PREFACE

The Local Messaging System is a network based project. It has two applications namely Client and the Server. The Client is an GUI application which runs at client or user side. With the GUI the user can easily interact with the application and he does not require any programming knowledge to work with it. It provides communication facility to is send and receive messages, files by the clients or users who are connected to the network. The Server is an application which runs at the server system. It does not has GUI, so it has to be controlled through the command prompt. The Server Application acts as the post office, that is, it receives data from a client which is destined to another client who is on the network. The server application processes the received data and it pushes data to the destined system or to destined client.

Functionalities

Managing Clients

In case of any failure or system down at the Client side then the user has to manually manage it.

Getting active Client List

In Local Messaging System each client connects to the Server with a Unique name. At the server application a String type array is used to maintain the name of the connected clients. This array will be updated when a new client connects and disconnects from the server. One of the active Client requests for this array while sending message or files. The server sends the list of client name which are in the array. Actual implementation of this will be as follows. When a client requests for active clients' name, the server first sends a number which indicates the number of active clients. According to the received number the client prepares an array for storing the Clients name and also this number of Clients says the Synchronisation time for the Client, means the name of the clients passes from server to requested Client one by one instead of sending the whole array at a time. Consider an example if there are 10 active clients , then the server sends the name ten times one by one and the Client receives or waits for receiving the client name 10 times. Hence the number of active clients notifies the synchronisation time.

Message passing

The message passing technique is little complicated. When a client wants to send the message, first it sends a Unique code that notifies the server that the client is going to send the message. Unique code is a unique number which is used to identify the action that is, in Local messaging System two different numbers are used in which one number identifies the Message passing and another Number identifies the File transfer. Then the server starts to run the message passing block or segment and automatically sends the list of active clients. Accordingly the Client automatically get ready or prepares to receive the active client list after sending the unique code. So the unique code also notifies the server to send the active Client list.

After the client receives the active client list from the server, the user gets the new GUI window containing place to write the message and the active client list specifically in the dropdown box. Then the user can fill up message and he can select the desired client from the dropdown box. After the client click on the send button , the selected client's name is sent to the server. The server receives that.

Before explaining the further process of server after receiving the destination client name, some techniques has to be explained. At the server a separate thread is created for each client and these threads are named according to the Client name that they have given at the Connection establishment time. Each client deals with thread which is created for them. Some global variable is used at the server like *Destination*, *Message*, *SenderName* which is accessible for all the threads. When the server receives the destination Client name it places that in a temporary variable and after it receives the message and the sender name from the client, stores these in a temporary variable. Now the Client's message passing task is complete. Then the server updates the Global variable *Message* with the received message from the client and also *SenderName* and

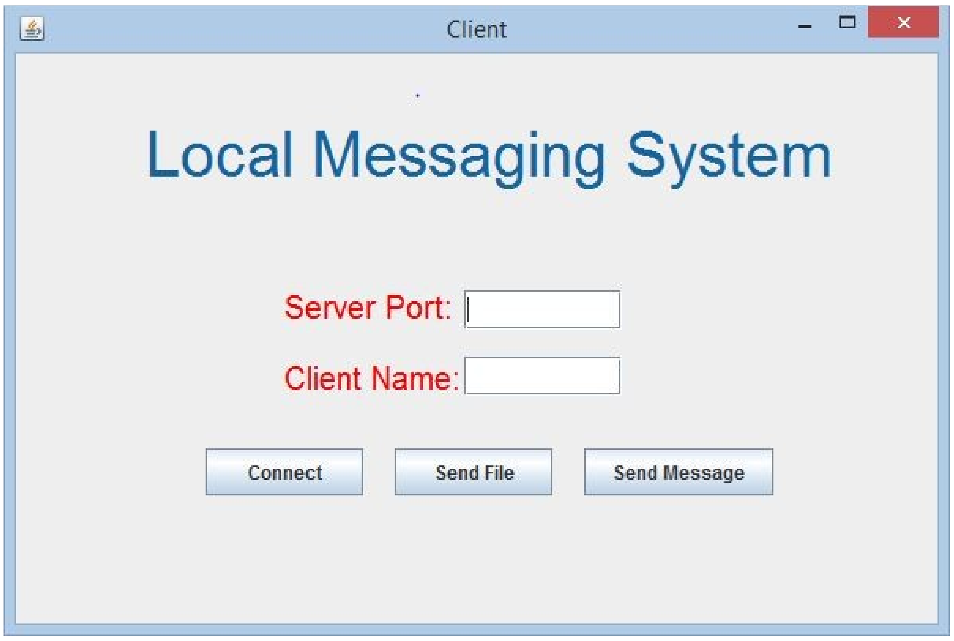
Destination. All the threads which are running at the server keeps checking the *Destination* with their thread name. The Thread whose thread name and *Destination* matches that thread is going to get the message data from *Message* variable and Sender name from the *SenderName* variable and then that thread pushes those data to the client that thread belongs to. Like this the Message passing technique is implemented in Local Messaging System.

