Lab 1: Peer-to-Peer File System
CSE 514 – Computer Networking
Pennsylvania State University
Fall 2010

Submitted by : Mahadevan Srinivasan

System Description

The System in the basic form has 4 main messages.
1. Register Command
2. File List Request Command
3. File Location Request Command
4. Leave Request

A Client can send any one of these messages to the server and can get back a response. Explanation for each command is provided below.

Register Command

When the user invokes this command, the client gets 2 inputs from the user. One is the number of files to register and other is the list of file names (one by one). If for some reason, the file name inputted by the user does not exist, the client software prompts for the file name again. Next, the client computes the file size in bytes by reading the file from the disk and then packs the filename and the filesize along with the port in which it's listening in a structure and sends to the server. The Server dynamically allocates memory for the new structure and updates its database. For every registered file, server sends a success/ failure message to the Client.

File List Request

When the user invokes this command, the client requests the server to send the list of files which is present in its database. The server packs all the information in a structure and sends it through the socket. Client then displays the received information to the user.

File Location Request

Using this command, the client requests for the location of the file it wants to download. The user can choose any file he/she wants by inputting the corresponding number of the file as outputted by the File List Request command. The Server sends all the information about the file like the file name, file size and the port number of the client to get the file from. Again, all these information is packed in a structure.

As soon as the client receives the information, it starts contacting the peers (clients containing the files) and requests for parts of the file. For example, if there are 3 peers containing the same file, our client requests for the first 1/3rd of the file from peer 1, second 1/3rd of the file from peer 2 and the third part from peer 3. Reception from multiple peers happens simultaneously by using threads in the client side.

Once the entire file is received, the client then merges the files and removes the temporary storage files. After doing this, the client register the file to the server so that some other client can download from it.

Leave Request

When this command is given by the user, the client tells the server that it's leaving and asks the server to remove the files that it has registered from the server's database. Server removes the files and displays the remaining files registered with it.

Features & Limitations

The File download from peers is done simultaneously. The way it works is as follows. When a client requests for information about a file location from the server, the server replies with 2 set of data. One is the number of peers having the file and then the list of port numbers to contact the peers.

Once our client gets this data, it starts initiating connections with the peers. If for some reason one of the peers leave in the middle of the transmission abruptly, then the client tries to receive that part of the file from one of the working peers. If there was only one peer, then the client gives up and unregisters the misbehaving peer from the server. This way, there won't be a problem for some other client after this.

If there are multiple peers having the same file and if one of them leaves, the client gets the data from one of the working peers. The Limitation in such a case the download does not happen simultaneously. It becomes sequential. This limitation is due to the fact that we are downloading part of the file from each peer. To remove this limitation, we should implement a system where we download segment by segment from the peers instead of one big part of the file.

One important feature that is present in the system is the "Unregistering of the Misbehaving Clients". If a client finds out that a peer has left in the middle of the transaction abruptly, it automatically unregisters that peer from the server. This results in the removal of all the files registered by that particular peer from the server.

Also, if some client leaves abruptly, the client does not crash. It just tries to recover from the loss or in the worst case gives up getting the file. This is the fault tolerance implemented in the system. This is achieved by using Signal “SIGPIPE” which is triggered when an attempt is made to write into a closed socket. In that event, instead of just exiting the program (which is the default behaviour), we set a flag and indicate the program that something has gone bad with a peer. We identify which peer has misbehaved by using another flag.

Download packet size is set to 1024 bytes and can be modified to any suitable value as required by the user.

Server can handle multiple clients at a time. This is achieved by creating a new thread for each client who requests for some assistance from the server. The number of clients that can be handled at a time is limited to 5.

As explained earlier, the clients can also handle multiple peers at a time. Again this is achieved through creating threads for each client. Again, a client can handle a maximum of 5 peers at a time.

Many of the failure conditions have been accounted for. For example, if a user tries to register for a file which is not present in it’s working directory, we throw an error and ask for the file name again. We also give an option to abort registering of the files.

When choosing for a file to download, we ask user to enter a number between 1 and No. of files present. If the user enters anything else, we check for it and prompts the user to enter the number again.

