CSE2005 - OPERATING SYSTEMS
DIGITAL ASSIGNMENT 2

PREPARED BY - HARGUR PARTAP SINGH BEDI (15BCE1257)

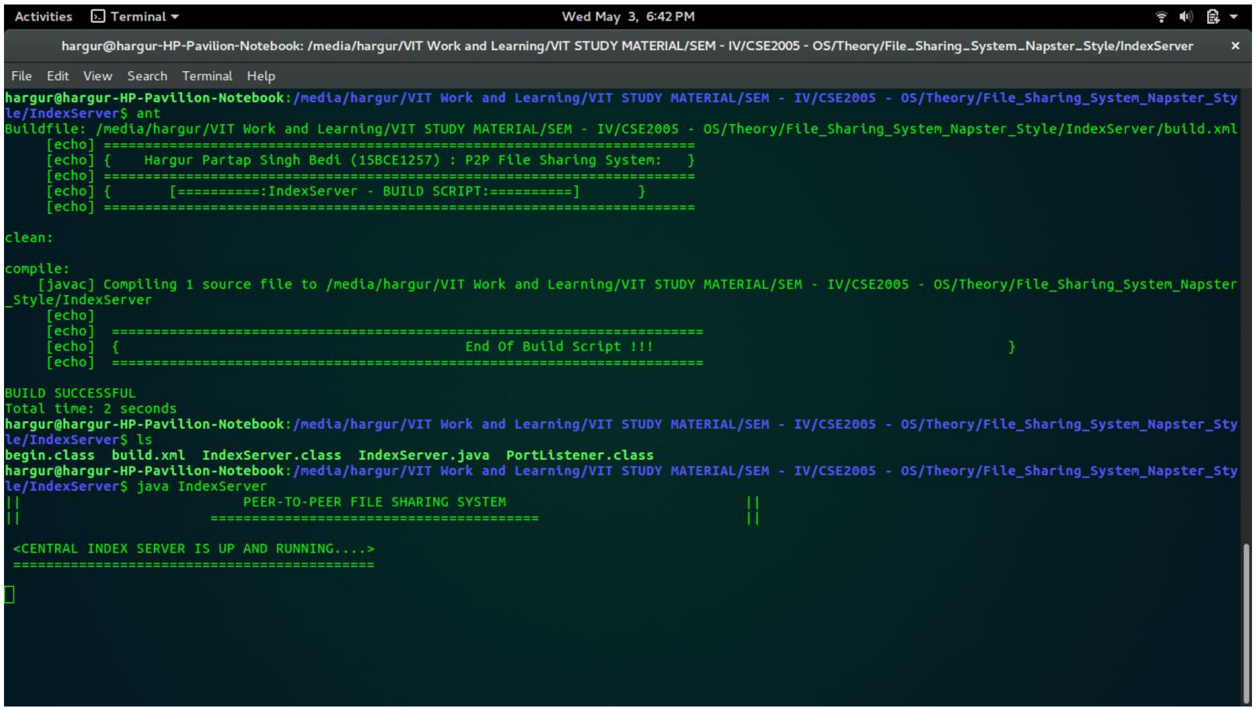
Output File

Step-by-Step Execution:

On Machine-A, **IndexServer** is Run.

The CIS starts listening for incoming connections from the Peers.

Screenshot when **IndexServer** is Running:

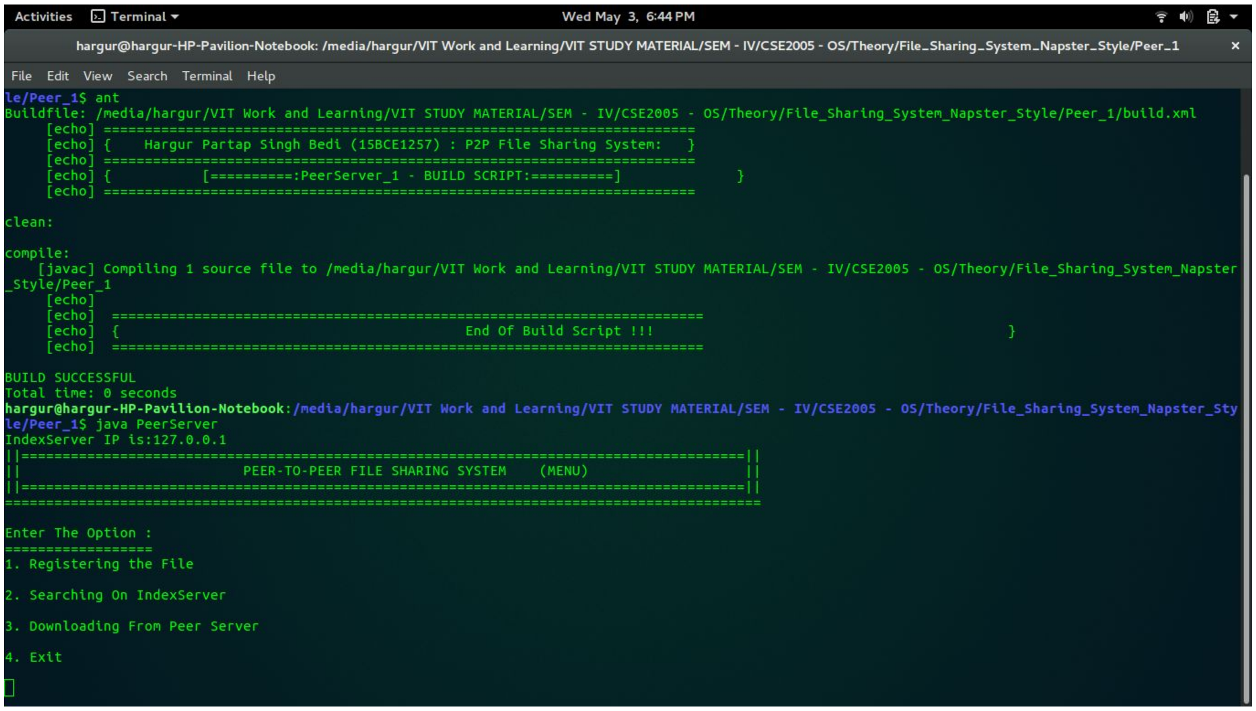


```
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/IndexServer$ ant
Buildfile: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/IndexServer/build.xml
     [echo] ======================================================================
     [echo] {    Hargur Partap Singh Bedi (15BCE1257) : P2P File Sharing System:   }
     [echo] ======================================================================
     [echo] {       [==========:IndexServer - BUILD SCRIPT:==========]       }
     [echo] ======================================================================

clean:

compile:
    [javac] Compiling 1 source file to /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster
_Style/IndexServer
     [echo]
     [echo]  ======================================================================
     [echo]  {                                          End Of Build Script !!!                                          }
     [echo]  ======================================================================

BUILD SUCCESSFUL
Total time: 2 seconds
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/IndexServer$ ls
begin.class  build.xml  IndexServer.class  IndexServer.java  PortListener.class
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/IndexServer$ java IndexServer
||                           PEER-TO-PEER FILE SHARING SYSTEM                             ||
||                       ========================================                         ||

 <CENTRAL INDEX SERVER IS UP AND RUNNING....>
 ============================================
```

On Machine-B, **Peer_1** is Run.

Peer_1 is allowed to Register its files with the **IndexServer**, Search the CIS for required files or Download from other peers.

Screenshot when **Peer_1** is Run:

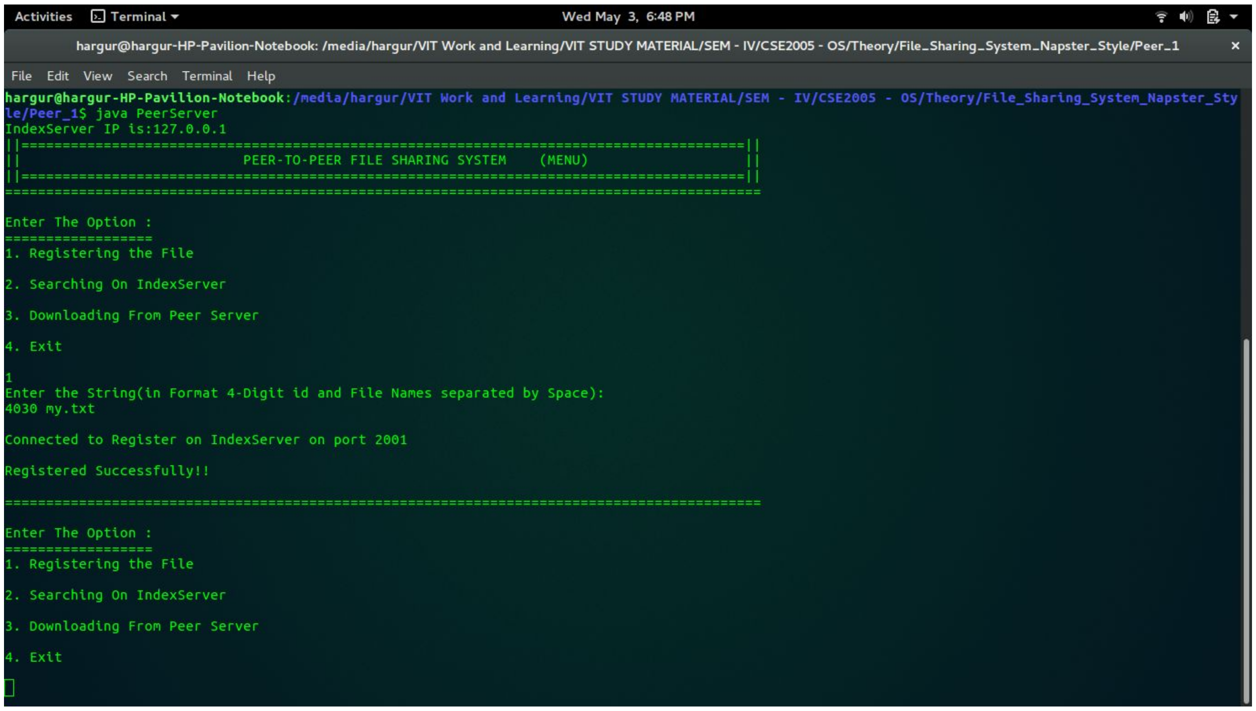

```
hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/Peer_1
le/Peer_1$ ant
Buildfile: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/Peer_1/build.xml
     [echo] ========================================================================
     [echo] {    Hargur Partap Singh Bedi (15BCE1257) : P2P File Sharing System:   }
     [echo] ========================================================================
     [echo] {           [==========:PeerServer_1 - BUILD SCRIPT:==========]              }
     [echo] ========================================================================

clean:

compile:
    [javac] Compiling 1 source file to /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster
_Style/Peer_1
     [echo]
     [echo]  ========================================================================
     [echo]  {                                          End Of Build Script !!!                                   }
     [echo]  ========================================================================

BUILD SUCCESSFUL
Total time: 0 seconds
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/Peer_1$ java PeerServer
IndexServer IP is:127.0.0.1
||=======================================================================================||
||                             PEER-TO-PEER FILE SHARING SYSTEM    (MENU)                ||
||=======================================================================================||
===========================================================================================

Enter The Option :
==================
1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