File transfer

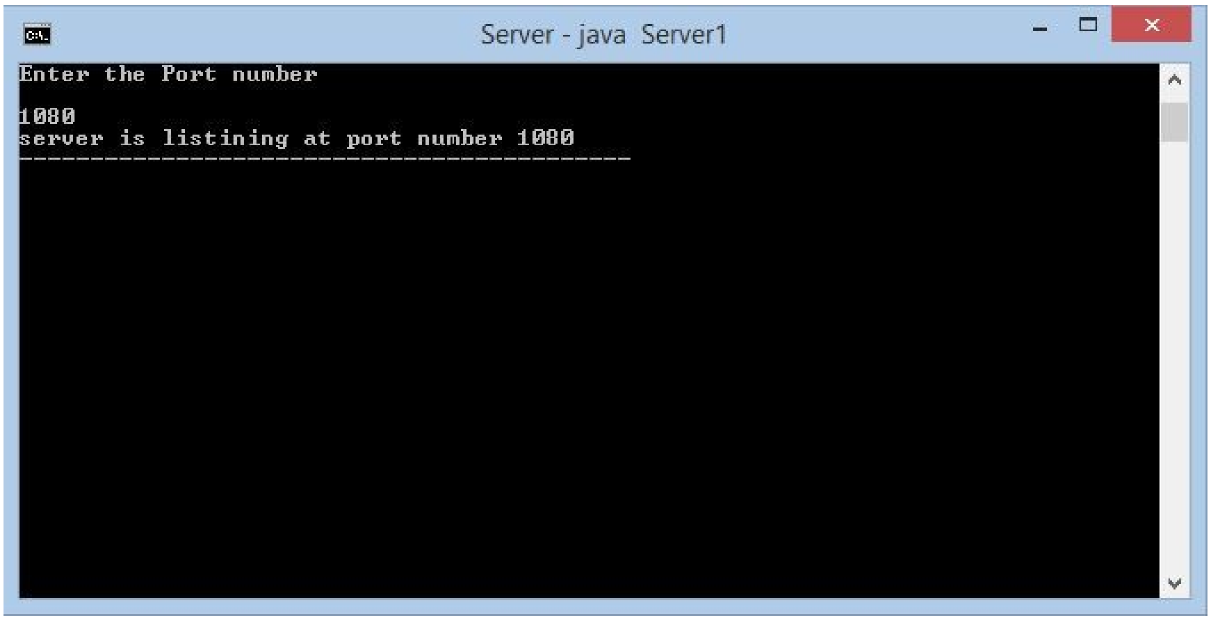
When a client wants to send a file, first it sends an Unique code that notifies the server that the client is going to send the file and then the server starts to run the file transfer block or segment. When the server starts to run the file transfer segment it automatically sends the list of the active clients. Accordingly the Client automatically get ready or prepares to receive the active client list after sending the unique code. So the unique code also notifies the server to send the active Client list.

After the client receiving the active client list from the server, the user gets the new GUI window containing a button to select the file and the active client list specifically in the dropdown box. When the click on the Select File button, a new window will get open showing the contents of the hard disk. After selecting the file the user have to select the desired client from the dropdown box. After client click on the send button , the selected client's name will be get from the dropdown box and sent to the server. The server receives that. Some global variable other than used for message transfer, is used at the server like *Destination*, *File*, *SenderName* which is accessible for all the threads. When the server receives the destination Client name it places that in a temporary variable and after it receives the file and sender name from the client and stores these in a temporary variable. Now the Client's file transfer task is complete. Then the server updates the Global variable that is *File* with the received file from the client and also *SenderName* and *Destination*. All the threads which are running at the server keeps checking the *Destination* with their thread name. The Thread whose thread name and *Destination* matches that thread is going to get the File from File variable and sender name from the *SenderName* variable and then that thread pushes those data to the client that thread belongs to. Like this the File passing technique is implemented in Local Messaging System.