One limitation which is not handled yet is what happens when a client tries to register a file with the same name as another client but of different size. We haven’t yet handled that possibility. This is because we use only the file names as the key in the database. To work around this problem, we can add one more level of checking to include the file size too.

Another limitation is that the system does not use IP address as of now. It just uses the port number to identify the client. This feature can be easily added.

Structure of the source code

Server Side

The Server opens a socket for receiving using bind and waits forever to be contacted by some client. As soon as a client sends data to the server, the server creates a new thread for it and again waits for some other client to contact it.

Inside the thread, the server waits for commands from the client. If it receives the Register command, it waits for the number of files information and then waits for the list of file names. Once it gets them, it adds them to the database and updates the number of files registered variable.

If it receives the File List Request command, it gets the information from the database and sends it across to the client.

If it receives the File Location Request command, it waits for the file name and then sends the information about the peers having the file to the client.

If it receives the Leave Request command, it gets the client information and removes the client's entries from the database.

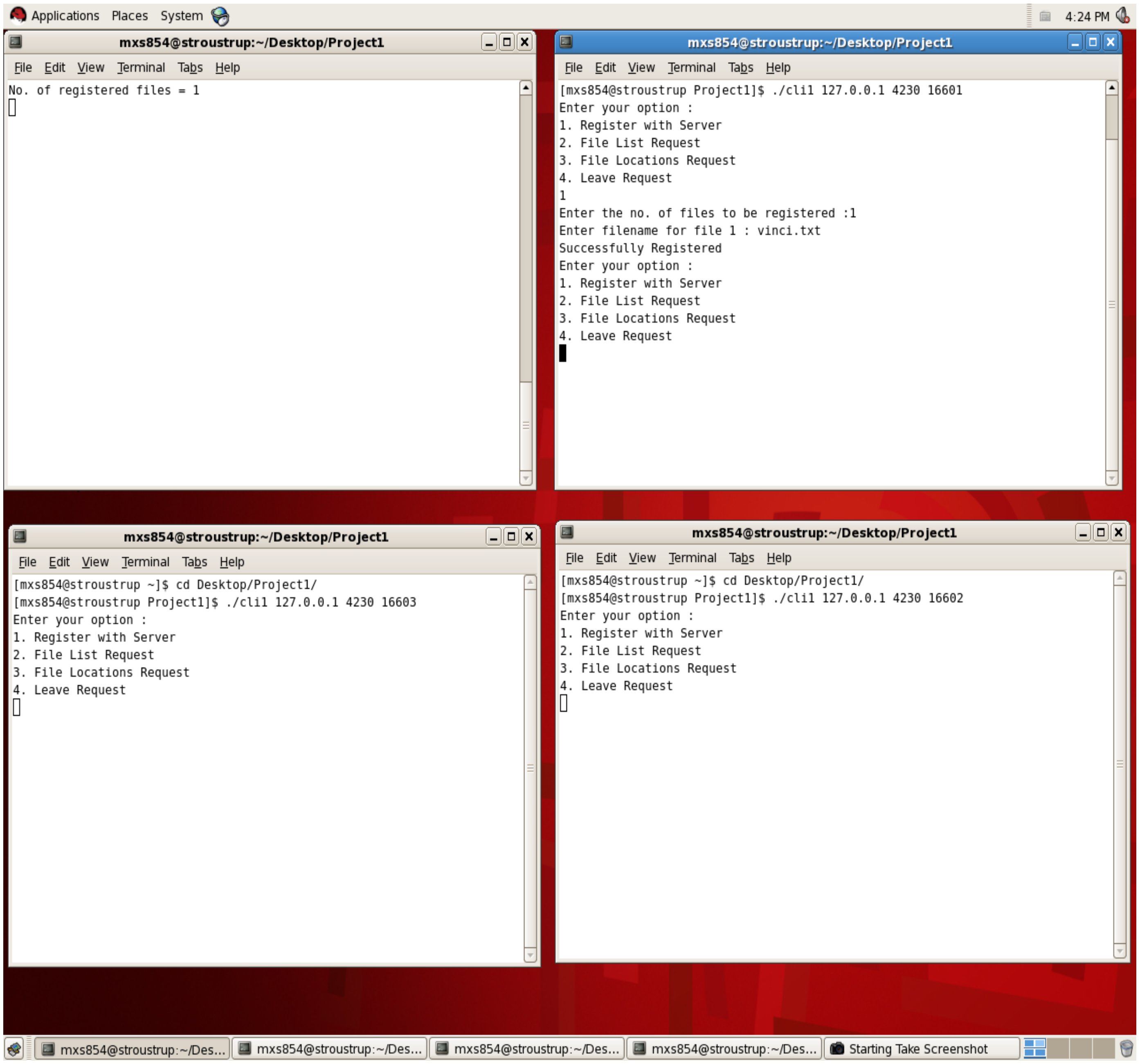
Client Side

First, the client creates a thread to listen to a port which is specified in the command line argument. It is through this port that the client can be contacted if some other peer wants a file from this client. The thread runs forever.

After that it waits for the user input. User can choose to register files, ask for file list, ask for file locations or request to leave. Based on the user input, the client sends the particular command to the server and the commands are implemented as explained in the previous paragraphs.

Screenshots