```

<u>Registration Process:</u>

Peer_1:

Peer_1 is Registering the File “**my.txt**” with the **IndexServer**.

Screenshot when **Peer_1** Registers the file on the **IndexServer**:

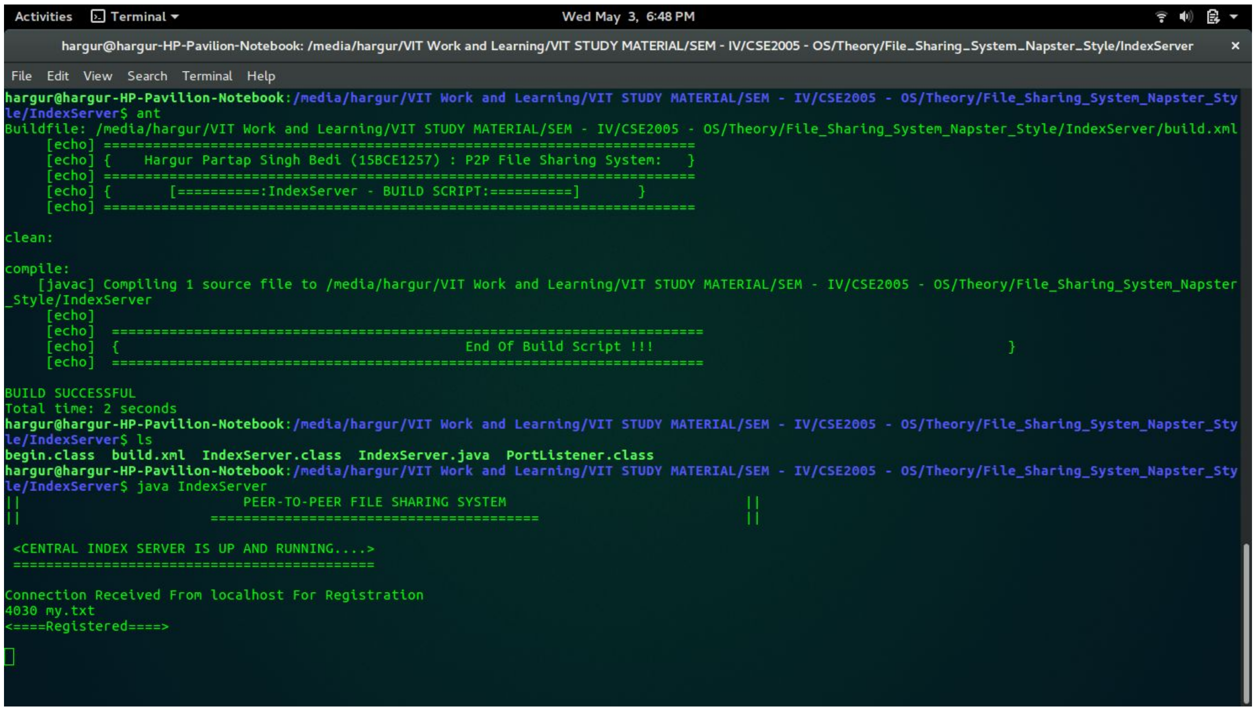

Activities
Terminal ▾
Wed May 3, 6:48 PM
hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/Peer_1
File Edit View Search Terminal Help

```
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/Peer_1$ java PeerServer
IndexServer IP is:127.0.0.1
||=========================================================================================||
||                             PEER-TO-PEER FILE SHARING SYSTEM    (MENU)                  ||
||=========================================================================================||
=============================================================================================

Enter The Option :
==================
1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

1
Enter the String(in Format 4-Digit id and File Names separated by Space):
4030 my.txt

Connected to Register on IndexServer on port 2001

Registered Successfully!!

=============================================================================================

Enter The Option :
==================
1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

```

IndexServer:

Register Request received on the **IndexServer** from **Peer_1** and the CIS registers the filename and the ID in the Index Catalogue.

Screenshot for Register Request from **Peer_1** on the **IndexServer**:


Activities
Terminal ▾
Wed May 3, 6:48 PM
hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/IndexServer
File Edit View Search Terminal Help

```
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/IndexServer$ ant
Buildfile: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/IndexServer/build.xml
     [echo] ========================================================================
     [echo] {    Hargur Partap Singh Bedi (15BCE1257) : P2P File Sharing System:    }
     [echo] ========================================================================
     [echo] {       [==========:IndexServer - BUILD SCRIPT:==========]       }
     [echo] ========================================================================

clean:

compile:
    [javac] Compiling 1 source file to /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster
_Style/IndexServer
     [echo]
     [echo]  ========================================================================
     [echo]  {                                          End Of Build Script !!!                                           }
     [echo]  ========================================================================

BUILD SUCCESSFUL
Total time: 2 seconds
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/IndexServer$ ls
begin.class  build.xml  IndexServer.class  IndexServer.java  PortListener.class
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/IndexServer$ java IndexServer
||                             PEER-TO-PEER FILE SHARING SYSTEM                              ||
||                         ========================================                          ||

 <CENTRAL INDEX SERVER IS UP AND RUNNING....>
 ============================================

Connection Received From localhost For Registration
4030 my.txt
<====Registered====>

```

Search Process:

Peer_2:

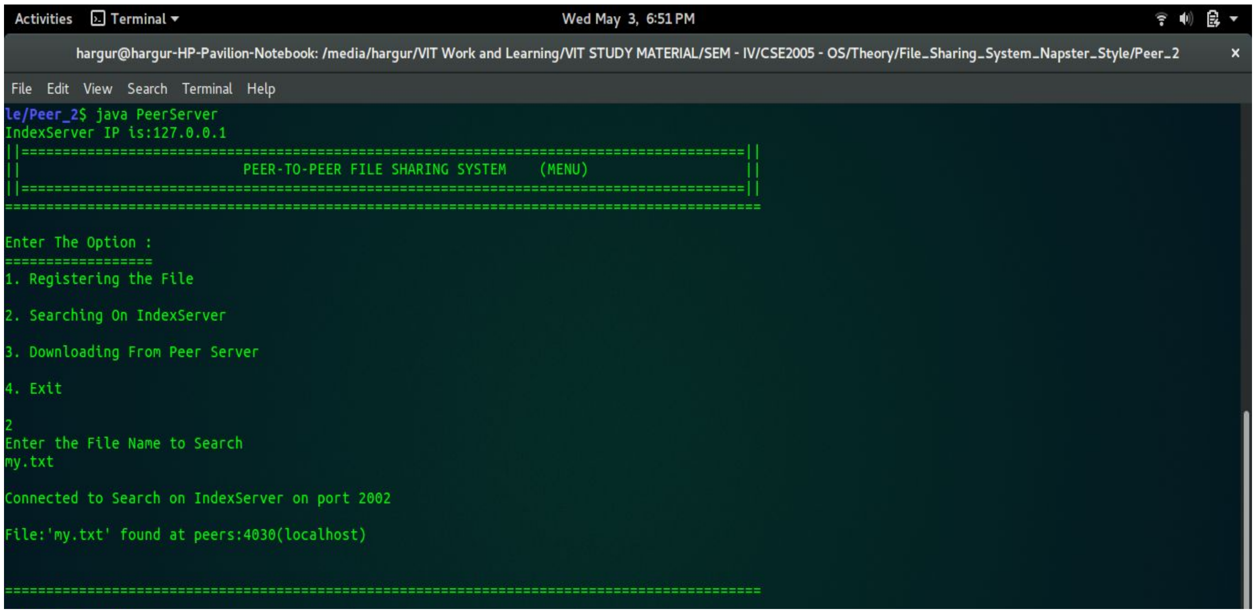
On Machine-C, **Peer_2** is Run.

Peer_2 is allowed to Register its files with the **IndexServer**, Search the CIS for required files or Download from other peers.

