NA-1

Network Architecture-1
Project-2

Fall 2016

Submitted by:

Moulika Chadalavada (16234180)
Lakshmi Nikitha Kona (16231555)
Sri Sai Anusha Gandu (16230560)

Table of Contents

1. Introduction

In this project, we are developing a simple Chat application. TCP Client and Server programs are executed for communication between resources using GENI. In Question 1, we create a chat server which accepts a single client connection and displays everything the client types and the communication get terminated from both sides when the client types 'exit'. In Question 2, extending the Question 1, when one client terminates, the server remains open for the other client's other clients that wish to communicate and can handle only one connection at a time. In Question 3, extending the Question 2, the server will be able to handle multiple clients at a time and the server window displays the messages from the clients. In Question 4, extending the Question 3, the server even echoes the messages from a client to all the remaining clients in the network.

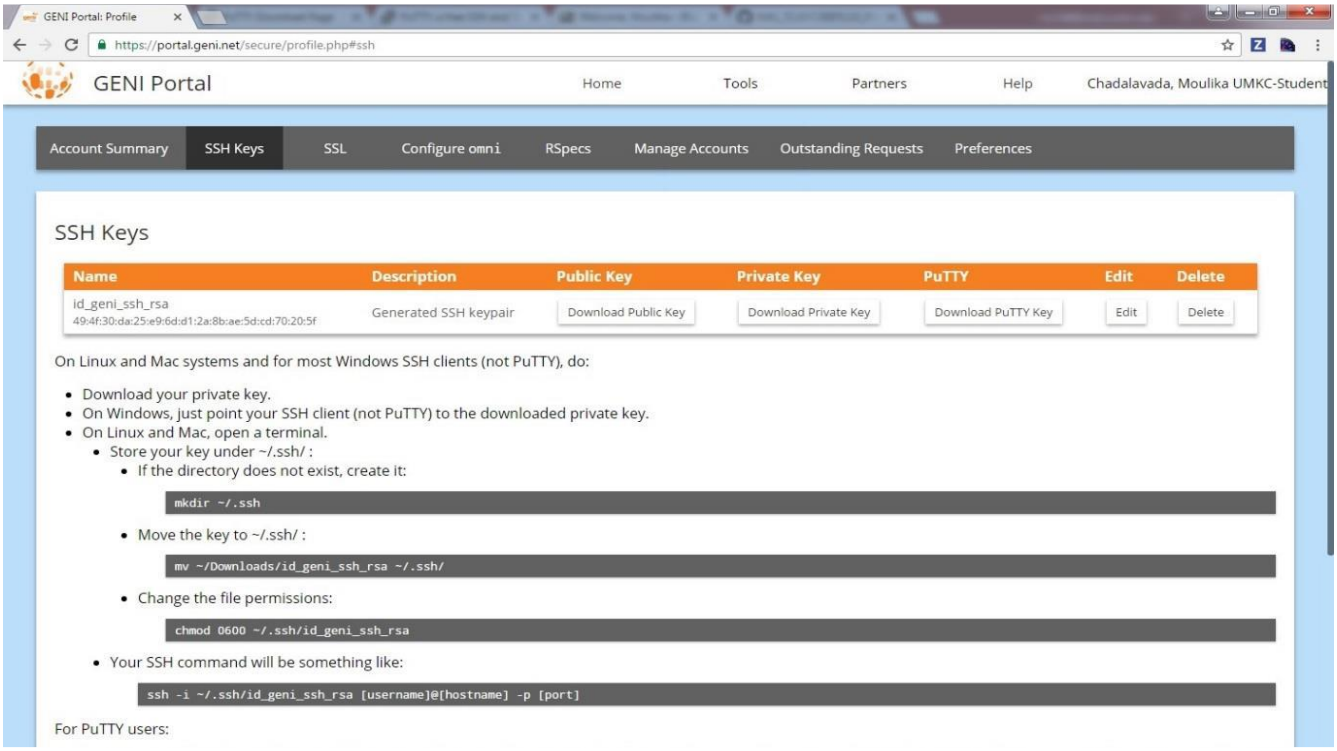
2. About GENI and Putty

To continue with the project, we need to first create a GENI account and a slice where we can reserve resources on which to work on.

GENI Account Creation:

GENI (Global Environment for Network Innovations) provides a virtual laboratory for networking and distributed systems research and education. It is well suited for exploring networks at scale, thereby promoting innovations in network science, security, services and applications.

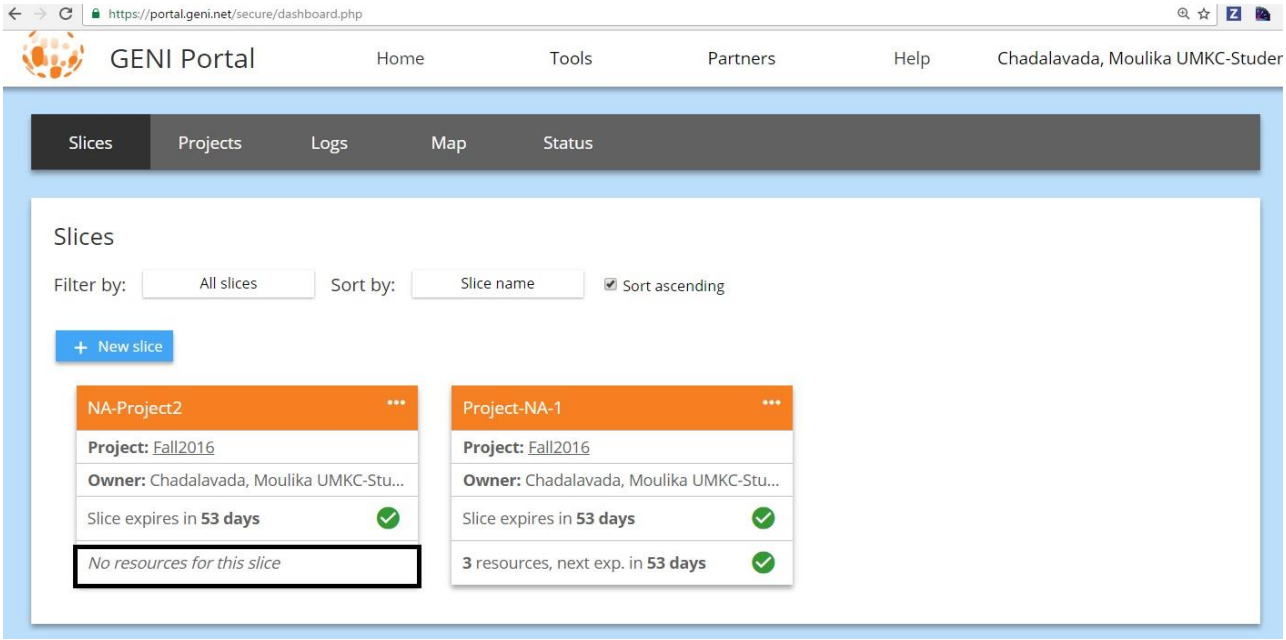
1. Login to portal.geni.net
2. Activate the GENI account with the required credentials.
3. Download the SSH keys (Putty) for authentication process.
4. Using the Putty key generator, generate a private key which will be used to open the Client and Server windows.