The following are the screenshots of various scenarios. We have three clients open. The top left window is the server. The other three are clients.



```
No. of registered files = 1
```

```
[mxs854@stroustrup Project1]$ ./cli1 127.0.0.1 4230 16601
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
1
Enter the no. of files to be registered :1
Enter filename for file 1 : vinci.txt
Successfully Registered
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

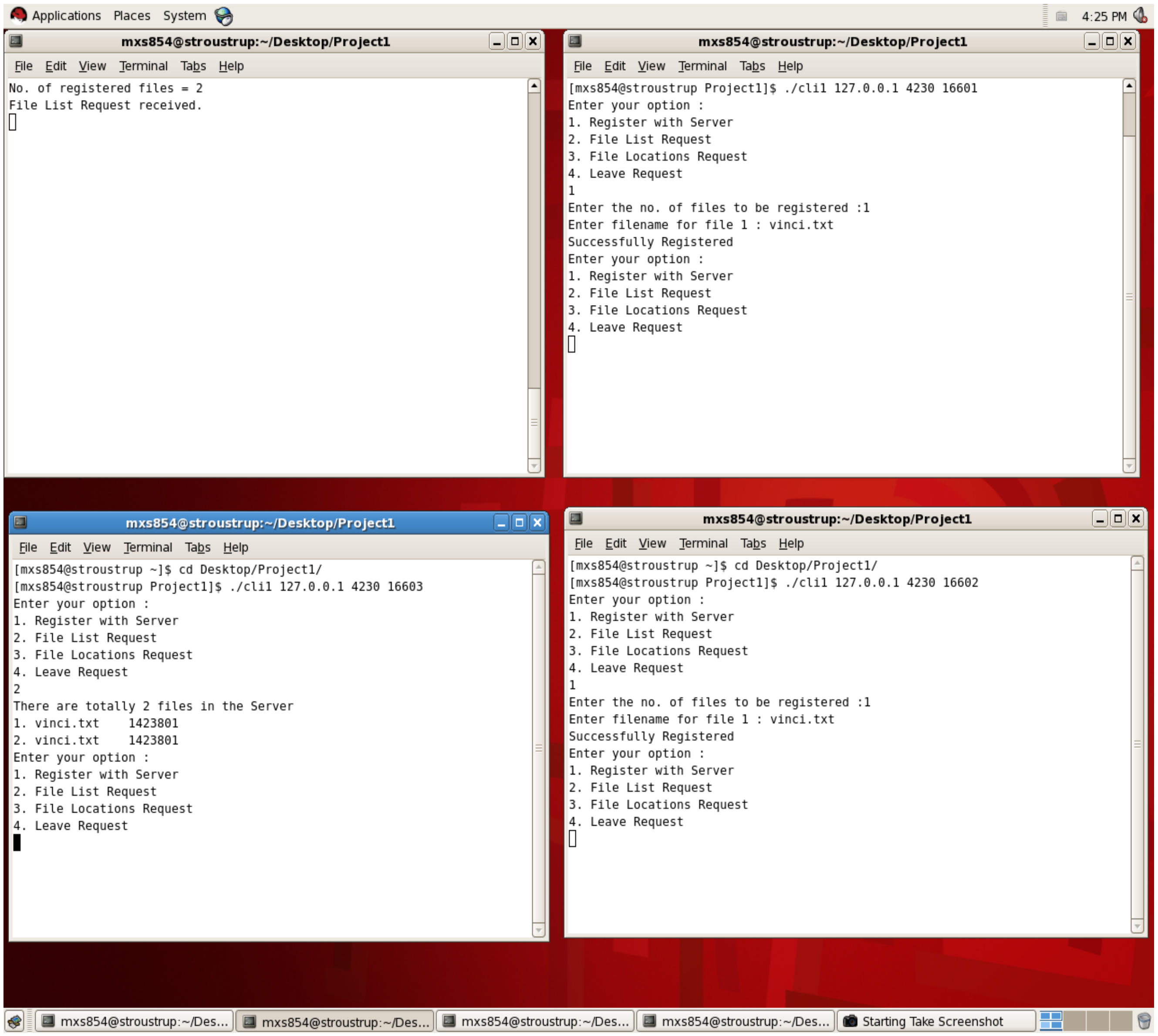
```
[mxs854@stroustrup ~]$ cd Desktop/Project1/
[mxs854@stroustrup Project1]$ ./cli1 127.0.0.1 4230 16603
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

```
[mxs854@stroustrup ~]$ cd Desktop/Project1/
[mxs854@stroustrup Project1]$ ./cli1 127.0.0.1 4230 16602
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

Applications Places System
4:24 PM
mxs854@stroustrup:~/Desktop/Project1
File Edit View Terminal Tabs Help
mxs854@stroustrup:~/Des...
Starting Take Screenshot

Register with Server:

This Image shows that the client in top right has registered a file by name “vinci.txt” with the server.