SCREEN SHOTS



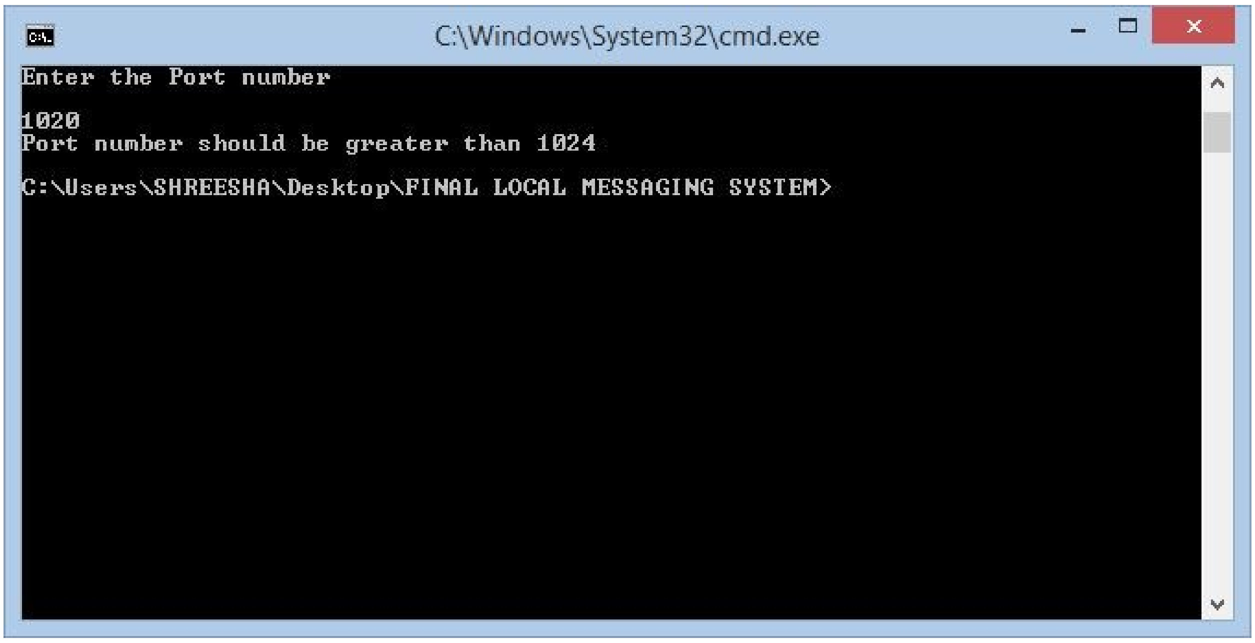
Above is the screen shot of the GUI Client application which runs at the client side. It contains two fields, *Server Port* and *Client Name*. In the *Server Port* field the user has to provide the port address of the server that should match with server's` port number and in *Client Name* field a unique name has to be provided and other clients in the network uses this name as destination while sending files and message and files. *Connection* button establishes the connection with the server. *Disconnect* button is used to disconnect the connection with the server. *Send File* button opens a GUI window which provide necessary options to send a file and *Send Message* button opens a GUI window which provides necessary options to send message.



Server - java Server1

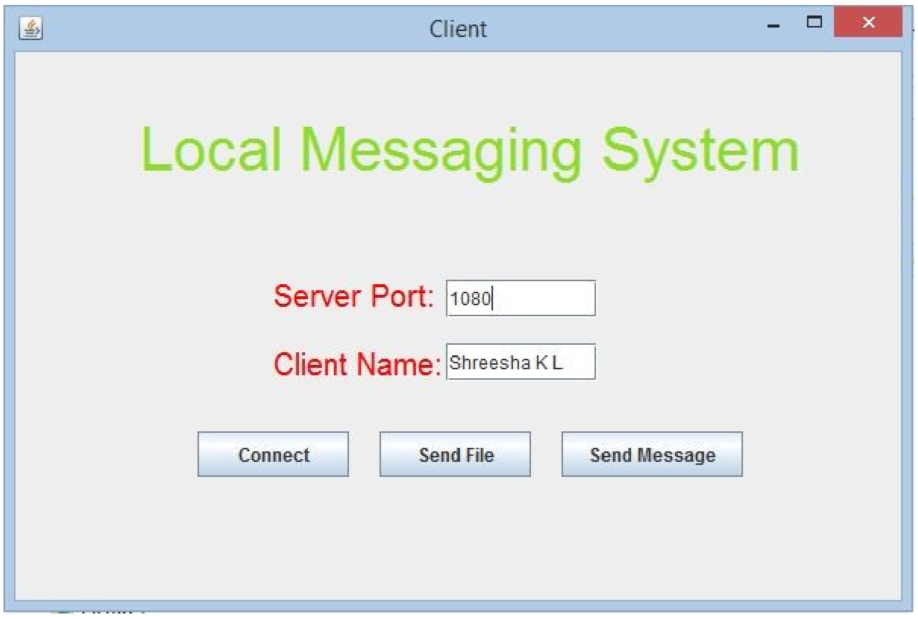
```
Enter the Port number

1080
server is listining at port number 1080
-------------------------------------------
```

