompt will appear:

```
        LIST == Lists all user Connected
        CONNECT == For chatting with a peer
        ACCEPT == to Accept Connection from peer.
Please Enter Server Ip-Address: -127.0.0.1
Please Enter your Listening Port: -1234
JOIN
Connected
LEAVE
You Are Disconnected
```

If “LIST” command is used, then it will print out all online users or current users of the program and the output will look similar to the prompt below

```
Console ✕
Client (1) [Java Application] /Library/Java/JavaVirtualMachines/jd
Please Enter Server Ip-Address: -127.0.0.1
Please Enter your Listening Port: -9876
JOIN
Connected
LIST
Ip Address:- 127.0.0.1 PortNumber:- 1234
Ip Address:- 127.0.0.1 PortNumber:- 4567
Ip Address:- 127.0.0.1 PortNumber:- 9876
```

If “CONNECT” command is used, then it will prompt for port number of peer you want to connect and IP address of it and it looks like this

```
Client (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.
Connected
LIST
Ip Address:- 127.0.0.1 PortNumber:- 1234
Ip Address:- 127.0.0.1 PortNumber:- 4567
Ip Address:- 127.0.0.1 PortNumber:- 9876

CONNECT
Please Enter your friends portNumber: -1234
Please Enter your friends Ip Address: -127.0.0.1
```

Once the information provided it will ask that peer whether it wants to Accept connection or not

```
Client (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java
USAGE:- JOIN == Connect
        LEAVE == Exit
        LIST == Lists all user Connected
        CONNECT == For chatting with a peer
        ACCEPT == to Accept Connection from peer.
Please Enter Server Ip-Address: -127.0.0.1
Please Enter your Listening Port: -1234
Join
JOIN
Connected
You have a request from a peer. Accept?
ACCEPT
|
```

If it types “ACCEPT” the connection is live and now the Client which instantiated the Connection will message first remember it’s not concurrent so each client has to wait to other to message. Its look similar to this

<pre>Client (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java USAGE:- JOIN == Connect LEAVE == Exit LIST == Lists all user Connected CONNECT == For chatting with a peer ACCEPT == to Accept Connection from peer. Please Enter Server Ip-Address: -127.0.0.1 Please Enter your Listening Port: -1234 Join JOIN Connected You have a request from a peer. Accept? ACCEPT </pre>	<pre>CONNECT Please Enter your friends portNumber: -1234 Please Enter your friends Ip Address: -127.0.0.1 hello 1234 >how are you i am good</pre>
---	--

After they connected they can talk and any client can end the connection by using “LEAVE”.

Overview and Possible Improvements

The program seems to be free of any serious bugs and appears to be working as intended in OSX and Linux environments. In the current version, its only a console based application, if further time was given then the possible improvement would have a GUI interface. The only bug in the program is that sometimes the program when in chat session with other client treat “LEAVE” command as message instead of command and hence sometimes leaves program unresponsive so try to restart the program. Another possible enhancement for peer clients can be about concurrent client threads which will allow both to chat to each other without any constraint and each client can send message at same time, in this version it is not concurrent so each client have to wait for other to respond first. The port number for server connection and client are hardcoded in this program to avoid using the same port as this version have no protection against it and hence that will crash the program with an exception about the thread is currently in use, these could be a possible improvement is the design.

So far there are not security measures have been taken in the program and anyone can run and leave it open for day which in case can cause server outrage and crash the server. So far this is the only way to connect to the server’s list. A possible improvement would be to allow the server administrator to have more control over the LIST of clients. The administrator could, for example, be given the ability to add and delete clients from the list in between chats. Alternatively, the administrator could create the list of clients numbers at start of the program

Compiling

The programs client.java and server.java compile as follows:

There are actually four classes Server.java ServerThread.java Client.java ClientThread.java and ClientObject.java

Server and ServerThread classes are for Server in Server.java is the one which we compile and in Client and Client Thread we will compile Client as ClientThread is there for only Peer to Peer connection can automatically instantiate inside Client Class. The ClientObject class is for server to store the list of Clients