Peer_2 issues a Search Request for the file "**my.txt**" which was registered by **Peer_1**

The Search result returned will be the Peer ID of the peer and the IP Address of the Peer which contains the file.

Screenshot for the Search Result returned from the **IndexServer** for the file "**my.txt**":

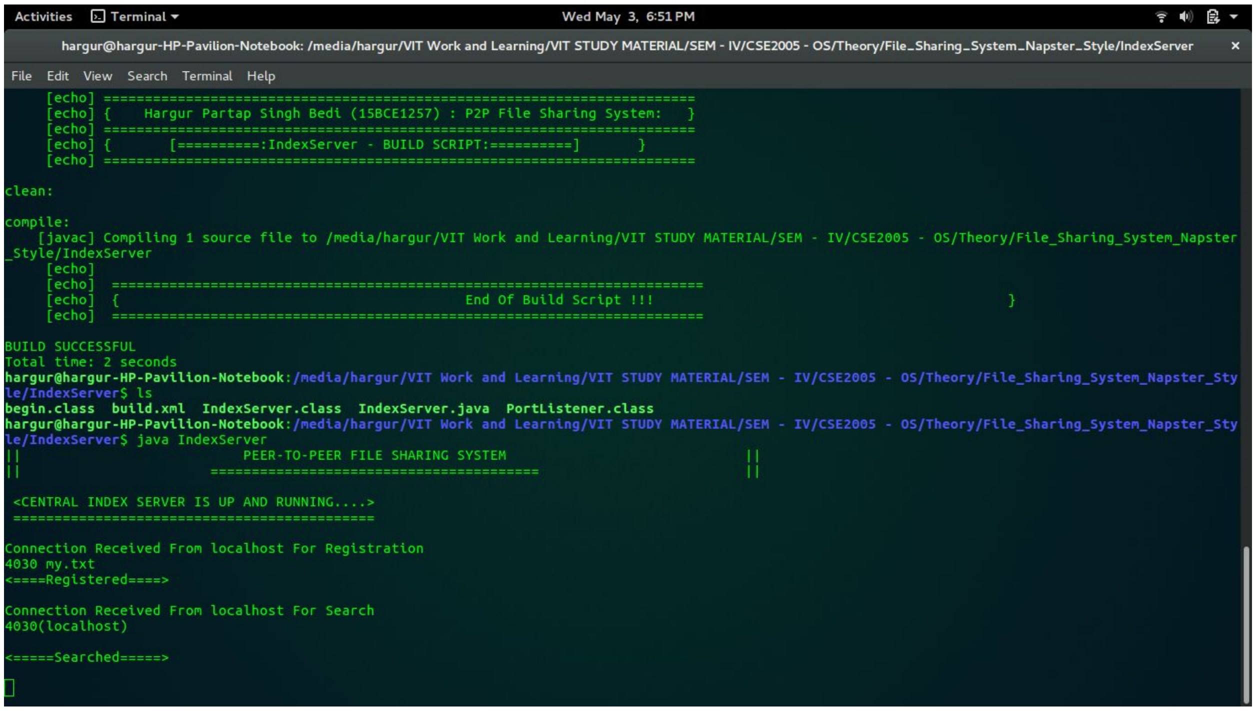

Activities
Terminal
Wed May 3, 6:51 PM
hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/Peer_2
File Edit View Search Terminal Help

```
le/Peer_2$ java PeerServer
IndexServer IP is:127.0.0.1
||==========================================================================================||
||                              PEER-TO-PEER FILE SHARING SYSTEM    (MENU)                  ||
||==========================================================================================||
==============================================================================================

Enter The Option :
==================
1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

2
Enter the File Name to Search
my.txt

Connected to Search on IndexServer on port 2002

File:'my.txt' found at peers:4030(localhost)


==============================================================================================
```

IndexServer:

Search Request received on the **IndexServer** from **Peer_2** and the CIS Searches the Filename in the Index Catalogue.

Screenshot for Search Request from **Peer_2** on the **IndexServer**:



Activities   Terminal ▾   Wed May 3, 6:51 PM

hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/IndexServer

File Edit View Search Terminal Help

```
     [echo] ========================================================================
     [echo] {    Hargur Partap Singh Bedi (15BCE1257) : P2P File Sharing System:   }
     [echo] ========================================================================
     [echo] {       [==========:IndexServer - BUILD SCRIPT:==========]       }
     [echo] ========================================================================

clean:

compile:
    [javac] Compiling 1 source file to /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster
_Style/IndexServer
     [echo]
     [echo]  ========================================================================
     [echo]  {                                          End Of Build Script !!!                                          }
     [echo]  ========================================================================

BUILD SUCCESSFUL
Total time: 2 seconds
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/IndexServer$ ls
begin.class  build.xml  IndexServer.class  IndexServer.java  PortListener.class
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/IndexServer$ java IndexServer
||                             PEER-TO-PEER FILE SHARING SYSTEM                             ||
||                         ========================================                         ||

 <CENTRAL INDEX SERVER IS UP AND RUNNING....>
 ============================================

Connection Received From localhost For Registration
4030 my.txt
<====Registered====>

Connection Received From localhost For Search
4030(localhost)

<=====Searched=====>
```

<u>Download Process:</u>

Peer_3:

On Machine-D, **Peer_3** is Run.

Peer_3 is allowed to Register its files with the **IndexServer**, Search the CIS for required files or Download from other peers.

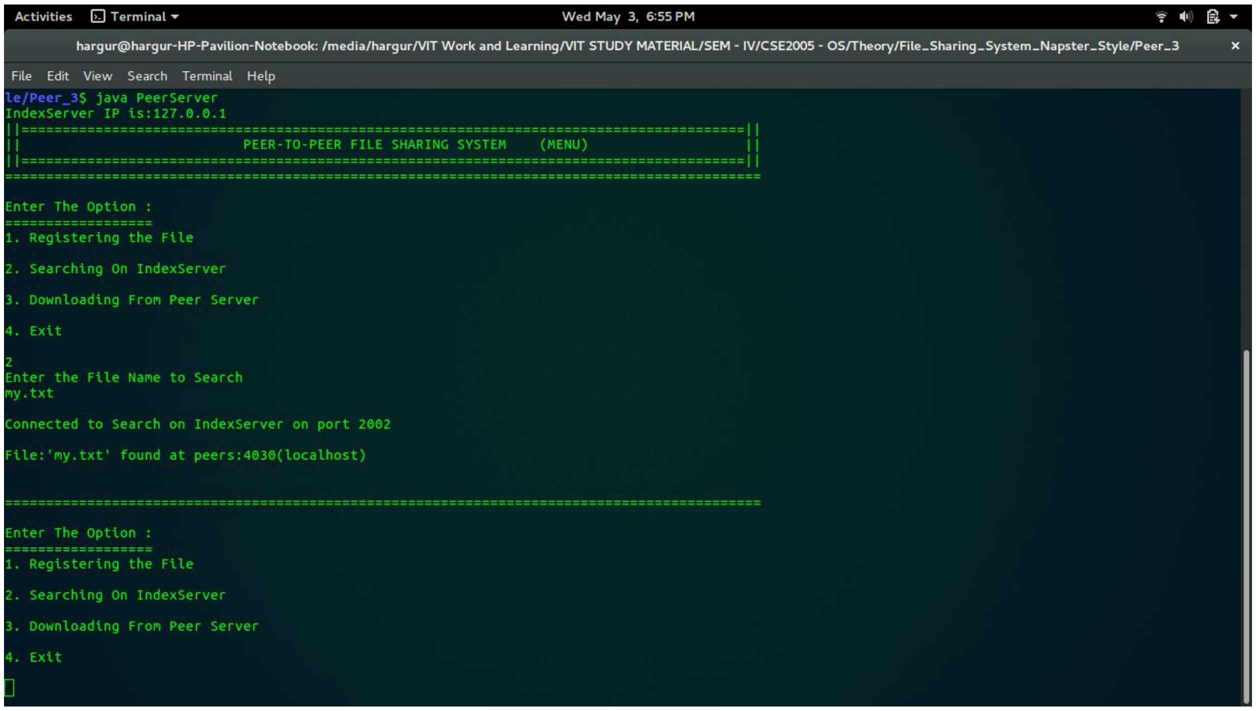
Peer_3 issues a Search Request for the file "**my.txt**" which was registered by **Peer_1**

The Search result returned will be the Peer ID of the peer and the IP Address of the Peer which contains the file.