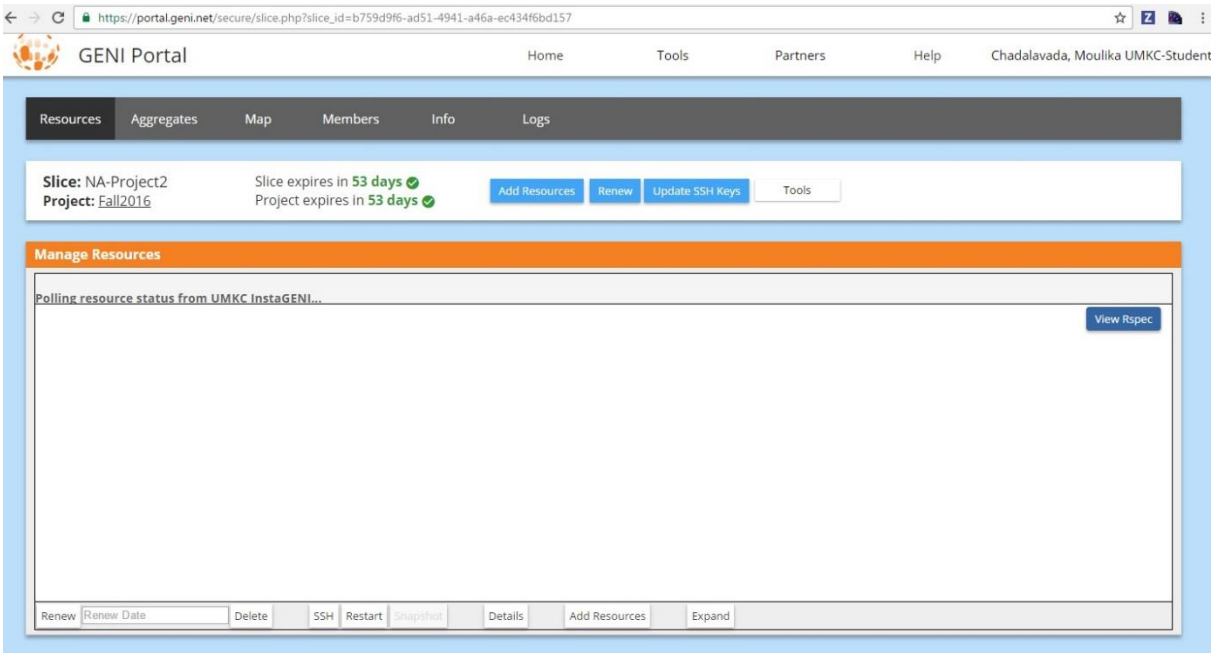
SSH Keys

Slice Creation:

Create a slice from where the resources can be reserved. For this project, New Slice 'NA-Project2' is created, where initially there are no resources which are reserved.



Slice NA-Project2 Creation

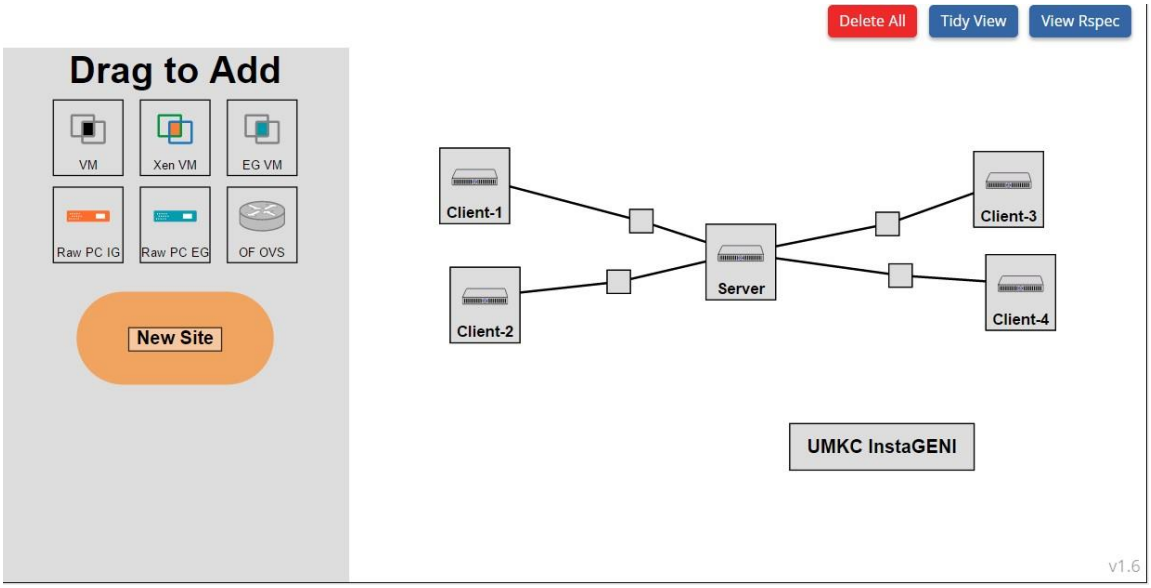


No Resources under Slice

Resource Reservation:

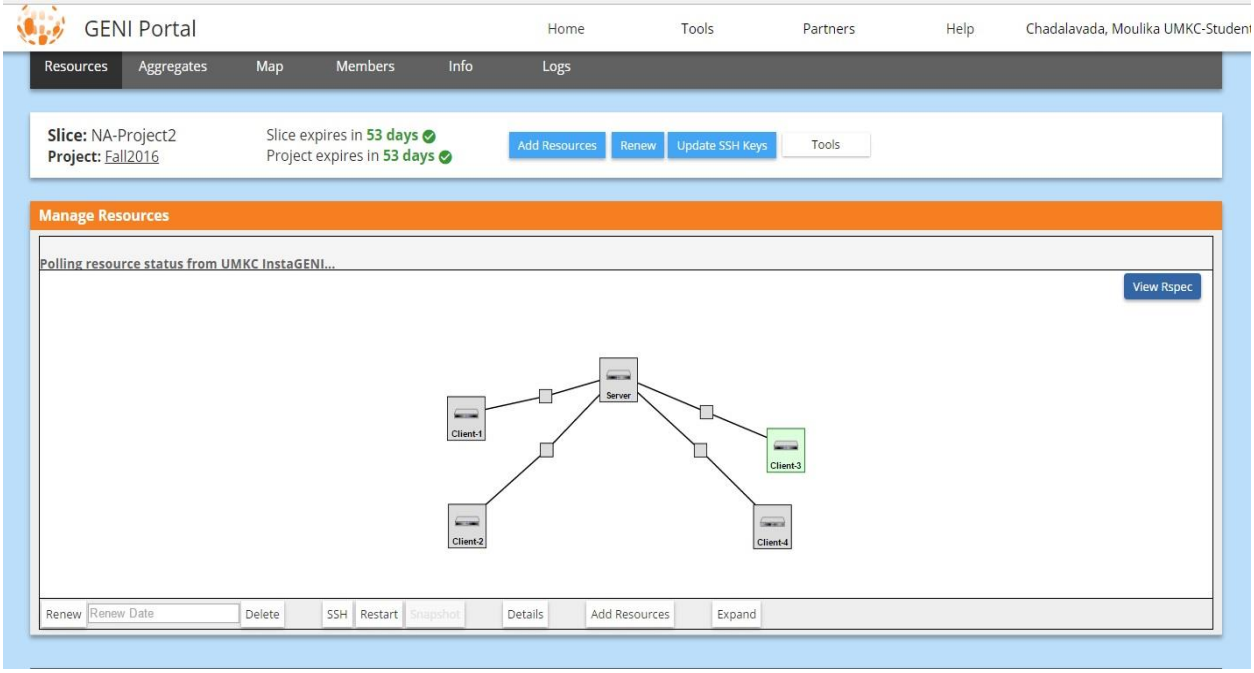
In the Slice under Add Resources, where VM's are added named them as 'n' Clients and Server. Also, established connection (link) between them and provided the IP address and subnet mask along with bandwidth in the link interface.