This is Server application which runs at server system and it does not contain GUI, so it has to be controlled through the command prompt. First the administrator or user who runs and manages server has to provide the port number on which the server is listens to continuously for the request.



```
Enter the Port number

1020
Port number should be greater than 1024

C:\Users\SHREESHA\Desktop\FINAL LOCAL MESSAGING SYSTEM>
```

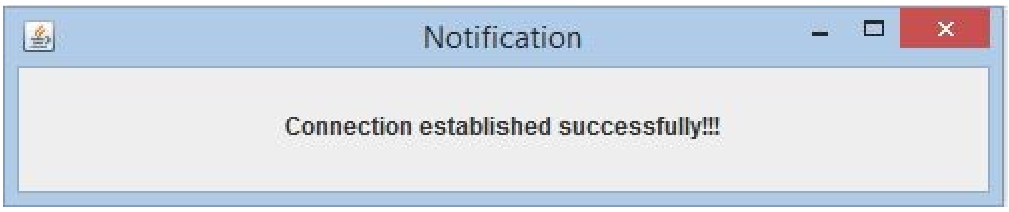
The port number of the server must be greater than or equal to 1024. The numbers up to 1024 are reserved port numbers. If a wrong port number is given then the above error will occur.



Server Port: 1080

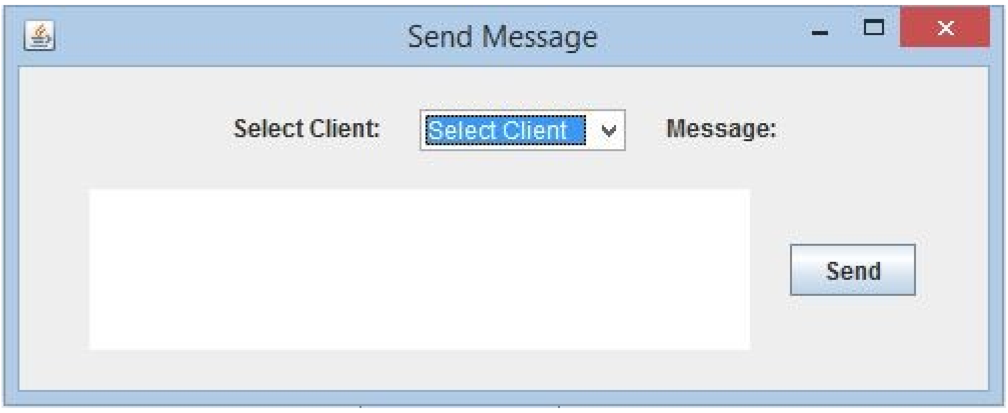
Client Name: Shreesha K L

This is the Client application containing Server port and Client Name.

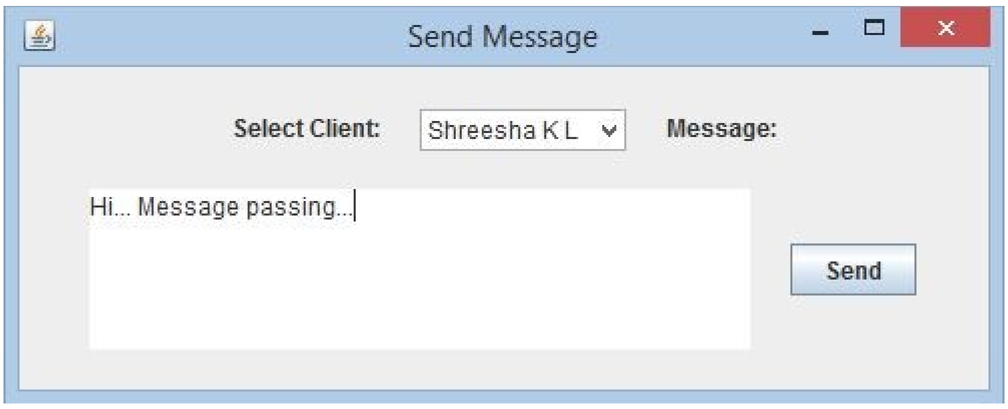


Connection established successfully!!!