Then It issues a Download Request for the file "**my.txt**" which was registered by **Peer_1**

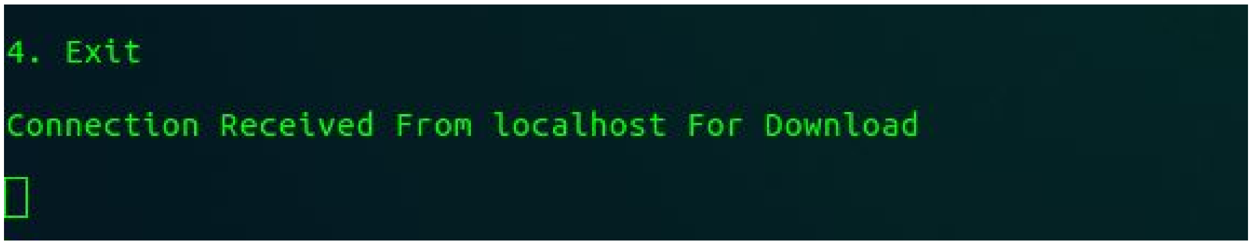
Screenshot for the Search Result returned from the **IndexServer** for the file

"**my.txt**"


Activities
Terminal
Wed May 3, 6:55 PM
hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/Peer_3
File Edit View Search Terminal Help

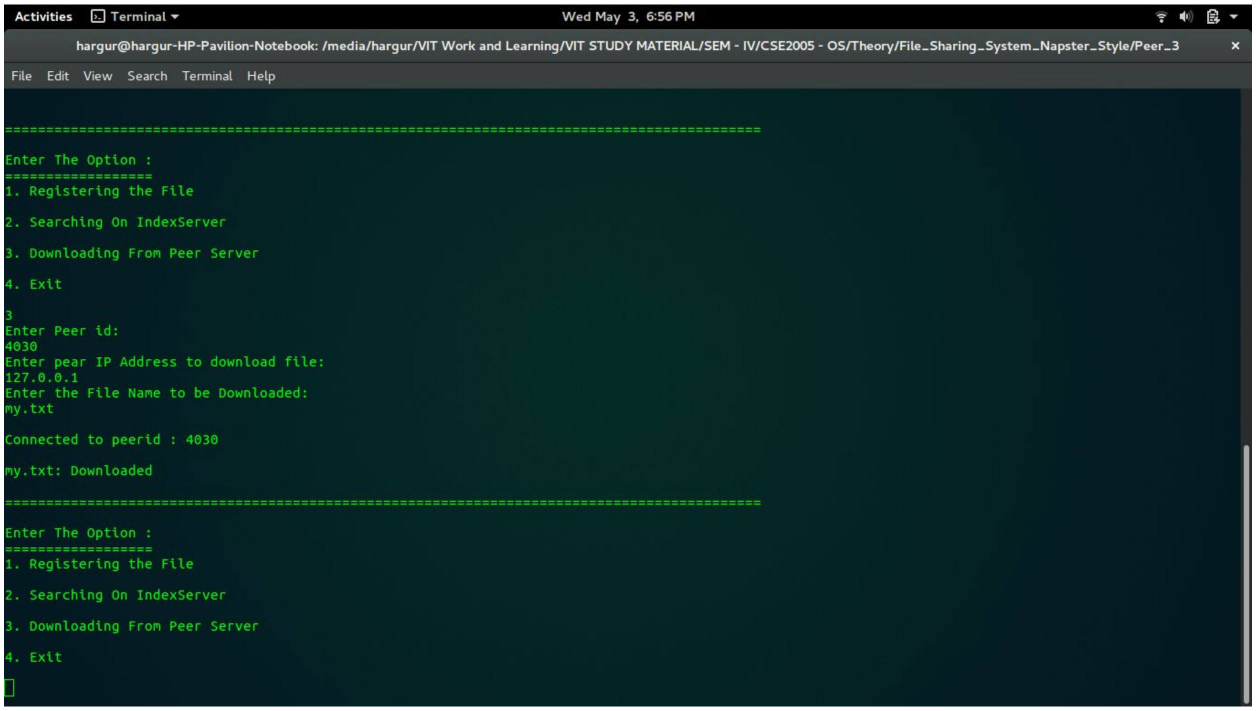
```
le/Peer_3$ java PeerServer
IndexServer IP is:127.0.0.1
||=========================================================================================||
||                             PEER-TO-PEER FILE SHARING SYSTEM    (MENU)                  ||
||=========================================================================================||
=============================================================================================

Enter The Option :
==================
1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

2
Enter the File Name to Search
my.txt

Connected to Search on IndexServer on port 2002

File:'my.txt' found at peers:4030(localhost)


=============================================================================================

Enter The Option :
==================
1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

```

Screenshot: Download Request from **Peer_3** to **Peer_1** for downloading the File “**my.txt**”



```
4. Exit

Connection Received From localhost For Download

```

Screenshot: **Peer_3** Downloads the File “**my.txt**” from **Peer_1**


Activities
Terminal
Wed May 3, 6:56 PM
hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/Peer_3
File Edit View Search Terminal Help

```
==============================================================================================

Enter The Option :
==================
1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

3
Enter Peer id:
4030
Enter pear IP Address to download file:
127.0.0.1
Enter the File Name to be Downloaded:
my.txt

Connected to peerid : 4030

my.txt: Downloaded

==============================================================================================

Enter The Option :
==================
1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

```

Design Document

The P2P File sharing system is designed keeping in mind the P2P architecture and its

underlying protocols.

The entire design is implemented using Java and some of the abstractions used are: Sockets

and Threads.

The project is built on Linux Operating system

The P2P file sharing system has two components:

1. **Central Index Server:**

   This server indexes the contents of all the peers that register with it. It also provides a search facility to the peers.

   The Central Index Server provides the following Interfaces to the Peers:

   a. **Registry (peer id, filename)** – invoked by a peer to register its files with the Central Index server. The CIS then builds an Index for the peers.

   b. **Search (filename)** – this procedure searches the index and then returns all the matching peers to the requestor.

2. **Peer:**

   The peer acts as both a client and a server. As a client the user specifies the filename with the indexing server using "lookup". The Indexing server then returns a list of all other peers that hold the file. The user can then pick one such peer and the client then connects to this peer and downloads the file. As a Server, the peer waits for requests from other peers and sends the requested file when receiving a request.The Peer Server provides the following interface to the Peer client:

   a. **Obtain/Download (Peer ID, filename)**- invoked by a peer to download file from another peer.

Implementation

The Implementation is done using Socket Libraries for communication between the Central Index Server and the Peers as well as between the Peer Servers. The CIS and the Peer-Servers are Multi-Threaded.

The various Functional Modules are :

a. **Register**
b. **Search**
c. **Download**

<u>The Central Index Server [CIS]:</u>

When the CIS is run, Initially the Index Server will be in the Idle State.Peer-To-Peer File Sharing System

The CIS Socket then keeps listening on a Port for incoming connection requests. Whenever a Peer makes a request on the CIS port, the CIS serves the request for the connected peer. The Registry and Search method is implemented on the Central Index Server.

1. The **<u>Register</u>** Method is implemented as follows:

When the peer invokes the register method to Register its files with the Index server, the Server has a RegisterRequestThread(); running and listening on a Port to serve the Register request.

The Peer enters the Peer ID followed by the Filename which it wants to Register.

The Peer then invokes the RegisterWithIServer() method. As soon as this is done, a Socket connection is automatically established with the Central Index Server with the CIS_ip and the port number.

The Index Server maintains an ArrayList to hold all the values of the Peer ID's and the Filenames with the corresponding Peers. This ArrayList serves as the Index for all other peers for their requests. The Peer ID and the Filename that the Peer wants to register is then stored in this ArrayList as an Index catalogue. This completes the Registration procedure.

Registration Pseudocode:

Index Server–Runs a RegisterRequestThread() and listens for active registration requests.

- Request for Register from the Peer
- Peer enters the 4-Digit Peer ID and the Filename it wants to register

  Example: 3001 testfile.txt
- RegisterWithIServer()
- Request for Socket with (CIS_ip, Port No)
- Peer sends Peer ID and filename over the socket through the Output stream.

- Server stores the Peer ID and the Filename received over the Input Stream into the Index ArrayList.
- Index Server serves Register request for the Peer.

2. The **<u>Search</u>** Method is implemented as follows:

   When the peer invokes the Search method to search for files with other peers on the Index Server, the Server has a SearchRequestThread(); running and listening on a Port to serve the Search request.

   The Peer enters the Filename to Search.

   The Peer then invokes the SearchWithIServer() method. As soon as this is done, a Socket

   connection is automatically established with the Central Index Server with the CIS_ip and the port number.

   The Index Server maintains an ArrayList to hold all the values of the Peer ID's and the

   Filenames with the corresponding Peers.

   The Index Server then receives the Filename to search from the Input stream and then it

   traverses the ArrayList to find a match for the Filename searched. If the filename is present. In the Index, the Server then returns the Search result with the Filename and the Peer ID with which the File is present. This completes the Search procedure.

   Search Pseudocode:

   Index Server – Runs a SearchRequestThread() and listens for active search requests.

- Request for search from the Peer
- Peer enters the Filename it wants to search

  Example: testfile.txt
- SearchWithIServer()
- Request for Socket with (CIS_ip, Port No)
- Peer sends the filename over the socket through the Output stream.
- Server receives the Filename over the Input Stream and then traverses the Index ArrayList.
- If a Match is found, it returns the Peer ID and the IP Address of the Peer which contains the searched File.
- If no match is found, it returns "File Not Found"

- Index Server serves Search request for the Peer.

3. The **Download** Method is implemented as follows:

   When the peer invokes the Download method to download the files with other peers, the Peer-Server which has the file invokes AttendFileDownloadRequest(); thread running and listening on a Port to serve the Download request.

   The Peer-Client enters the Peer ID and the Filename to Download.

   The Peer then invokes the DownloadFromPeerServer() method. As soon as this is done, a Socket connection is established with the Peer-Server which now acts as the Server to serve the Download request.

   The Peer-Server then receives the Filename to be downloaded from the Input stream and then it then opens a Filereader object to check for the file and read the contents and then uses a writeObject to store the contents from the output stream and finally on getting the writeobject with the Filename, it then writes to the file from the Input stream. This completes the Download procedure.

   Download Pseudocode:

   Peer-Server – Runs a AttendFileDownloadRequest() and listens for active download requests.

- Request for download from the Peer
- Peer enters the Peer ID, IP Address of the Peer holding the file and the Filename it wants to download.