Initially 4 Client's and 1 Server is added for this project and Reserved these resources.

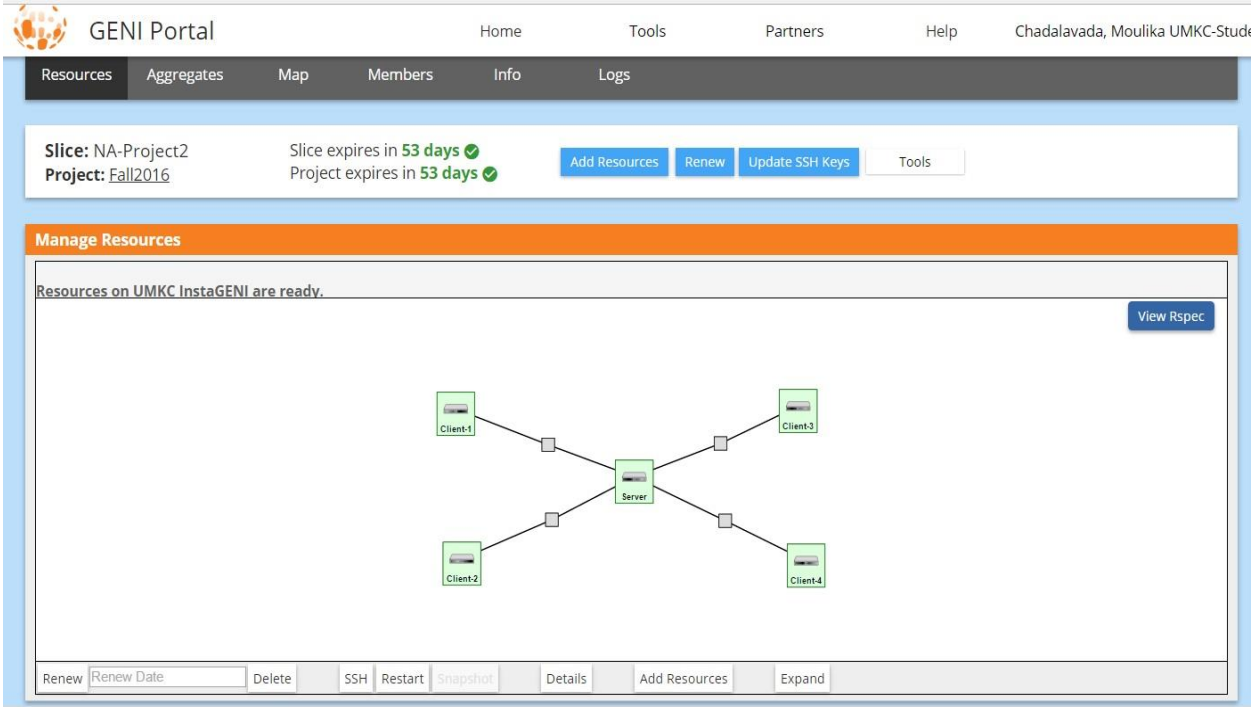


Adding VM's under Slice

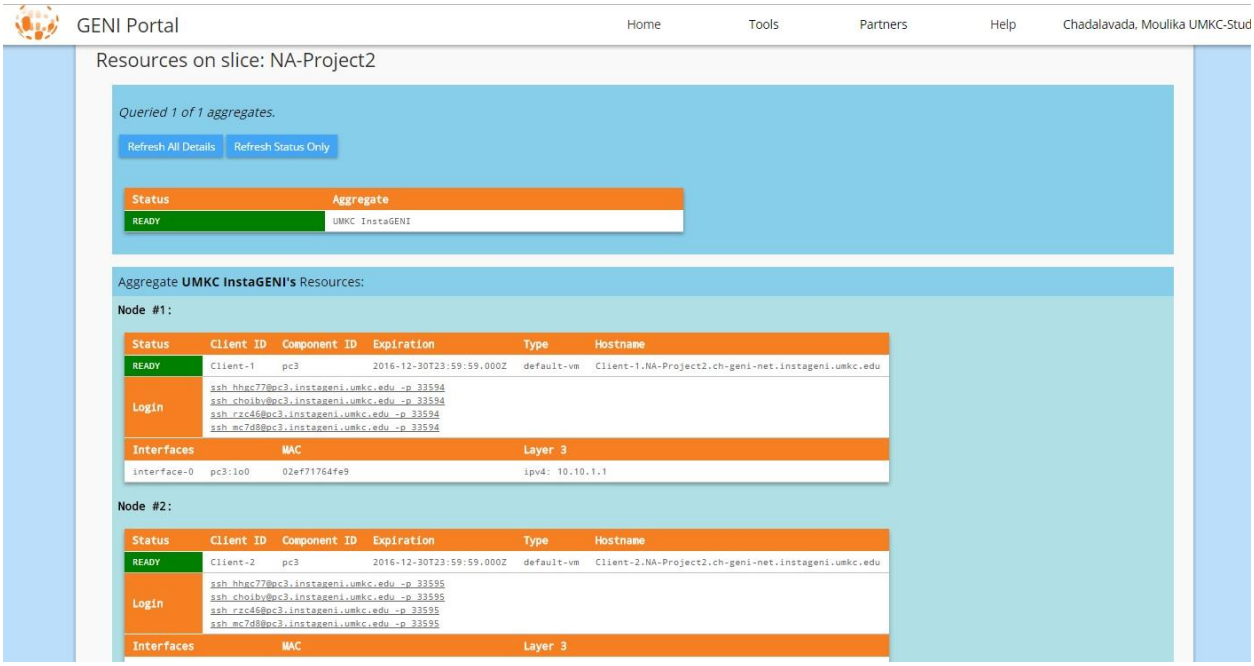
After adding the resources the status is unknown and later the resources are ready as shown below.