```
No. of registered files = 2
File List Request received.
```

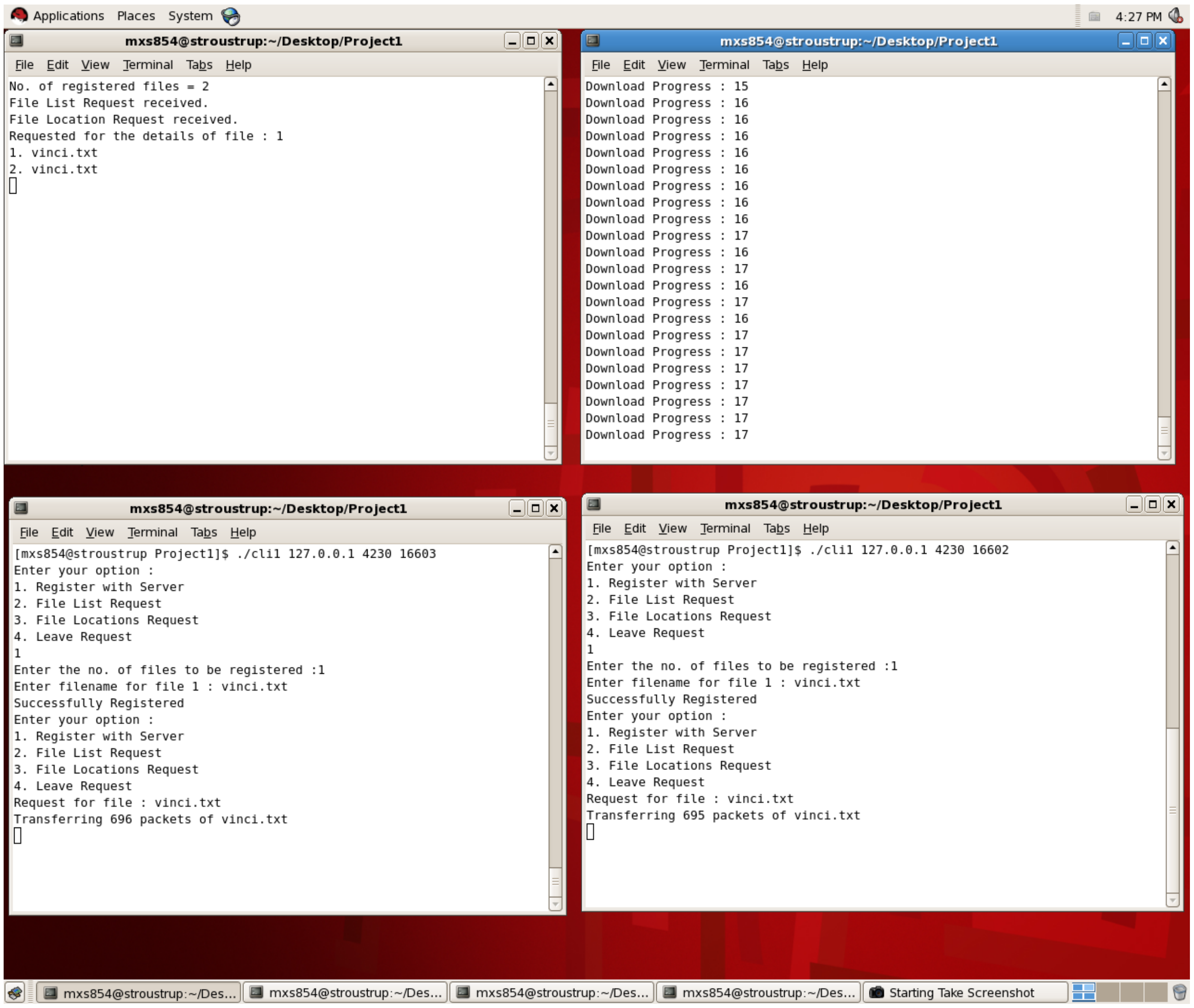
```
[mxs854@stroustrup Project1]$ ./cli1 127.0.0.1 4230 16601
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
1
Enter the no. of files to be registered :1
Enter filename for file 1 : vinci.txt
Successfully Registered
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

```
[mxs854@stroustrup ~]$ cd Desktop/Project1/
[mxs854@stroustrup Project1]$ ./cli1 127.0.0.1 4230 16603
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
2
There are totally 2 files in the Server
1. vinci.txt     1423801
2. vinci.txt     1423801
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