  Example: Peer ID: 3001

  Peer IP Address: 10.0.0.13

  Filename: testfile.txt
- DownloadFromPeerServer ()
- Request for Socket with (Peer_Ip_Addr, Peer ID1)
- Peer sends the filename over the socket through the Output stream.
- Peer-Server receives the Filename over the Input Stream and then readObject to read the contents of the file and sends it over the Output Stream.
- The WriteToFile Object then writes the contents to the file, on the requestor Peer
- Peer-Server serves Download request for the Peer.

<u>Pros and Cons of the P2P Model</u>

Pros

- Less Network Resource consumption.
- High Scalability

Cons

- Central Index Server dependent.
- Less fault tolerant.

<u>**Tradeoffs**</u>

- Used ArrayList instead of HashMap.
  The ArrayList has O(n) performance for every search, so for n searches its performance is O(n^2).
  The HashMap has O(1) performance for every search (on average), so for n searches its performance will be O(n). For random operations of content retrieval, a HashMap is better. For retrieving items in an Order, ArrayList is better. So, there is a tradeoff in the speed in this model due to the use of ArrayList instead of HashMap.
- Files of large size say few 100MB's cannot be transmitted/downloaded.
- Automatic Registration of all the files in the folder at one shot is not implemented.

Instead, Multiple files can be registered simultaneously by the Peer, by specifying the Peer ID

and the Filenames, it wants to register.

Example: Enter the 4-Digit ID and the Filename(s) to Register:

3001 A.txt B.txt C.txt D.txt E.txt .... And so on.....

Program List/ Directory Structure

1. The **IndexServer** folder contains:
   a. The Source file of the **IndexServer** – "***IndexServer.java***"
   b. Compiled files- ["IndexServer.class", "PortListener.class" and "begin.class"]
   c. "***build.xml***" which is an "ant"-BuildScript to compile and build the **IndexServer**.
2. The **Peer_1** folder contains:
   a. The Source file of **Peer_1** – "***PeerServer.java***"
   b. Compiled files – ["PeerServer.class" and "PortListenerSend.class"]

c. “***indxip.txt***” – Which contains the IP Address of the IndexServer [This text file should be edited – And the IP Address of the Machine which runs the IndexServer should be entered in this file before running the Peer].

d. Few Testfiles : “my.txt”, testfile.txt, 1.txt, 2.txt and so on upto 10.txt [12 Testfiles of Peer_1 – which can be used for registration, search and download functionalities].

e. “***build.xml***” which is an “ant”-BuildScript to compile and build the **Peer_1**.

3. The **Peer_2** folder contains:

a. The Source file of **Peer_2** – “***PeerServer.java***”

b. Compiled files – [“PeerServer.class” and “PortListenerSend.class”]

c. “***indxip.txt***” – Which contains the IP Address of the IndexServer [This text file should be edited – And the IP Address of the Machine which runs the IndexServer should be entered in this file before running the Peer].

d. Few Testfiles : 11.txt, 12.txt and so on upto 20.txt [10 Testfiles of **Peer_2** – which can be used for registration, search and download functionalities].

e. “***build.xml***” which is an “ant”-BuildScript to compile and build the **Peer_2**.

4. The **Peer_3** folder contains:

a. The Source file of **Peer_3** – “***PeerServer.java***”

b. Compiled files – [“PeerServer.class” and “PortListenerSend.class”]

c. “***indxip.txt***” – Which contains the IP Address of the IndexServer [This text file should be edited – And the IP Address of the Machine which runs the IndexServer should be entered in this file before running the Peer].

d. Few Testfiles : 21.txt, 22.txt and so on upto 30.txt [10 Testfiles of **Peer_3** – which can be used for registration, search and download functionalities].

e. “***build.xml***” which is an “ant”-BuildScript to compile and build the **Peer_3**.

User Manual

Steps to Run the Project:

1. Go to this Link and download the whole working project.
2. The System should be running on Linux Environment.
3. Java Development Kit(JDK) and JRE(Java Runtime Environment) should be installed on the System.
   If not installed, open Terminal and type **sudo apt-get install openjdk-8-jdk-headless** and enter your password for installation.
4. Ant Tool should be installed to run the Build Script.
   If not installed, open Terminal and type **sudo apt-get install ant** and enter your password for installation.
5. Launch the Terminal application and Extract the given zip file: "File_Sharing_System_Napster_Style.zip" Using the command: **unzip File_Sharing_System_Napster_Style.zip**
6. Navigate into the Extracted folder > There will be subfolders which contains the Central Index Server(CIS) IndexServer and the Peer(s) Modules: Peer_1, Peer_2, Peer_3
7. There are 2 ways to simulate the entire project:
   a. Running the entire project on a Single system by running Separate processes on "localhost" or by running Multiple Virtual Machines.
   b. Running the project on Multiple Machines.

Steps for running the entire project on a Single System by running separate processes on "localhost" or by Multiple Virtual Machines.

1. Launch the Terminal and Navigate into the File_Sharing_System_Napster_Style folder from the above zip.
2. Open Multiple-tabs and navigate into IndexServer folder on the First tab, into the Peer_1 folder on the second tab, into the Peer_2 folder on the third tab and into the Peer_3 folder on the fourth tab.
3. Navigate into the First tab, where IndexServer files are present.

4. On the Command prompt Run the build-script using the ***ant*** command.

   $ ant

5. The Build-script is executed and it cleans and compiles the files of the IndexServer present in the IndexServer directory. After successful build, a message is displayed on the terminal:

   Build Successful

   Total Time: 1 second

6. Run the IndexServer by the command:

   $ java IndexServer

   A message is displayed saying the "<Central Index Server is Up and Running>"

   This runs the IndexServer and it starts listening to other Peers on the Port

7. Now, Navigate into the Second tab, where **Peer_1** files are present.

8. Edit the file "***indxip.txt***" present inside **Peer_1** folder using "***vim***" or "***gedit***". Edit and Enter the IP Address of the Machine which is running the IndexServer into this file. The IP Address of the Machine running IndexServer can be obtained by running the "***ifconfig***" command on the IndexServer Machine.

   Example: If the IP Address of the IndexServer Machine is: 10.0.0.16 or 127.0.0.1(for Localhost)

   $vim indxip.txt

   Inside the file-> Remove the existing IP Address and Enter:

   10.0.0.16

   Save the file and exit.

9. On the Command prompt Run the build-script using the ant command.

   $ ant

10. The Build-script is executed and it cleans and compiles the files of the Peer_1 present in the Peer_1 directory. After successful build, a message is displayed on the terminal:

    Build Successful

    Total Time: 1 second

11. Run Peer_1 using the command:

    $ java PeerServer

12. The Peer is run and a MENU is displayed on the screen with the following options for the Peer to select:

Enter The Option:

==================

1. Registering the File

2. Searching On IndexServer

3. Downloading From Peer Server

4. Exit

Registration Process:

13. If the Peer_1 wants to Register the file, it can choose option 1.
14. The Peer_1 then has to:

    Enter the String(in Format 4-Digit id and File Names separated by Space):

    Example: 3001 my.txt

15. Then the Peer_1 is Connected to Register on IndexServer on port 2001
16. And the file "my.txt" is "Registered Successfully" on the IndexServer.
17. The Peer_1 can also register multiple files by specifying the ID and filenames separated by a space

    Example: 4001 A.txt B.txt C.txt D.txt……..and so on.

Search Process:

18. If Peer_2 wants to Search the file "my.txt", it can Search on the IndexServer to find out which Peer has the required file.
19. Navigate into the Third tab, where Peer_2 files are present.
20. Edit the file "***indxip.txt***" present inside Peer_2 folder using "***vim***" or "***gedit***". Edit and Enter the IP Address of the Machine which is running the IndexServer into this file. The IP Address of the Machine running IndexServer can be obtained by running the "***ifconfig***" command on the IndexServer Machine.
    Example: If the IP Address of the IndexServer Machine is: 10.0.0.16

$vim indxip.txt

Inside the file-> Remove the existing IP Address and Enter:

10.0.0.16

21. Save the file and exit.
22. On the Command prompt Run the build-script using the ant command.

    $ ant
23. The Build-script is executed and it cleans and compiles the files of the Peer_2 present in the Peer_2 directory. After successful build, a message is displayed on the terminal:

    Build Successful

    Total Time: 1 second
24. Run Peer_2 using the command:

    $ java PeerServer
25. Then once it is run, select 2nd which is the Search option from the MENU to search for the desired file.

Then Peer_2 has to:

Enter the filename to search:

Example: ***my.txt***

Then the Peer_2 is Connected to Search on IndexServer on port 2002

And the Search result is returned from the IndexServer:

File:'my.txt' found at peers:3001 (10.0.0.13).

Here 3001 is the Peer ID and "10.0.0.13" is the IP address of the Peer holding the File "***my.txt***". In this case it is Peer_1 which has the file "***my.txt***".

Download Process:

26. If Peer_3 wants to download the file "my.txt" which is registered by Peer_1, then

    It can Download from Peer_1 which acts as the PeerServer.

    Navigate into the Fourth tab, where the Peer_3 Files are present.
27. Edit the file "***indxip.txt***" present inside Peer_3 folder using "***vim***" or "***gedit***". Edit and Enter the IP Address of the Machine which is running the IndexServer into this file. The IP Address of the Machine running IndexServer can be obtained by running the "***ifconfig***" command on the IndexServer Machine.

Example: If the IP Address of the IndexServer Machine is: 10.0.0.16

$vim indxip.txt

Inside the file-> Remove the existing IP Address and Enter:

10.0.0.16

Save the file and exit.