Resources when status is unknown

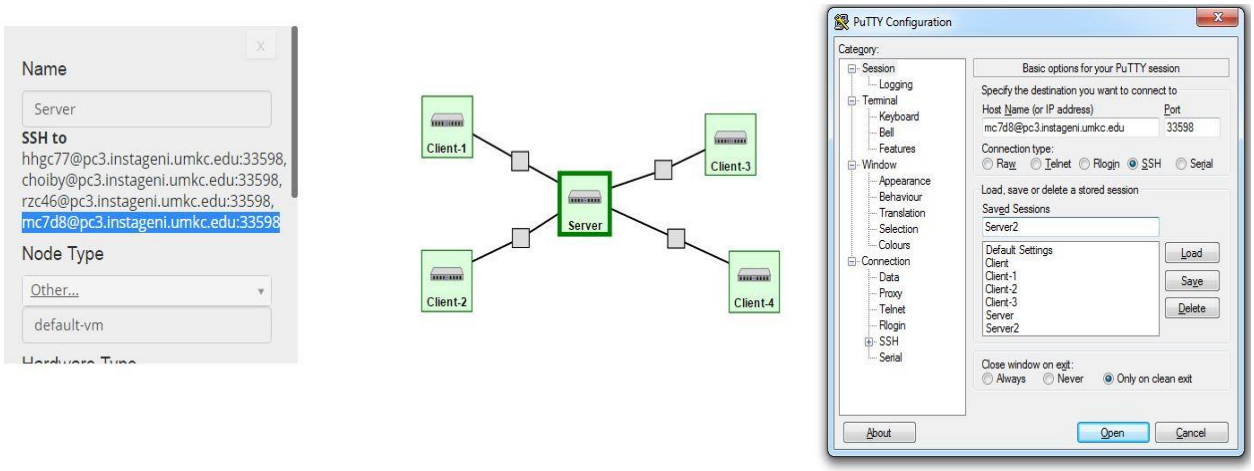


When all the resources are ready

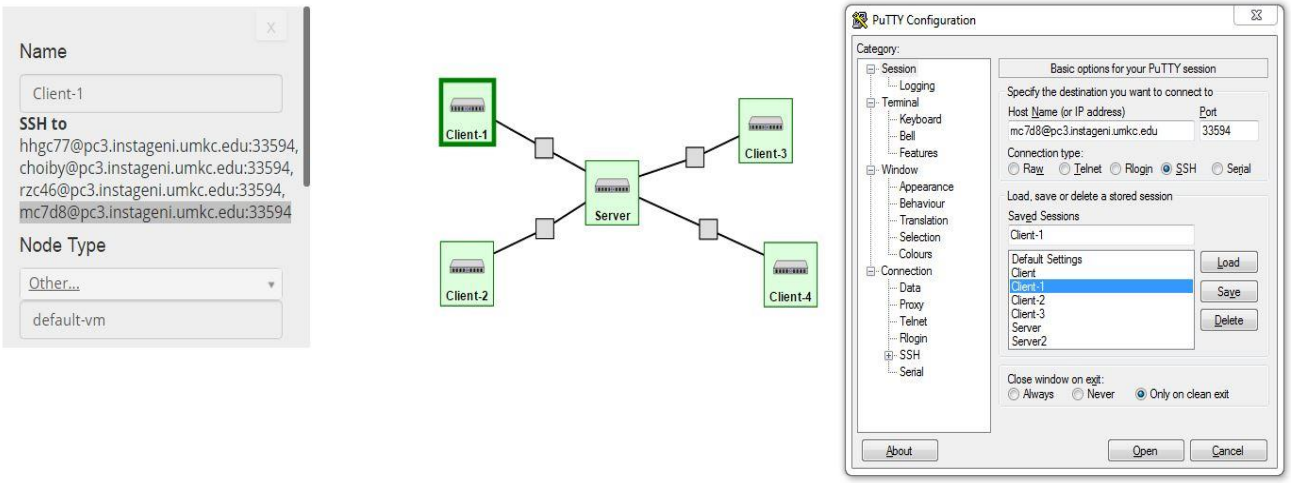


All Client's and Server resources are Ready

Once the resources are created in Slice and SSH keys are downloaded, in **puttygen** private key is generated using Putty key are is downloaded from GENI. Once Private key is generated in putty, logged in to VM's of Clients and Server with its respective Host Name and Port Number. Also before logging in Private key is loaded under SSH -> Auth.



Opening Server in Putty by giving Valid Details



Opening Client in Putty by giving Valid Details

3. Project Description

Need to develop a simple chat program (like google hangout and skype chat).

3.1 Question-1

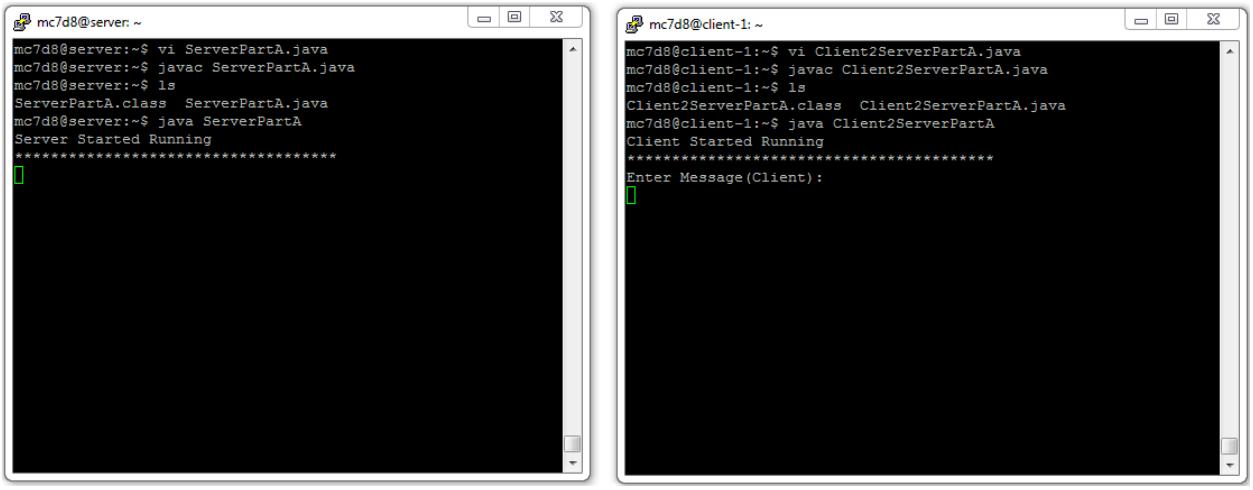
A chat server will accept a single client connection and display everything the client types. If the client user types 'exit', both client and server will end the program.

Solution:

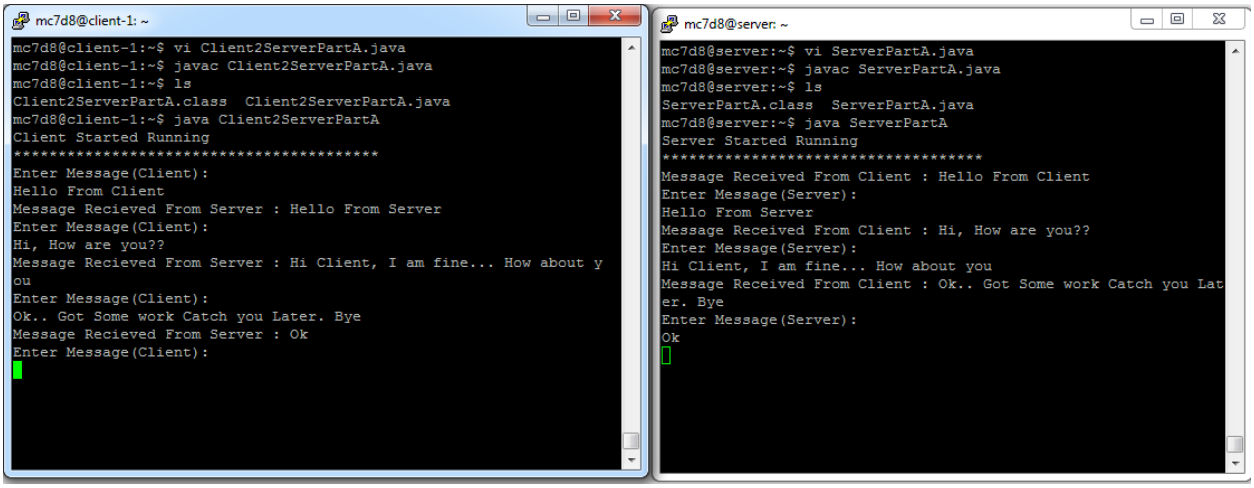
In this query connection is built between Client and Server, messages are sent from Client to Server and vice-versa. Once the Client enters '**exit'** both Server and Client exits from Chat Application. Java program is written for each Client **(Client2ServerPartA.java)** and Server **(ServerPartA.java)** through which messages are received and sent back.