This notification window will be shown when the connection with the server is established successfully.

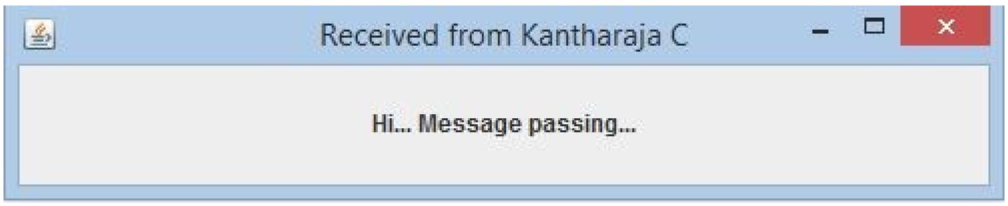


This is the window which provides facility to send the Message. Dropdown list contains the active clients' name. In the white field message has to be written which is needed to be sent. Send button sends the message to the selected client.



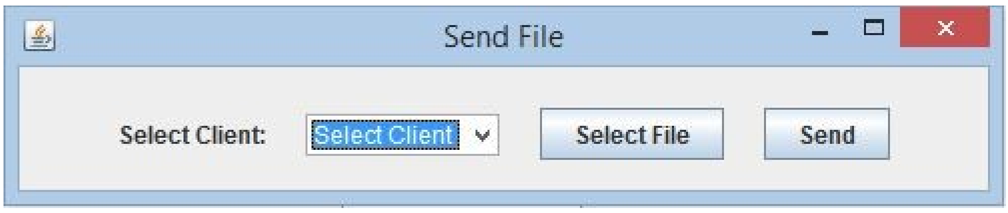
Hi... Message passing...

Message window with filled information. Here the user is going to send the message to another user called *Shreesha K L*. The message is "*Hi... Message passing*...".