28. On the Command prompt Run the build-script using the ant command.

    $ ant

29. The Build-script is executed and it cleans and compiles the files of the Peer_3 present in the Peer_3 directory. After successful build, a message is displayed on the terminal:

    Build Successful

    Total Time: 1 second

30. Run Peer_3 using the command:

    $ java PeerServer

Then once it is run, select 3rd option from the MENU which is for Downloading from the Peer_1 which acts as the PeerServer.

The Peer_3 has to then:

Enter the Peer ID: (The Peer ID of the Peer which contains the file "my.txt" has to be entered-This can be obtained by the search result)

Example: ***3001***

Enter the Peer IP Address to Download the file:

Example: ***10.0.0.13***

Enter the Filename to be downloaded:

Example: ***my.txt***

The Peer_1 Receives a connection request for Download from Peer_3. Then the Peer_3 is Connected to PeerServer with the peerid: 3001 and IP Address (10.0.0.13)

And the requested file is downloaded.

my.txt: Downloaded.

Steps for running the project on different machines.

The only change in this process than the above given process is that there are different machines, one working as the Central Indexing Server and three other working as individual Peer-Server.

So, for this run IndexServer one Machine A and check the IP Address of CIS using command:
$ ifconfig

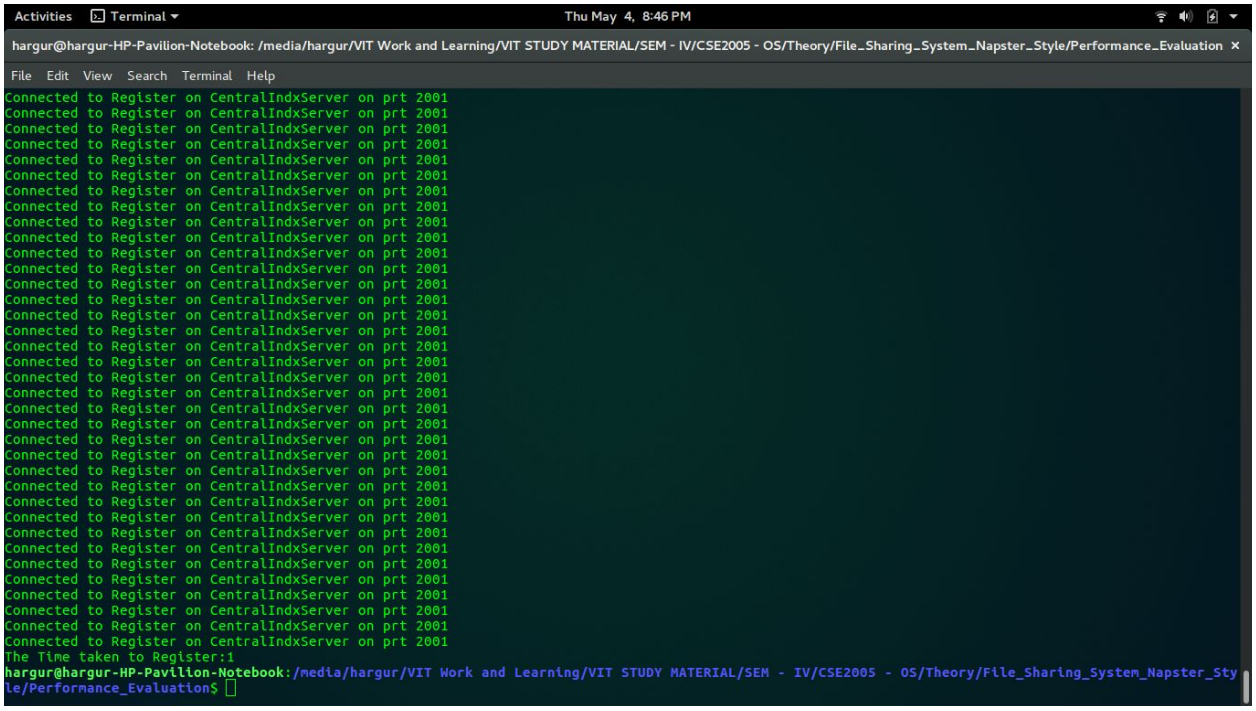
So now in other machines i.e. Machine B, Machine C and Machine D, change the IP Address in ***indxip.txt*** using any text editor.

Run Peer_1, Peer_2 and Peer_3 respectively.

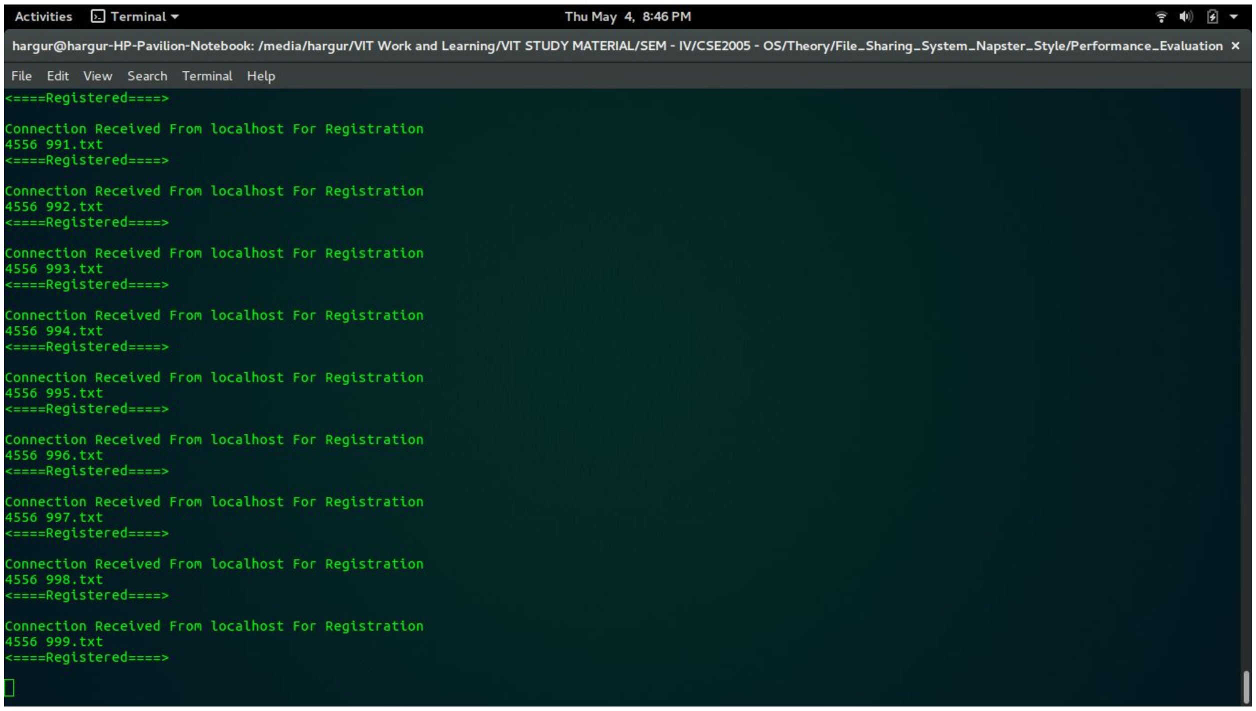
Rest all Procedures is same.

Performance Evaluation

Run the **IndexServer.java** on one machine and run the **CreateFiles.java** to create 1000 files of 42 Bytes each. Now run **PerfTest.java** on other terminal, to register these 1000 files it took **1.34200 secs.**


Activities
Terminal
Thu May 4, 8:46 PM
hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/Performance_Evaluation
File
Edit
View
Search
Terminal
Help

```
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
Connected to Register on CentralIndxServer on prt 2001
The Time taken to Register:1
hargur@hargur-HP-Pavilion-Notebook:/media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Sty
le/Performance_Evaluation$ 
```


Activities
Terminal
Thu May 4, 8:46 PM
hargur@hargur-HP-Pavilion-Notebook: /media/hargur/VIT Work and Learning/VIT STUDY MATERIAL/SEM - IV/CSE2005 - OS/Theory/File_Sharing_System_Napster_Style/Performance_Evaluation
File
Edit
View
Search
Terminal
Help