Once the java files are compiled and the class file is executed, Client and Server connections are started as shown below. Once chat between Client and Server is completed, Client sends 'exit' which is received by the Server and the connection between Client and Server gets terminated.

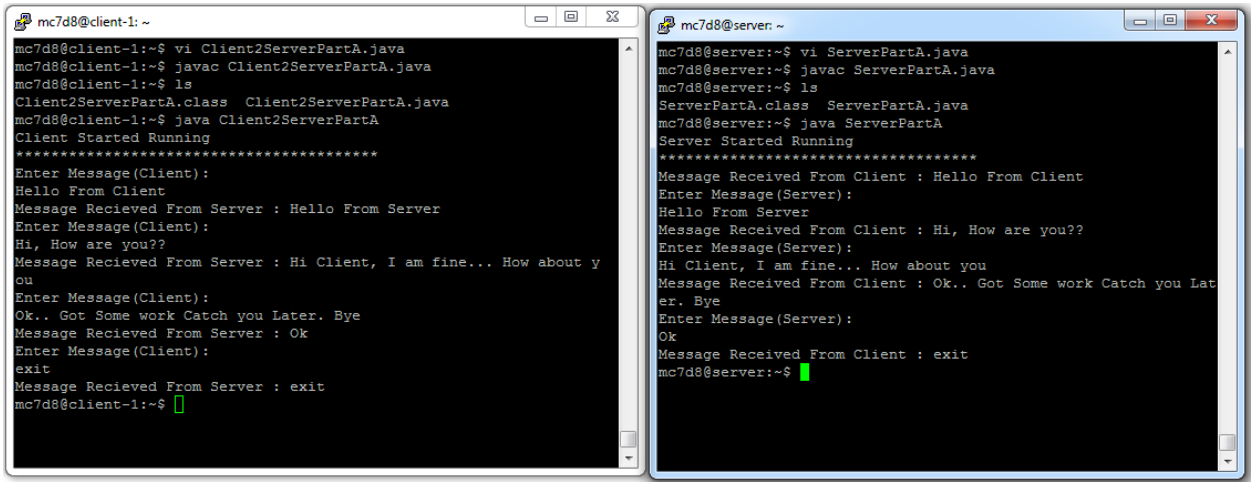
When ServerPartA java file is executed the Server, connection is started similarly Client2ServerPartA is executed to start Client. When Client Enters Message the message is sent to Server and the message entered in Server is sent back to Client as shown below. Finally, when Client exits, Server also exits from Chat.



Client and Server Connection Started



Chat Communication between Client and Server



Client and Server exits from Chat

3.2 Question-2

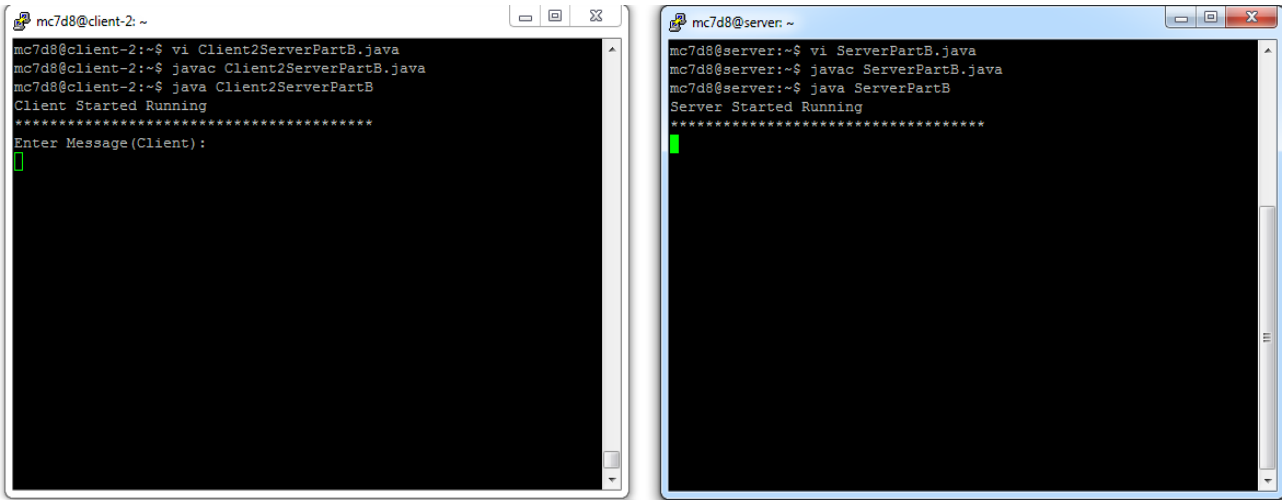
A server now remains 'open' for additional connection once a client quits. The server can handle at most one connection at a time.

Solution:

In this query connection is built between Client and Server, messages are sent from Client to Server and vice-versa. When one of the Client terminates its connection with the Server, the Server waits for the connection from the Clients remaining. At a given time, only one of the Clients can communicate with the Server.