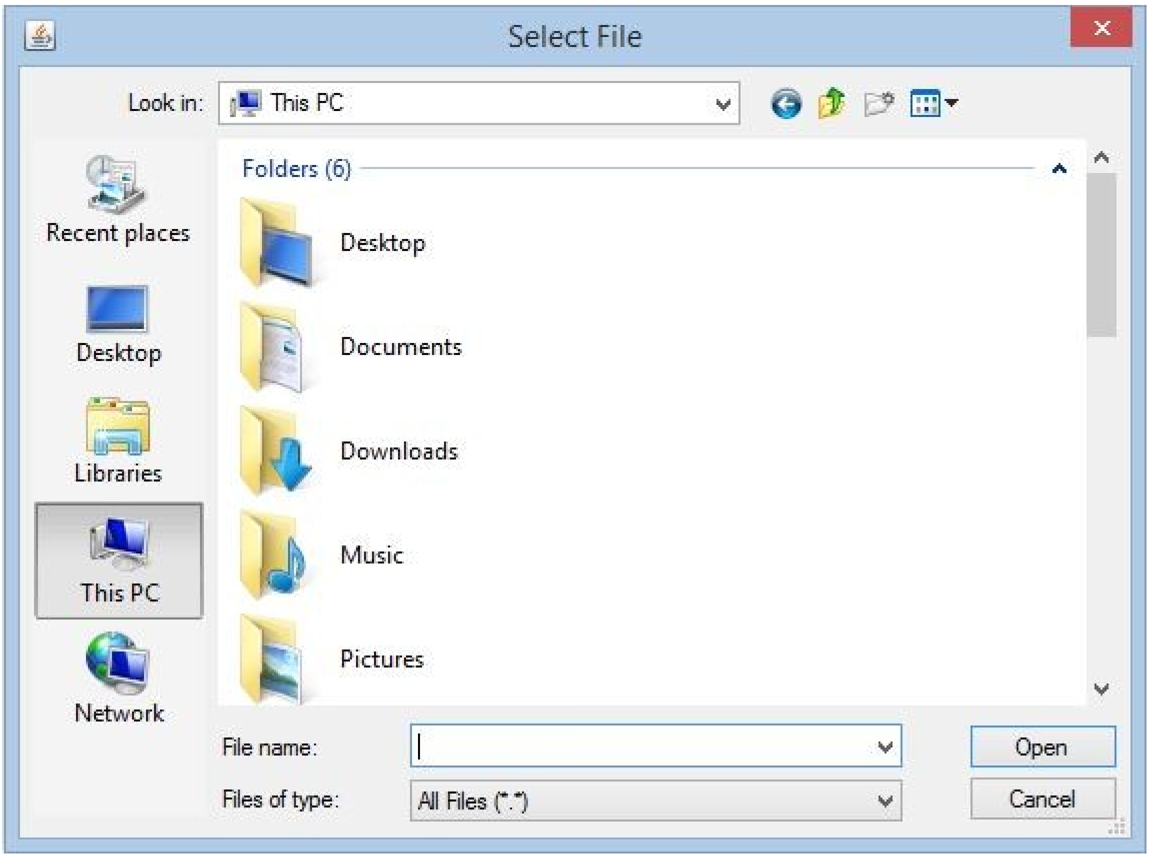

Received from Kantharaja C

Hi... Message passing...

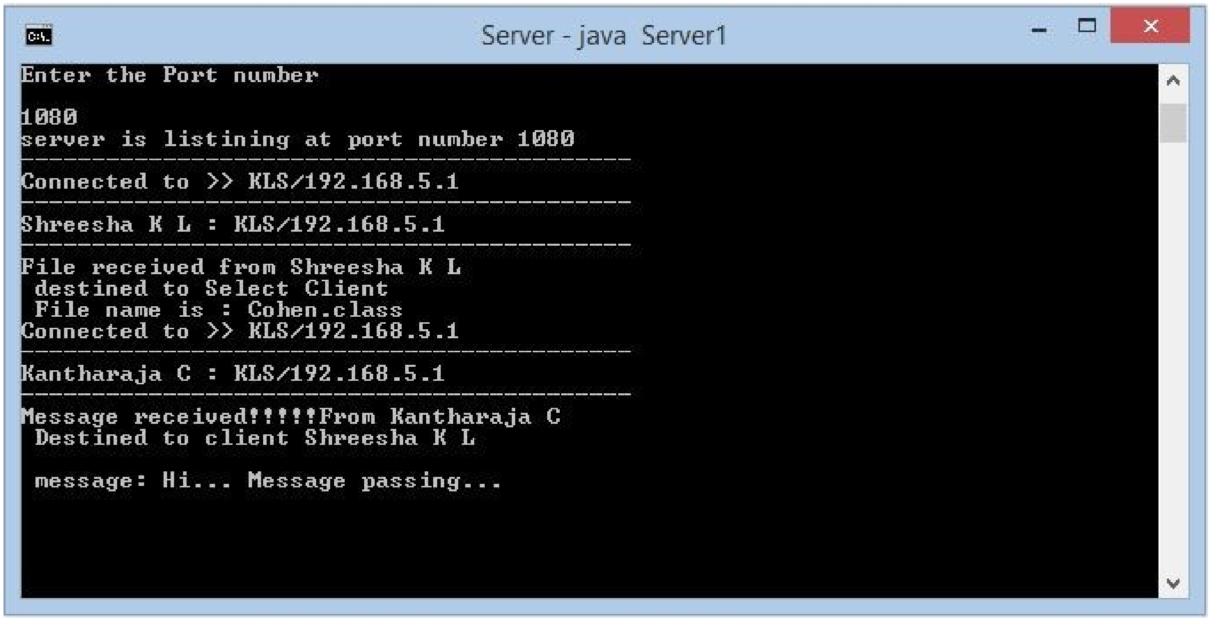
The window is displayed at the destination when it receives the message. The title of the window contains the message sender’s name.



This window provides the facility to send a file. Select File button opens another window to choose the file from hard disk. Send button sends the file to the selected client.