The following are the Steps

→ Javac Server.java

→ Java Server

→ Javac Client.java

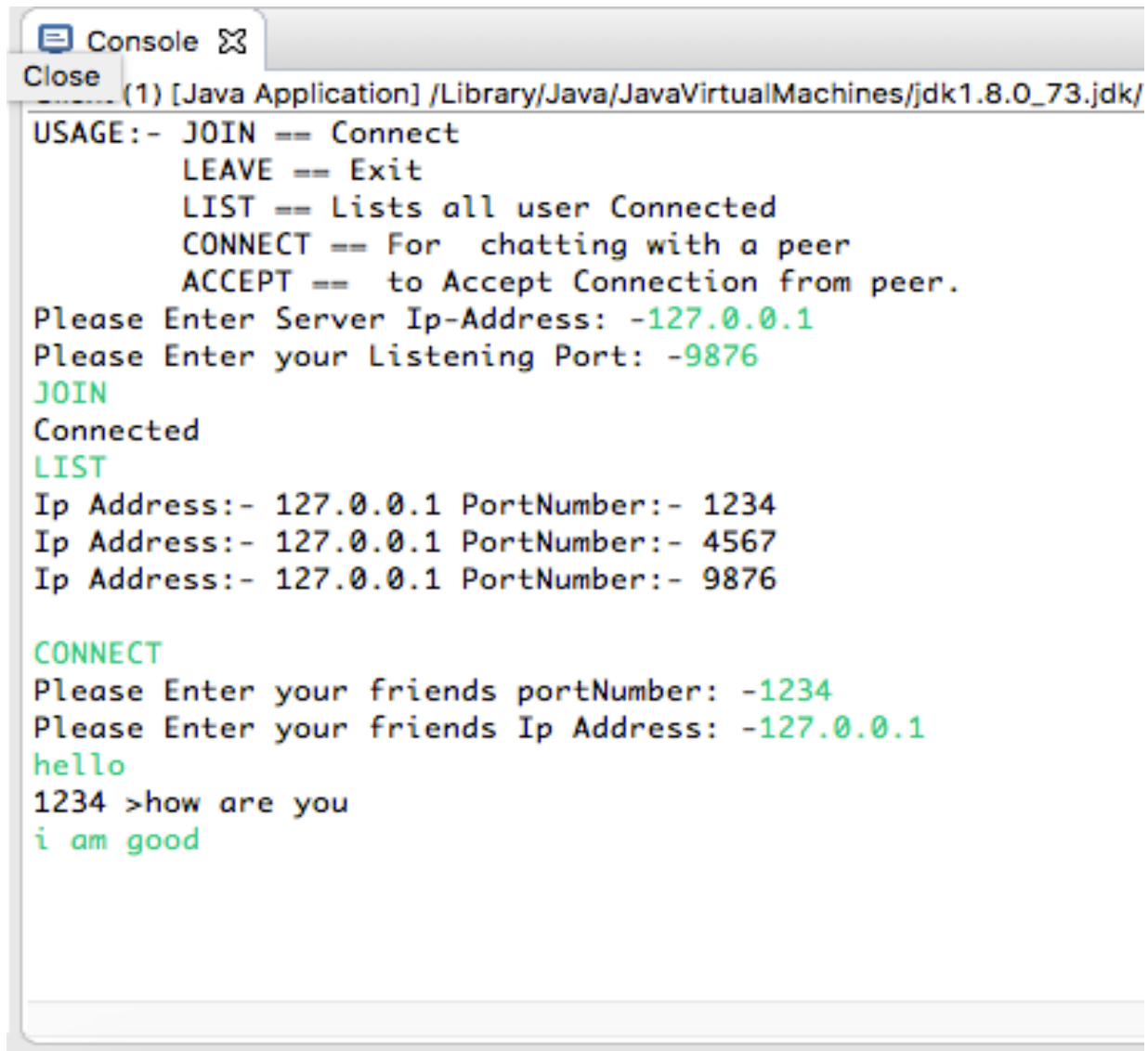
→ Java Client

The Client Files requires information about IP address of server which in our case is Local Host

And it will ask you Listening Port which other peers use to connect to you

I/O sample

An Input Output sample for program. In this example there are two clients which joins and talk to each other call all Commands such as LIST to LIST all users and LEAVE to exit the chat. In the second Screenshots it shows how the client notify the server that it exited the chat



```
Console [X]
Close (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/
USAGE:- JOIN == Connect
        LEAVE == Exit
        LIST == Lists all user Connected
        CONNECT == For chatting with a peer
        ACCEPT == to Accept Connection from peer.
Please Enter Server Ip-Address: -127.0.0.1
Please Enter your Listening Port: -9876
JOIN
Connected
LIST
Ip Address:- 127.0.0.1 PortNumber:- 1234
Ip Address:- 127.0.0.1 PortNumber:- 4567
Ip Address:- 127.0.0.1 PortNumber:- 9876

CONNECT
Please Enter your friends portNumber: -1234
Please Enter your friends Ip Address: -127.0.0.1
hello
1234 >how are you
i am good
```

Console

Client (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/Contents

USAGE:- JOIN == Connect

LEAVE == Exit

LIST == Lists all user Connected

CONNECT == For chatting with a peer

ACCEPT == to Accept Connection from peer.

Please Enter Server Ip-Address: -127.0.0.1

Please Enter your Listening Port: -1234

Join

JOIN

Connected

You have a request from a peer. Accept?

ACCEPT

9876 >hello

how are you

9876 >i am good

|