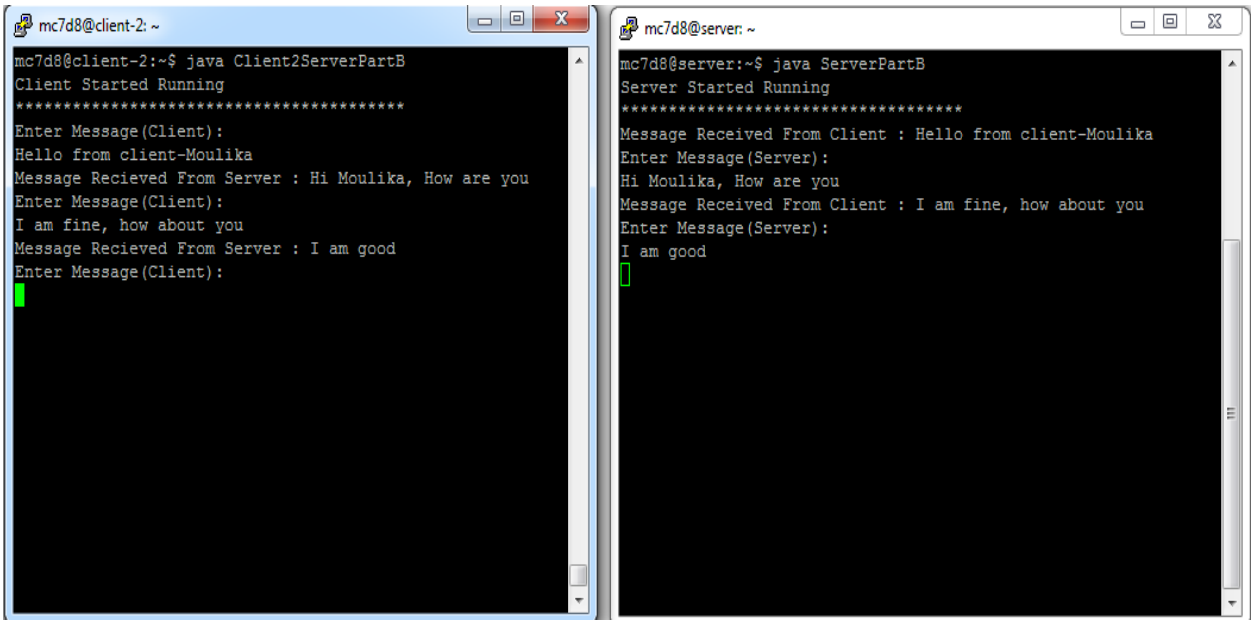
Java program is written for each Client **(Client2ServerPartB.java)** and Server **(ServerPartB.java)** through which messages are received and sent back. Once the java files are compiled and the class file is executed, Client and Server connections are started as shown below. Once chat between Client and Server is completed, Client sends 'exit' which ends Client connection, but Server connection remains and waits for another Client to join.

1. When ServerPartB java file is executed in Putty the Server, connection is started, similarly Client2ServerPartB is executed to start Client.



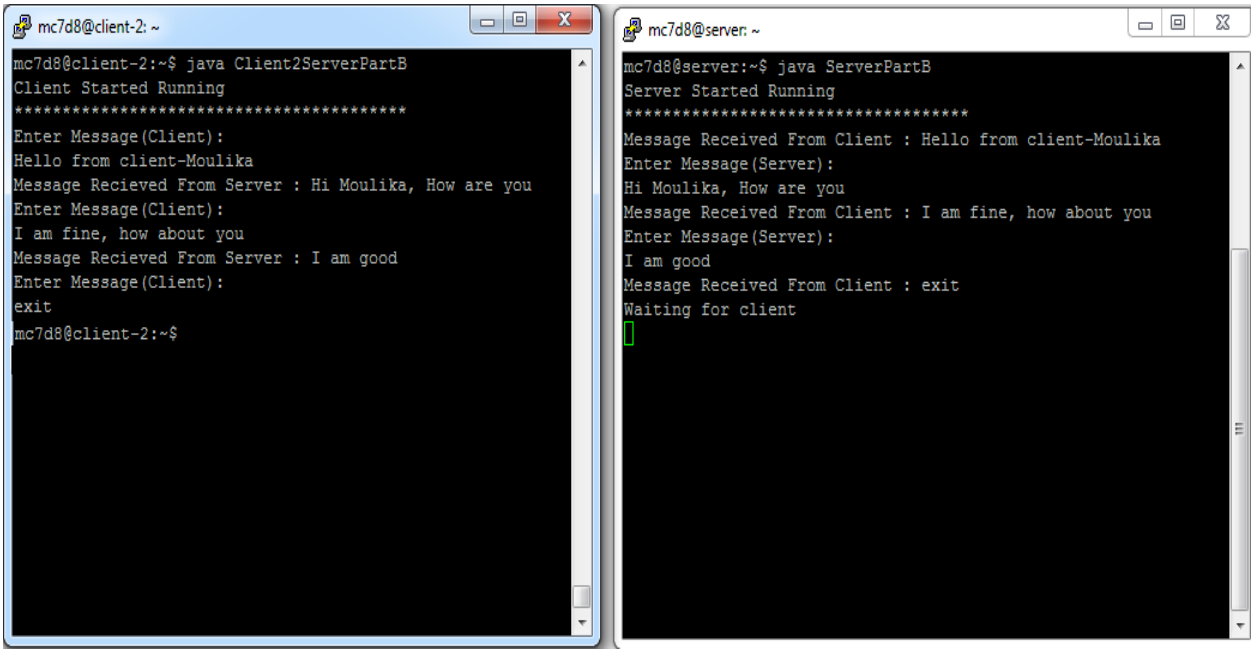
Client and Server Connection Started

2. When Client Enters Message the message is sent to Server and the message entered in Server is sent back to Client as shown below.



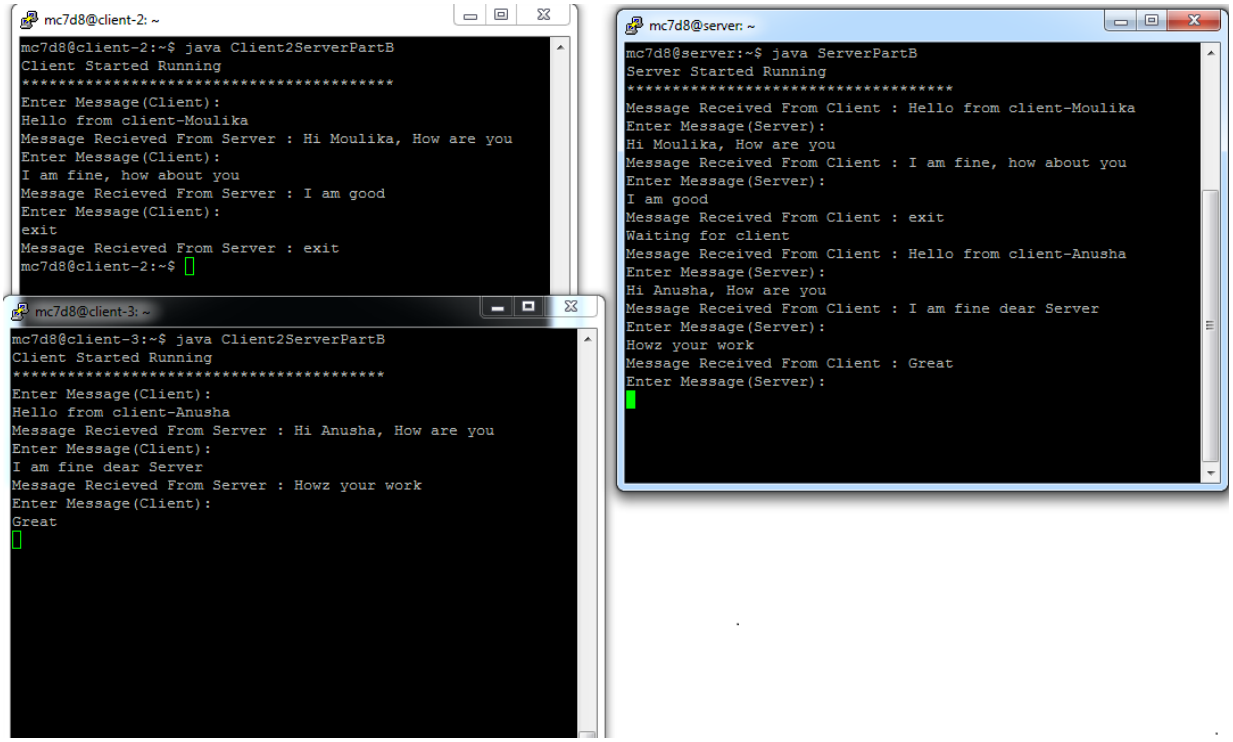
Chat Communication between Client and Server