```
[mxs854@stroustrup ~]$ cd Desktop/Project1/
[mxs854@stroustrup Project1]$ ./cli1 127.0.0.1 4230 16602
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
1
Enter the no. of files to be registered :1
Enter filename for file 1 : vinci.txt
Successfully Registered
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

File List Request

This image shows that the client in the top right and the bottom right has registered the same file "vinci.txt" with the server. The client in the bottom left has requested for the files registered with the server and has got the response.


Applications Places System
4:27 PM
mxs854@stroustrup:~/Desktop/Project1
File Edit View Terminal Tabs Help
mxs854@stroustrup:~/Des...
Starting Take Screenshot

```
No. of registered files = 2
File List Request received.
File Location Request received.
Requested for the details of file : 1
1. vinci.txt
2. vinci.txt
```

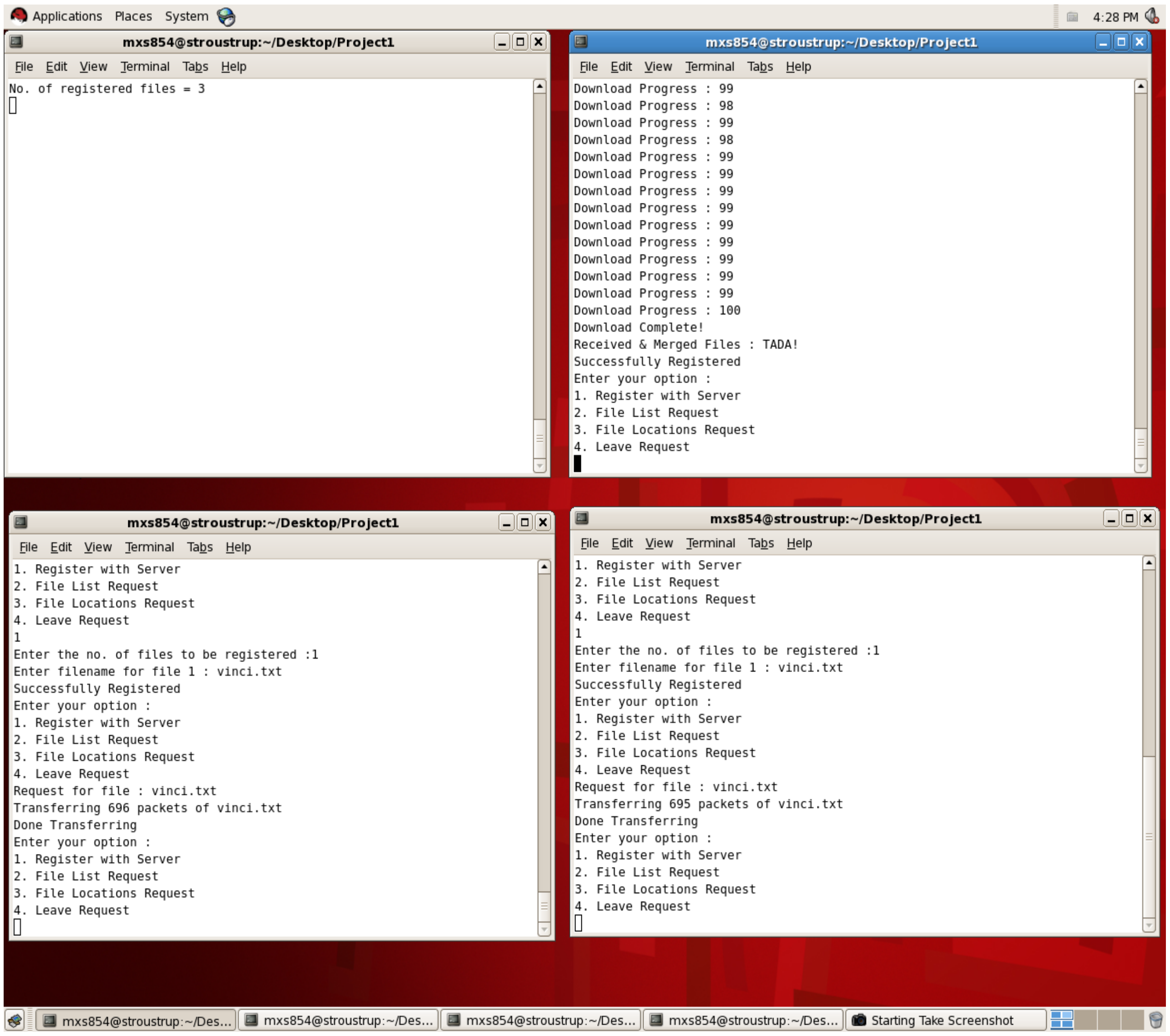
```
Download Progress : 15
Download Progress : 16
Download Progress : 16
Download Progress : 16
Download Progress : 16
Download Progress : 16
Download Progress : 16
Download Progress : 16
Download Progress : 16
Download Progress : 17
Download Progress : 16
Download Progress : 17
Download Progress : 16
Download Progress : 17
Download Progress : 16
Download Progress : 17
Download Progress : 17
Download Progress : 17
Download Progress : 17
Download Progress : 17
Download Progress : 17
Download Progress : 17
```

```
[mxs854@stroustrup Project1]$ ./cli1 127.0.0.1 4230 16603
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
1
Enter the no. of files to be registered :1
Enter filename for file 1 : vinci.txt
Successfully Registered
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
Request for file : vinci.txt
Transferring 696 packets of vinci.txt
```

```
[mxs854@stroustrup Project1]$ ./cli1 127.0.0.1 4230 16602
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
1
Enter the no. of files to be registered :1
Enter filename for file 1 : vinci.txt
Successfully Registered
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
Request for file : vinci.txt
Transferring 695 packets of vinci.txt
```

File Download in progress

This image shows that the clients in the bottom half of the image is transmitting two different parts of the same file “vinci.txt” to the client in the top right. The Receiving client is showing that the download is in progress and is showing the percentage of the file downloaded.