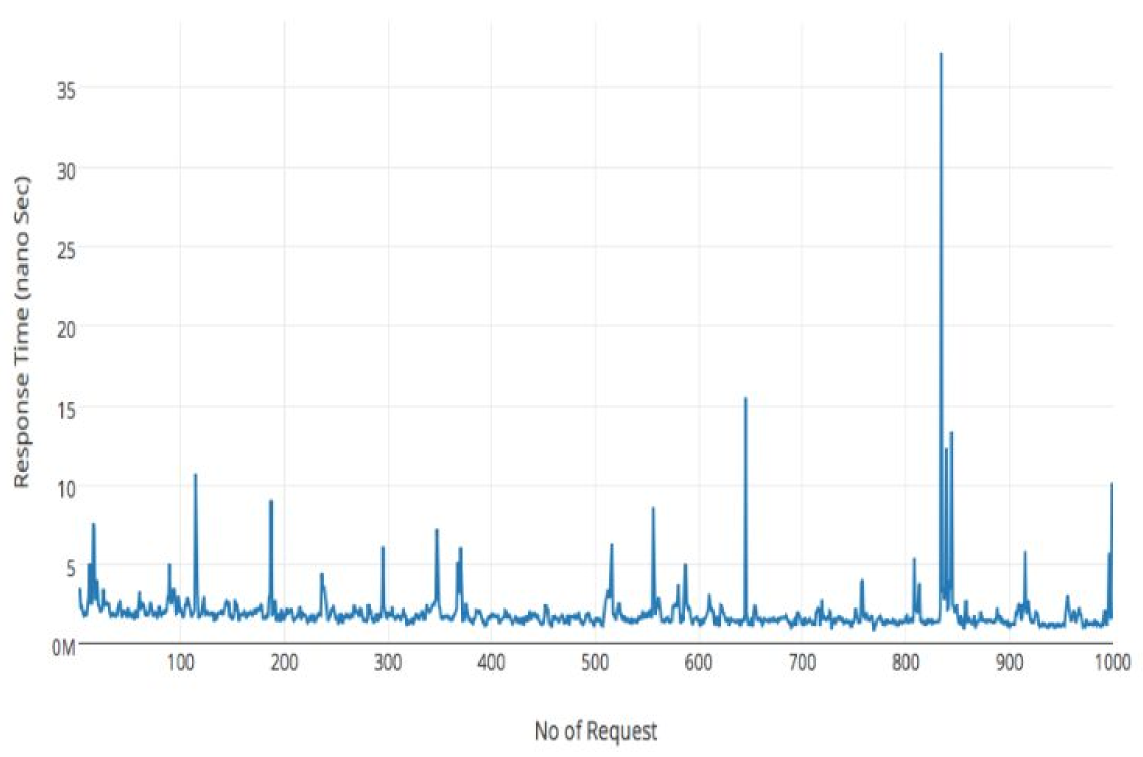
```
<====Registered====>

Connection Received From localhost For Registration
4556 991.txt
<====Registered====>

Connection Received From localhost For Registration
4556 992.txt
<====Registered====>

Connection Received From localhost For Registration
4556 993.txt
<====Registered====>

Connection Received From localhost For Registration
4556 994.txt
<====Registered====>

Connection Received From localhost For Registration
4556 995.txt
<====Registered====>

Connection Received From localhost For Registration
4556 996.txt
<====Registered====>

Connection Received From localhost For Registration
4556 997.txt
<====Registered====>

Connection Received From localhost For Registration
4556 998.txt
<====Registered====>

Connection Received From localhost For Registration
4556 999.txt
<====Registered====>
```

Both clients and Peer are running on same machine

Test 1:

We are making 1000 sequential request from one peer to server for searching functionality of file names present in central index server. Below graph Represent the response time for each request taken by server. Unit of Time is in nanosec for better accuracy of result.

Response Time Graph



Source Code

Once in File_Sharing_System_Napster_Style folder, navigate to IndexServer subfolder to find IndexServer.java and build.xml

IndexServer Subfolder:

Code : IndexServer.java

```
//IndexServer Implementation
import java.io.*;

//PeerServer
import java.io.BufferedReader;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

class begin
{
	String filename;		//Declaration filename,peerid and 
ipAddress
	int peerid;
	String ipAddress;
}

class PortListener implements Runnable {

	ServerSocket server;
	Socket connection;
	BufferedReader br = null;
	Boolean flag;
	public String strVal;
	int port;
	static int maxsize = 0;
	static begin[] myIndexArray = new begin[9000];
//ArrayList Initialisation

	public PortListener(int port) {
		this.port = port;
		flag = true;//Initial Idle state
		strVal = "Waiting For PEER Connection";
	}
```

```
     /* Beginning of Run Method */
     public void run() {
          if(port==2001)
//Listening For Register on port 2001
          {
               try {
                    server = new ServerSocket(2001);
                    while (true) {
                         connection = server.accept();
                         System.out.println("Connection Received
From " +connection.getInetAddress().getHostName()+ " For
Registration");
                         ObjectInputStream in = new
ObjectInputStream(connection.getInputStream());
                         strVal = (String)in.readObject();
                         System.out.println(strVal);

System.out.println("<====Registered====>\n");
                         //Split string "strVal" using Space as
Delimeter store as {peerid ,filename} format;
                         String[] var;
                         var = strVal.split(" ");
                         int aInt = Integer.parseInt(var[0]);
                         String ipstrtmp =
connection.getInetAddress().getHostName();
                         /* print substrings */
                         for(int x = 1; x < var.length ; x++){

                              //  myIndexArray[maxsize].peerid =
.;
                              begin myitem = new begin();
                              myitem.filename = var[x];
//Storing Peer ID and Filename in the ArrayList
                              myitem.peerid = aInt  ;
                              myitem.ipAddress = ipstrtmp;
                              myIndexArray[maxsize] = myitem;
                              maxsize++;
                         }

                         in.close();
                         connection.close();
                    }
               }

               catch(ClassNotFoundException noclass){
//To Handle Exceptions for Data Received in Unsupported/Unknown
Formats
```

```
                      System.err.println("Data Received in Unknown
Format");
                }
                catch(IOException ioException){
//To Handle Input-Output Exceptions
                      ioException.printStackTrace();
                } finally {
                }

           }
           if(port==2002)                                 //Listening
for Search on port 2002
           {
                try {
                      server = new ServerSocket(2002);

                      while (true) {
                           connection = server.accept();
                           System.out.println("Connection Received
From " +connection.getInetAddress().getHostName()+ " For Search");

                           ObjectInputStream in = new
ObjectInputStream(connection.getInputStream());
                           strVal = (String)in.readObject();
                           String retval = "";
                           //   Peer-id's separated by space are
returned for given file

                           for (int idx =0; idx < maxsize ;idx++)
//Traversing the ArrayList
                           {
                                 if
(myIndexArray[idx].filename.equals(strVal))
//To Compare the filename with the Registered filenames in the
ArrayList
                                 {
                                      retval = retval +
myIndexArray[idx].peerid + "("+myIndexArray[idx].ipAddress +")\n\r ";
//Returns the list of Peerid's which has the searched file
                                 }
                           }
                           if (retval == "")
                           {
                                 retval = "File Not Found\n";
                           }
                           System.out.println(retval);

System.out.println("<=====Searched=====>\n");
```

```
                          ObjectOutputStream out = new
ObjectOutputStream(connection.getOutputStream());
                          out.flush();
                          out.writeObject(retval);
//Write the List of peer id's to the output stream
                          out.flush();
                          in.close();
                          out.close();
                          connection.close();
                     }
                }

                catch(ClassNotFoundException noclass){
//To Handle Exceptions for Data Received in Unsupported/Unknown
Formats
                     System.err.println("Data Received in Unknown
Format");
                }
                catch(IOException ioException){
//To Handle Input-Output Exceptions
                     ioException.printStackTrace();
                } finally {
                }

           }
     }
}


/*IndexServer Class Begin*/
public class IndexServer {

     public IndexServer() {
           RegisterRequestThread();
//RegisterRequest and SearchRequest Threads
           SearchRequestThread();
     }

     public static void main(String[] args) {

           System.out.println("||
PEER-TO-PEER FILE SHARING SYSTEM                             ||");
           System.out.println("||
========================================
||");
           System.out.println("\n <CENTRAL INDEX SERVER IS UP AND
RUNNING....>");
```

```
            System.out.println("
============================================\n");


            IndexServer mainFrame = new IndexServer();

      }
      public void RegisterRequestThread()
      {
            Thread rthread = new Thread (new PortListener(2001));
//Register Request Thread
            rthread.setName("Listen For Register");
            rthread.start();
      }
      public void SearchRequestThread()
      {
            Thread sthread = new Thread (new PortListener(2002));
//Search Request Thread
            sthread.setName("Listen For Search");
            sthread.start();

      }
}
```

Code : build.xml

```
<?xml version="1.0"?>
<project name="P2P_File_Sharing_System" default="compile"
basedir=".">
<echo>==============================================================
=========</echo>
<echo>{    Hargur Partap Singh Bedi (15BCE1257) : P2P File Sharing
System:   }</echo>
<echo>==============================================================
=========</echo>
<echo>{       [==========:IndexServer - BUILD SCRIPT:==========]
}</echo>
<echo>==============================================================
=========</echo>
<!-- set global properties for this build -->
<property name="IndexServer" location="IndexServer"/>

<target name="init">
<!-- Create the time stamp -->
<tstamp/>
```

```
</target>

<target name="clean">
<delete>
     <fileset dir="${basedir}" includes="**/*.class"/>
</delete>
</target>

<target name="compile" depends="clean">
     <javac includeantruntime="true" srcdir="${basedir}"
destdir="${basedir}" includes="**/*.java"/>
<echo> </echo>
<echo>
=====================================================================
===</echo>
<echo> {                                 End Of Build Script !!!
                  }</echo>
<echo>
=====================================================================
===</echo>
</target>
</project>
```

Peers Subfolder (E.g. Peer_1, Peer_2 and Peer_3)