3. Finally, when Client exits, Server Waits for another Client to join and start the communication

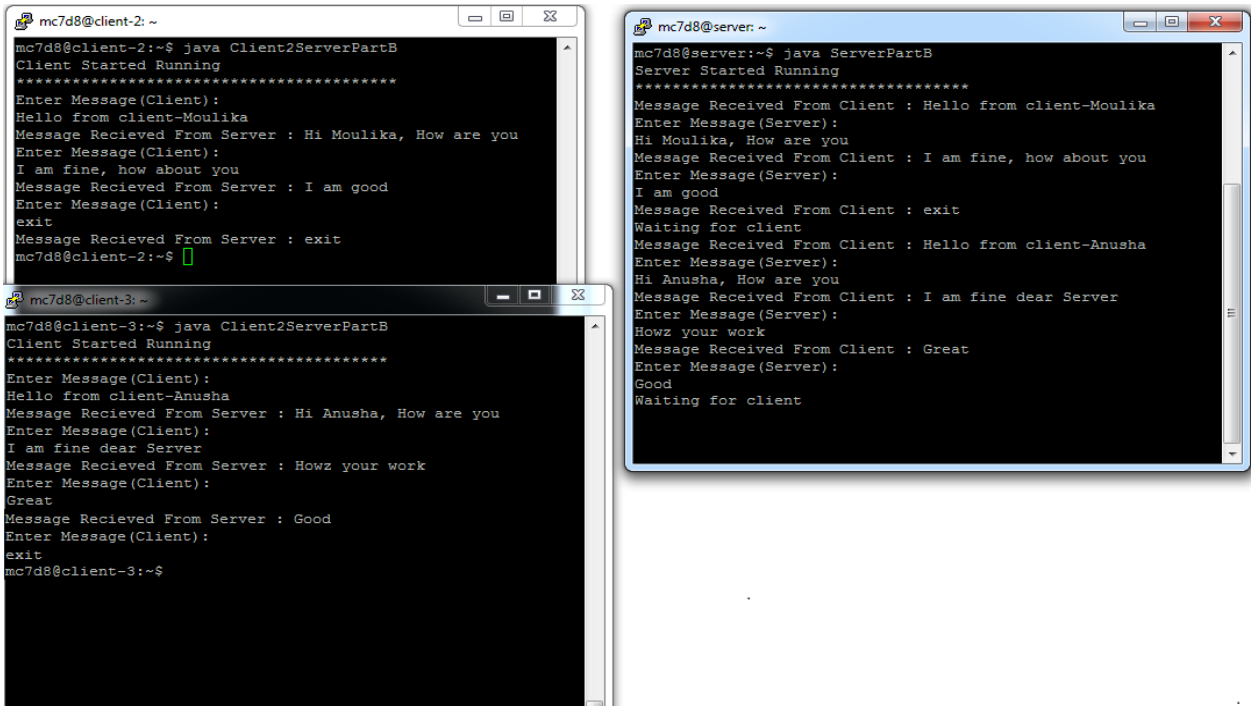


Client exists but Server waits for Client

4. When Client-2 connection is started, it starts communication with Server



Client 2 Enters Chat and communicate with Server



Client 2 exits from Chat

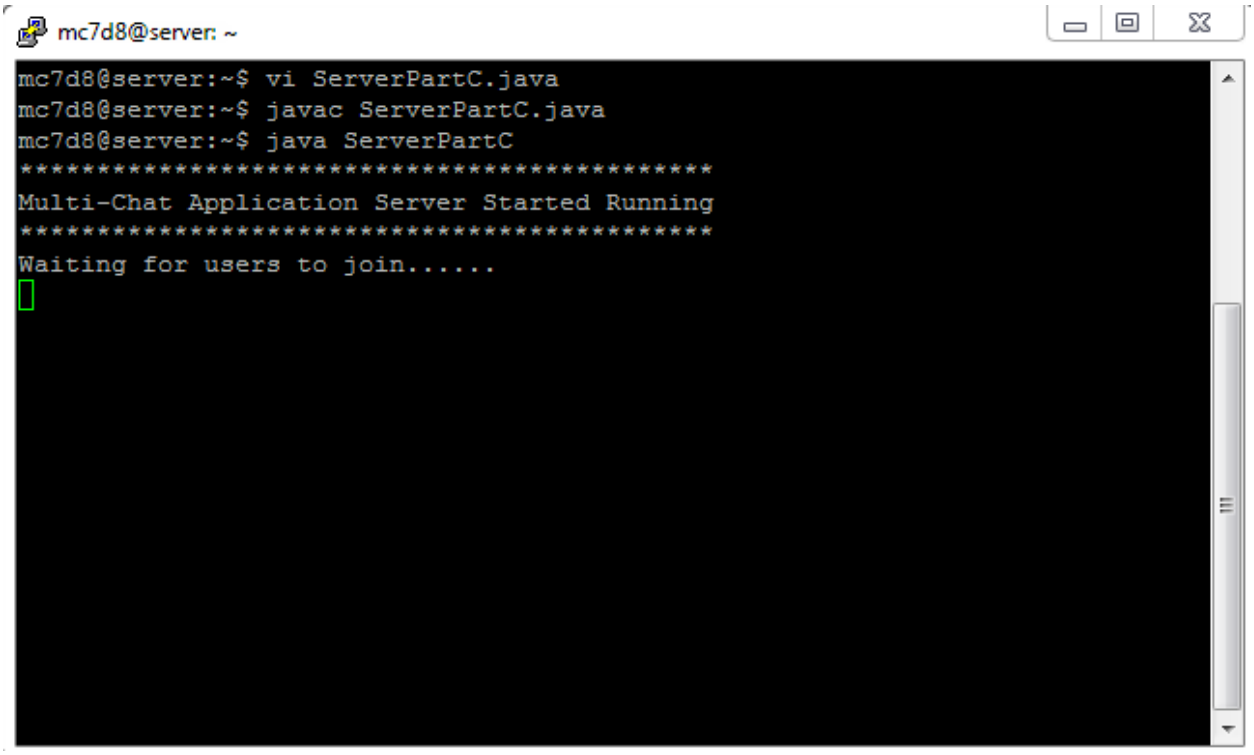
3.3 Question-3

A server now can handle multiple clients at the same time. The output from all the connected clients will appear on the server's screen.