Applications Places System
4:28 PM
mxs854@stroustrup:~/Desktop/Project1
File Edit View Terminal Tabs Help
mxs854@stroustrup:~/Des...
Starting Take Screenshot

```
No. of registered files = 3
```

```
Download Progress : 99
Download Progress : 98
Download Progress : 99
Download Progress : 98
Download Progress : 99
Download Progress : 99
Download Progress : 99
Download Progress : 99
Download Progress : 99
Download Progress : 99
Download Progress : 99
Download Progress : 99
Download Progress : 99
Download Progress : 100
Download Complete!
Received & Merged Files : TADA!
Successfully Registered
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

```
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
1
Enter the no. of files to be registered :1
Enter filename for file 1 : vinci.txt
Successfully Registered
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
Request for file : vinci.txt
Transferring 696 packets of vinci.txt
Done Transferring
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

```
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
1
Enter the no. of files to be registered :1
Enter filename for file 1 : vinci.txt
Successfully Registered
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
Request for file : vinci.txt
Transferring 695 packets of vinci.txt
Done Transferring
Enter your option :
1. Register with Server
2. File List Request
3. File Locations Request
4. Leave Request
```

Download complete

This image shows that the Download has completed and that the downloaded file is registered with the server. Notice the no. of files registered value shown in the server window (top left).

How to execute the Program

The zip file included has 4 folders. Three folders are for the clients and one is for the server. All the clients have "vinci.txt" common in their folders. Apart from that there is one more file in them in case we need to test some other functionality.

The Programs are to be executed in the following order:

1. Go to the server's folder. Compile the server code by typing the following command :
   ```
   gcc serv.c -o serv –lpthread
   ```
2. To run the server code, type the following:
   ```
   ./serv 4230
   ```
3. Here 4230 is the port used by the client.
4. Go to the client 1's folder. Compile as follows:
   ```
   gcc cli1.c –o cli1 –lpthread
   ```
5. Execute the code as follows:
   ```
   ./cli1 127.0.0.1 4230 16601
   ```
6. Here `127.0.0.1` represents the localhost, `4230` is the server's port number and `16601` is the listening port of the client 1.
7. Repeat steps 4 and 5 for clients 2 and 3 while ensuring that the listening port given in the command line is different.
8. Once all the clients and the server are up, type `1` in one of the clients to register a file. It'll then ask for the number of files to register. Type `1` again. It'll ask for file name. Type `vinci.txt`. Server should acknowledge the file registration.
9. Repeat step 9 with one more client.
10. Go to the third client and press `2`. This will display all the files registered with the server. There should be two files.
11. Press `3` in the same window. This is to send File Location Request command. When it asks you to choose a file, press either `1 or 2`.
12. If everything has worked perfectly till this point, you should see two clients transmitting and the third client receiving the file. The Third client will also display a download progress screen.
13. Once the download completes, you can see that the newly downloaded file is registered with the server. The number of files registered in the server side should read 3.

Folder Structure:

1. Folder 1 : client1
   a. File 1 : cli1.c
   b. File 2 : vinci.txt
   c. File 3 : dracula.txt
   d. File 4 : head_vars.h
   e. File 5 : headfun.h

2. Folder 2 : client2
    a. File 1 : cli1.c
    b. File 2 : vinci.txt
    c. File 3 : grimm.txt
    d. File 4 : head_vars.h
    e. File 5 : headfun.h
3. Folder 3 : client3
    a. File 1 : cli1.c
    b. File 2 : vinci.txt
    c. File 3 : moby.txt
    d. File 4 : head_vars.h
    e. File 5 : headfun.h
4. Folder 3 : server
    a. File 1 : serv.c
    b. File 2 : head_vars.h
    c. File 3 : headfun.h