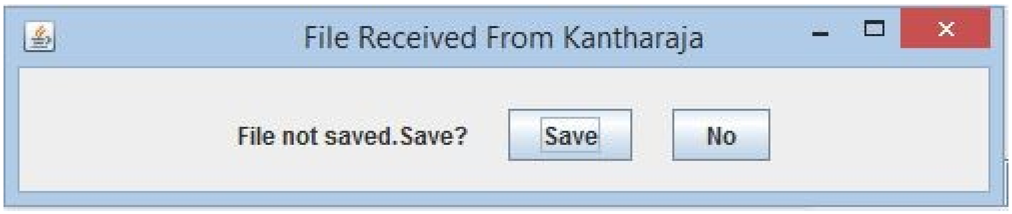


The above window will be opened when the user clicks on the Select File button. From this window the user can easily select the required file to be sent, anywhere from the system.

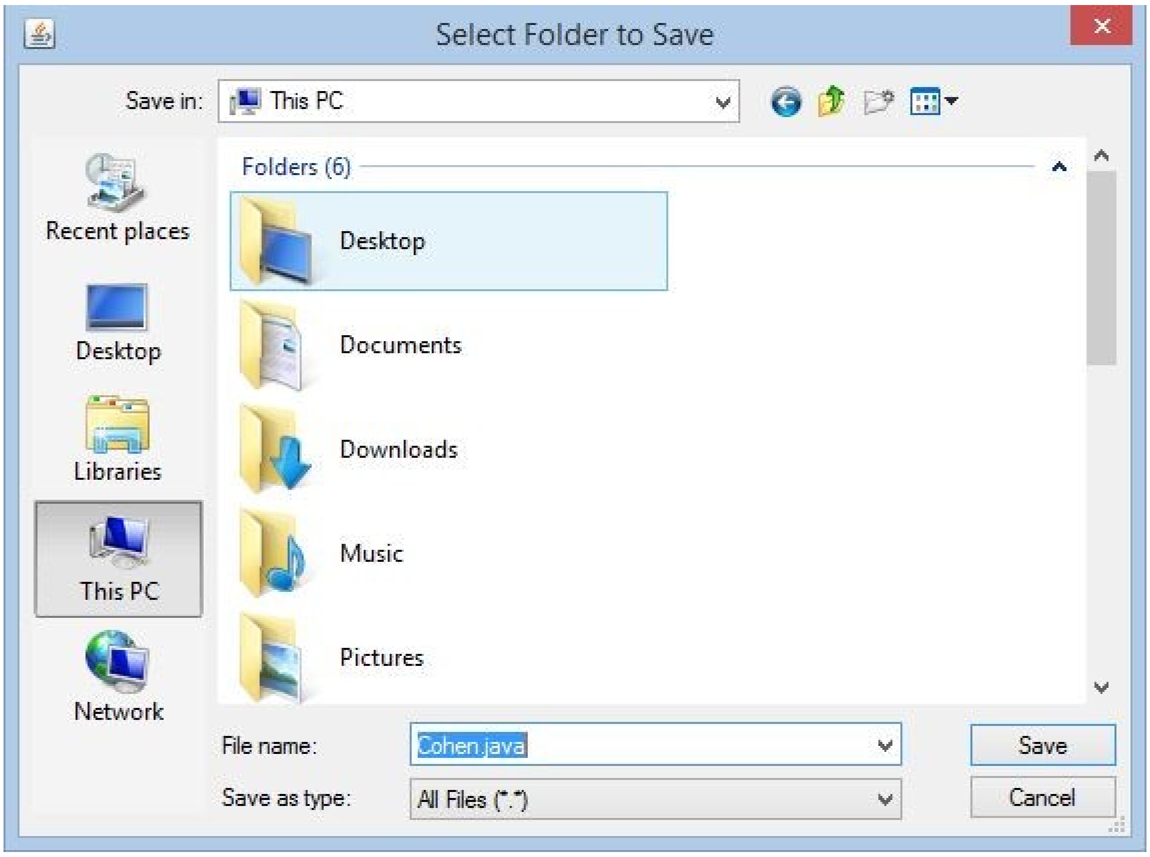

Server - java Server1

```
Enter the Port number

1080
server is listining at port number 1080
-------------------------------------------
Connected to >> KLS/192.168.5.1
-------------------------------------------
Shreesha K L : KLS/192.168.5.1
-------------------------------------------
File received from Shreesha K L
 destined to Select Client
 File name is : Cohen.class
Connected to >> KLS/192.168.5.1
-------------------------------------------
Kantharaja C : KLS/192.168.5.1
-------------------------------------------
Message received!!!!!From Kantharaja C
 Destined to client Shreesha K L

 message: Hi... Message passing...
```

This is a server window showing some file transfer and message transfer information between the clients.