Code : PeerServer.java

```
import java.io.*;
import java.net.*;
import java.util.Date;
import java.util.Scanner;

//PeerServer
class PortListenerSend implements Runnable {

     int port;
     public String strVal;
     Boolean flag;                              //declarations
     ServerSocket server;
     Socket connection;
     BufferedReader br = null;

     public PortListenerSend(int port) {
          this.port = port;
          flag = true;//Initial Idle state
          strVal = "Waiting For PEER Connection";
```

```
      }
      /* Beginning of Run Method */
      public void run() {
            try {
                  server = new ServerSocket(port);

                  while (true) {
//Listen for Download request
                        connection = server.accept();
                        System.out.println("Connection Received From "
+ connection.getInetAddress().getHostName()+" For Download\n");

                        ObjectInputStream in = new
ObjectInputStream(connection.getInputStream());
                        strVal = (String)in.readObject();

                        ObjectOutputStream out = new
ObjectOutputStream(connection.getOutputStream());
                        out.flush();

                        String str="";

                        try
                        {
                              FileReader fr = new FileReader(strVal);
//Reads the filename into Filereader
                              BufferedReader br = new
BufferedReader(fr);
                              String value=new String();
                              while((value=br.readLine())!=null)
//Appending the content read from the BufferedReader object until it
is null and stores it in str
                                    str=str+value+"\r\n";
                              br.close();
                              fr.close();
                        } catch(Exception e){
                              System.out.println("Cannot Open File");
                        }

                        out.writeObject(str);
                        out.flush();
                        in.close();
                        connection.close();
                  }
            }
```

```
          catch(ClassNotFoundException noclass){
//To Handle Exception for Data Received in Unsupported or Unknown
Formats
               System.err.println("Data Received in Unknown
Format");
          }
          catch(IOException ioException){
//To Handle Input-Output Exceptions
               ioException.printStackTrace();
          } finally {
          }
     }
}

/* PeerServer Class Begin */
public class PeerServer {

     //public String CIS_ip = "10.0.0.13";
//============>IP-address of the IndexServer has to be specified here
     public String CIS_ip = "localhost";
//============>IP-address of the IndexServer has to be specified here
     public String Clientid = "1001";
     String regmessage,searchfilename;
     ObjectOutputStream out;
     Socket requestSocket;

     public PeerServer() {


          try
          {
               FileReader fr = new FileReader("indxip.txt");//read
the filename in to filereader object
               String val1=new String();
               BufferedReader br = new BufferedReader(fr);
               val1 = br.readLine();
               System.out.println("IndexServer IP is:" + val1);
               CIS_ip = val1;
               br.close();
               fr.close();
          } catch(Exception e){
               System.out.println("Could not read indexserver ip
from indxip.txt");
          }
```

```
System.out.println("||================================================
==========================================||");
          System.out.println("||
PEER-TO-PEER FILE SHARING SYSTEM    (MENU)                  ||");

System.out.println("||================================================
==========================================||");

          while (true){

               //  System.out.println("\n");

System.out.println("================================================
============================================\n");
               System.out.println("Enter The Option
:\n==================\n1. Registering the File \n \n2. Searching On
IndexServer \n \n3. Downloading From Peer Server \n \n4. Exit\n");
               Scanner in = new Scanner(System.in);
               regmessage = in.nextLine();
               if (regmessage.equals("1")){

                    System.out.println("Enter the String(in Format
4-Digit id and File Names separated by Space): ");
                    //   Scanner in = new Scanner(System.in);
                    regmessage = in.nextLine();
                    //To Collect peer/client id from input string
                    String[] val;
                    val = regmessage.split(" ");
//Split the peerid and filename separated with a space
                    int PearPort  = Integer.parseInt(val[0]);

                    RegisterWithIServer();
//Register Method call
                    AttendFileDownloadRequest(PearPort);

               }
               if (regmessage.equals("2")){
                    SearchWithIServer();
//Search Method call
               }
               if (regmessage.equals("3")){
                    DownloadFromPeerServer();
//Download Method call
               }
               if (regmessage.equals("4")){
                    System.out.println("Exiting.");
                    System.exit(0);
```

```
                }
            }
        }

        /* Main Method Begin */
        public static void main(String[] args) {

            PeerServer psFrame = new PeerServer();

        }
        public void RegisterWithIServer()
//Register with IndexServer Method
        {
            try{
                //1. Creating a socket to connect to the server
                requestSocket = new Socket(CIS_ip, 2001);
                System.out.println("\nConnected to Register on
IndexServer on port 2001\n");
                //2. To Get Input and Output streams
                out = new
ObjectOutputStream(requestSocket.getOutputStream());
                out.flush();
                out.writeObject(regmessage);
                out.flush();
                System.out.println("Registered Successfully!!\n");
            }
            catch(UnknownHostException unknownHost){
//To Handle Unknown Host Exception
                System.err.println("Cannot Connect to an Unknown
Host!");
            }
            catch(IOException ioException){
//To Handle Input-Output Exception
                ioException.printStackTrace();
            }
            finally{
                //4: Closing connection
                try{
                    out.close();
                    requestSocket.close();
                }
                catch(IOException ioException){
                    ioException.printStackTrace();
                }
            }

        }
```

```
     public void SearchWithIServer()
//Search on the CentralIndexServer Method
     {
          try{
               System.out.println("Enter the File Name to Search");
               Scanner in1 = new Scanner(System.in);
//Takes Input from the Peer to search the desired file
               searchfilename = in1.nextLine();

               //1. Creating a socket to connect to the Index server
               requestSocket = new Socket(CIS_ip, 2002);
               System.out.println("\nConnected to Search on
IndexServer on port 2002\n");
               //2. To Get Input and Output streams
               out = new
ObjectOutputStream(requestSocket.getOutputStream());
               out.flush();
               out.writeObject(searchfilename);
//Writes the Search Filename to the Output Stream
               out.flush();
               ObjectInputStream in = new
ObjectInputStream(requestSocket.getInputStream());
               String strVal = (String)in.readObject();
               //  For File Not Found Print Condition
               if  (strVal.equals("File Not Found\n")) {

                    System.out.println("FILE Does Not Exist !!\n");
               }
               else {
                    System.out.println( "File:'"+searchfilename+ "'
found at peers:"+strVal+"\n");
               }

          }
          catch(UnknownHostException unknownHost){
//To Handle Unknown Host Exception
               System.err.println("Cannot Connect to an Unknown
Host!");
          }
          catch(IOException ioException){
//To Handle Input-Output Exception
               ioException.printStackTrace();
          } catch (ClassNotFoundException e) {
               e.printStackTrace();
          }
          finally{
               //4: Closing connection
               try{
                    out.close();
                    requestSocket.close();
                }
                catch(IOException ioException){
                    ioException.printStackTrace();
                }
            }
        }

        public void writetoFile(String s)
        {
            try
            {
                //To Append String s to Existing File
                String fname = searchfilename;
                FileWriter fw = new FileWriter(fname,true);
                fw.write(s);
//Write to file, the contents
                fw.close();

            } catch(Exception e){
                System.out.println("");
                //    System.out.println("Cannot Open File");     //
To Mask Print on Console
            }

        }

        public void DownloadFromPeerServer()
//Download Function Method
        {

            System.out.println("Enter Peer id:");
            Scanner in1 = new Scanner(System.in);
//Takes from the user the 4Digit Peer ID as input
            String peerid = in1.nextLine();

            System.out.println("Enter pear IP Address to download
file:");
            String ipadrs = in1.nextLine();
            System.out.println("Enter the File Name to be
Downloaded:");
            searchfilename = in1.nextLine();
//Takes from user the desired filename to be downloaded


            int peerid1 = Integer.parseInt(peerid);
            try{
                //1. Creating a socket to connect to the Index server
                requestSocket = new Socket(ipadrs, peerid1);
                System.out.println("\nConnected to peerid : 
"+peerid1+"\n");
                //2. To Get Input and Output streams
                out = new 
ObjectOutputStream(requestSocket.getOutputStream());
                out.flush();
                out.writeObject(searchfilename);
                out.flush();
                ObjectInputStream in = new 
ObjectInputStream(requestSocket.getInputStream());
                String strVal = (String)in.readObject();
                System.out.println( searchfilename+": Downloaded\n");
                writetoFile(strVal);
           }
           catch(UnknownHostException unknownHost){
//To Handle Unknown Host Exception
                System.err.println("You are trying to connect to an 
unknown host!");
           }
           catch(IOException ioException){
//To Handle Input-Output Exception

                System.err.println("FILE not Found at the Following 
PEER !!");
                System.err.println("Please enter a valid PEER ID!"); 
// To Avoid StackTrace Print on Console and Inform User
                DownloadFromPeerServer();                    // 
Calling Download Function Again to enable user to enter valid 
Filename and Port Number
                //   ioException.printStackTrace();
           } catch (ClassNotFoundException e) {
                e.printStackTrace();
           }
           finally{
                //4: Closing connection
                try{
                     //   in.close();
                     out.close();
                     requestSocket.close();
                }
                catch(IOException ioException){
                     ioException.printStackTrace();
                }
           }
      }
```

```
     public void AttendFileDownloadRequest(int peerid)
//FileDownload Request Thread
     {
          Thread dthread = new Thread (new
PortListenerSend(peerid));
          dthread.setName("AttendFileDownloadRequest");
          dthread.start();
     }
}
```

Code : build.xml

```
<?xml version="1.0"?>
<project name="P2P_File_Sharing_System" default="compile"
basedir=".">
<echo>===================================================================
=========</echo>
<echo>{    Hargur Partap Singh Bedi (15BCE1257) : P2P File Sharing
System:   }</echo>
<echo>===================================================================
=========</echo>
<echo>{         [==========:PeerServer_1 - BUILD SCRIPT:==========]
}</echo>
<echo>===================================================================
=========</echo>
<!-- set global properties for this build -->
<property name="Peer_1" location="Peer_1"/>

<target name="init">
<!-- Create the time stamp -->
<tstamp/>
</target>

<target name="clean">
<delete>
     <fileset dir="${basedir}" includes="**/*.class"/>
</delete>
</target>

<target name="compile" depends="clean">
     <javac includeantruntime="true" srcdir="${basedir}"
destdir="${basedir}" includes="**/*.java"/>
<echo> </echo>
<echo>
=====================================================================
===</echo>
```

```
<echo> {                             End Of Build Script !!!
                  }</echo>
<echo>
=====================================================================
===</echo>
</target>
</project>
```