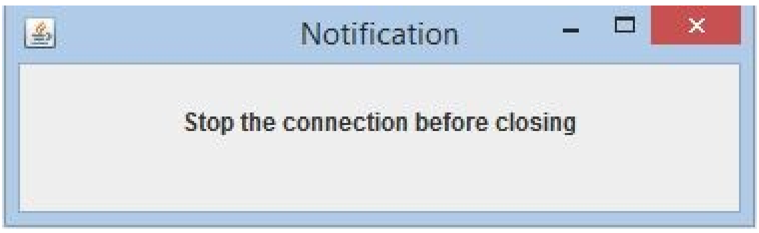
When a file is arrived at the Client that will not be saved automatically. Above shown window will be opened and if the user choose to save the file then a new GUI window will be opened else the file will be discarded.



File name: Cohen.java

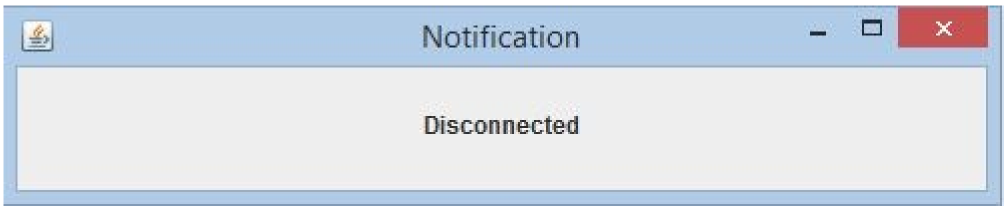
Save as type: All Files (*.*)

When the user choose to save the received file the above window will be opened. By this the user can easily select the place to save the file and name of the file will be automatically given.



Stop the connection before closing

All are very busy now-a-days. Hence there is more possibility that the user may close the Client application directly without pressing the Disconnect button. During this point of time above notification window will be displayed.



Disconnected

This notification window will be displayed when the connection terminated.

Kantharaja C	Divyashree R	Shreesha K L	Avinash G	Kavyashree

Logical array of clients' list

This is a logical diagram of active clients' list which will created at the server. The data within the rectangular boxes are the name of the clients which is provided at the time of connection establishment. When a client requests the server for the active clients' list, this array will be accessed and the name of the clients which are in this array will passed into the requested client. This array will be updated each time when a new client connects to the server or disconnects from the server.

Test Cases

In the Client application, if the Server Port number is given with a String or text instead of number, a 'Number Format Exception' will occur.

Another Important issue is about the Message and File passing technique. If a client sends the message or file to other client, no other messages or files sent by other clients to some other client, will pass at the server until the first sent message get passed or sent to the destined client. This may a major negative point because if any message is sent to the unconditionally disconnected client, then all other next coming messages at the server will not be passed.

Other than text files are not supported after receiving at Client.

Conclusion

The Local Messaging System is an application which provides a communication facility between the Clients through the Server system. By this application, clients or user who are connected to the server can exchange messages and files between them. This application can be implemented in small business firms, academic environments, hospitals many more.

The valid port number at which the server is listening must be given and messages and files must be sent to only connected clients in order to increase the efficiency. Direct closing of the Client application without disconnecting causes the server to behave in improper manner.

Bibliography

Text books:

1. Behrouz Forouzan A, *TCP/IP Protocol Suite*, 4th Edition, Tata McGraw-Hill Edition, 2010, pp 568

2. Andrew Tanenbaum S, *Computer Networks*, 3rd Edition, Eastern Economy Edition, 1999, pp 9

Web Reference:

1. **http://searchnetworking.techtarget.com/definition/Address-Resolution-Protocol-ARP**
   Routing mechanism.

2. **http://en.wikipedia.org/wiki/Computer_network**
   Information about computer networks .

3. **http://en.wikipedia.org/wiki/Routing_table**
   Information about Addressing Table.

4. **http://en.wikipedia.org/wiki/File_Transfer_Protocol**
   Meaning of FTP, running modes, data transfer modes, data representation in FTP.

5. **http://www.deskshare.com/resources/articles/ftp-how-to.aspx**
   FTP working mechanism.

6. **http://en.wikipedia.org/wiki/IP_address**
   Meaning of IP address.

7. **http://www.answers.com/Q/The_two_parts_of_MAC_addresses**
   Meaning, parts, diagram of MAC address.