Solution:

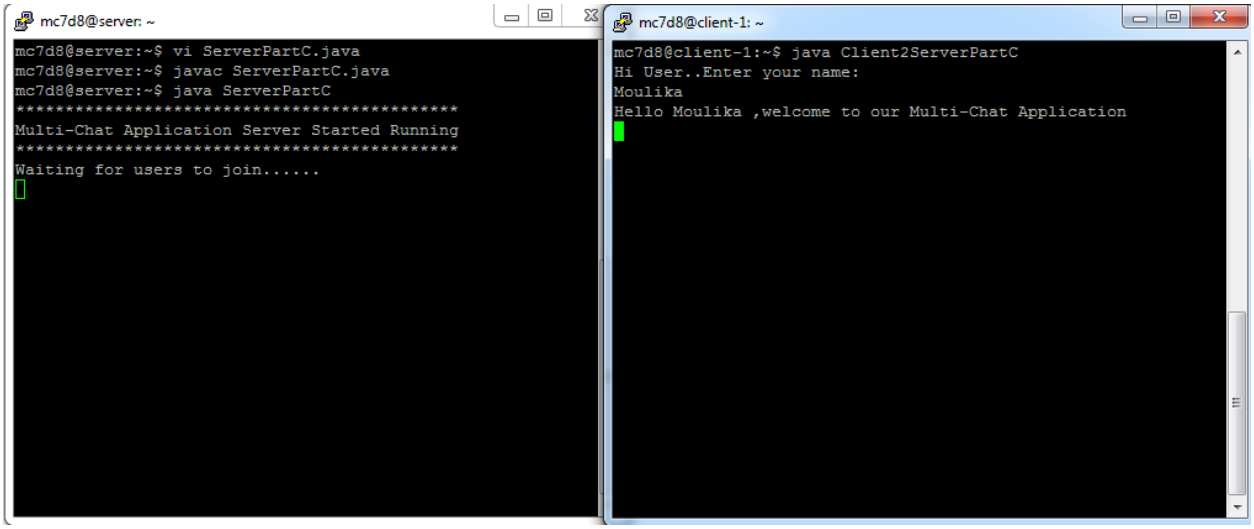
In this query connection is built between Client and Server, messages are sent from Client to Server and vice-versa**.** Java files are created for each of the Clients and Server through which messages are received and sent back. In this query, we illustrate that the Server can handle any number of Clients at the same time. As the Clients send the messages to the Server, they appear on the display screen of the Server.

1. Initially ServerPartC java file is executed, which waits for clients to join

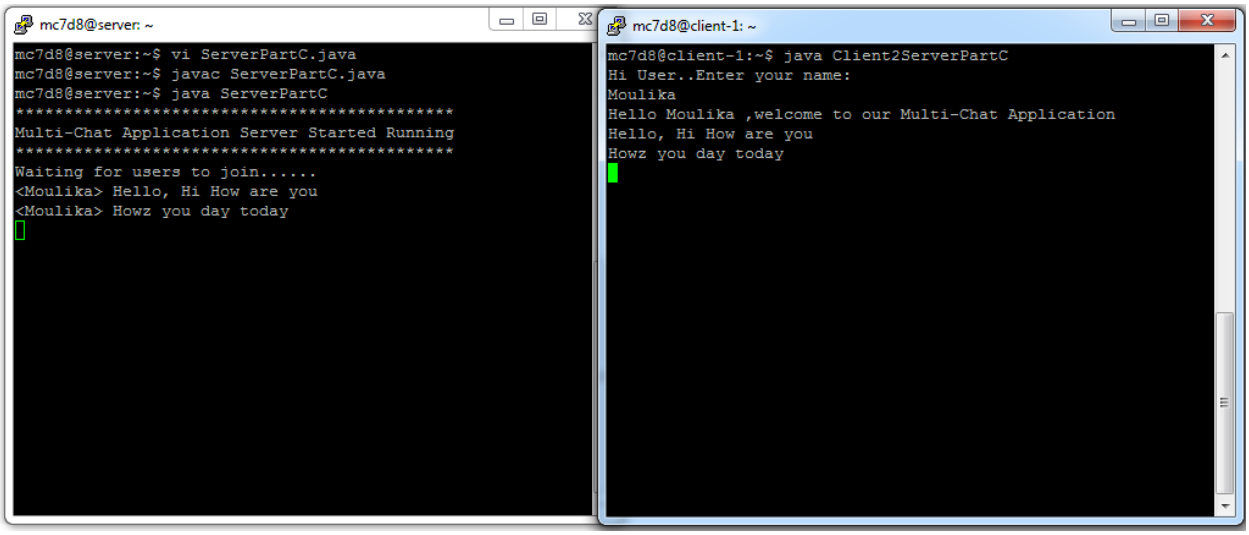


Server Starts and waits for clients

2. Next Client-1 is started, where the name of Client is entered, so that it will be easy to differentiate between clients. Here the name is entered as 'Moulika' and the chat entered in Client-1 is displayed on Server screen with Name of Client <Moulika>

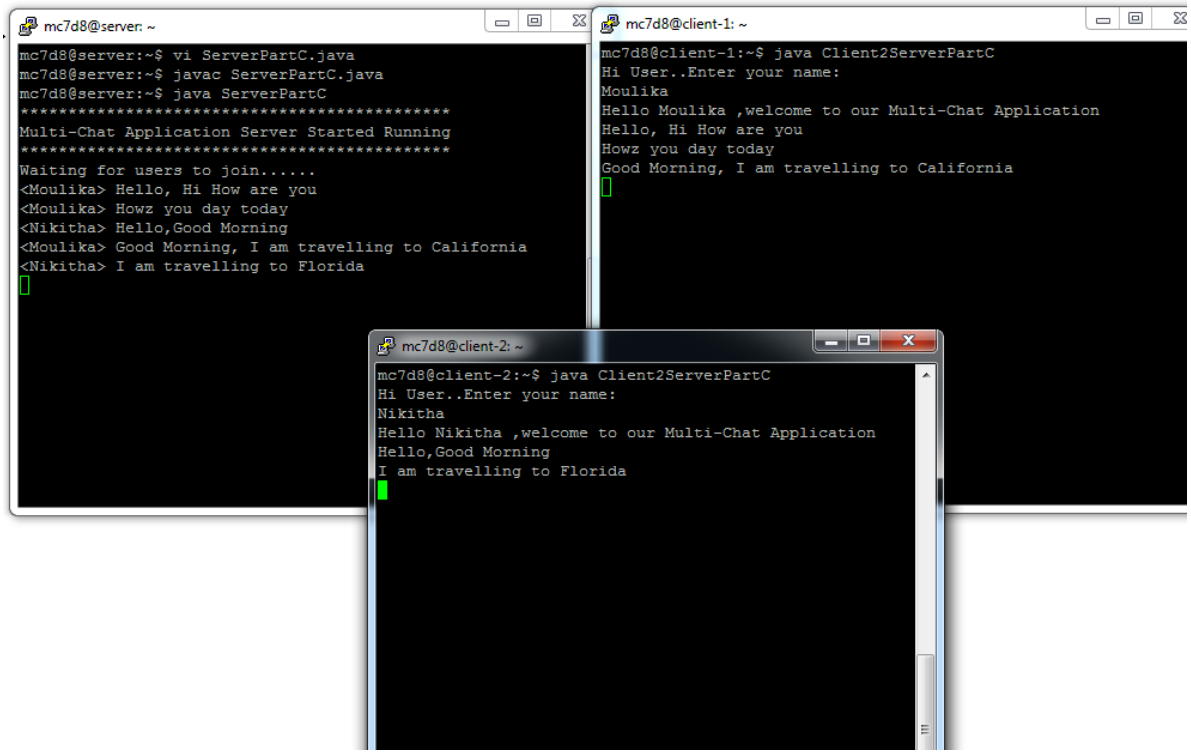


Client-1 is added into Chat



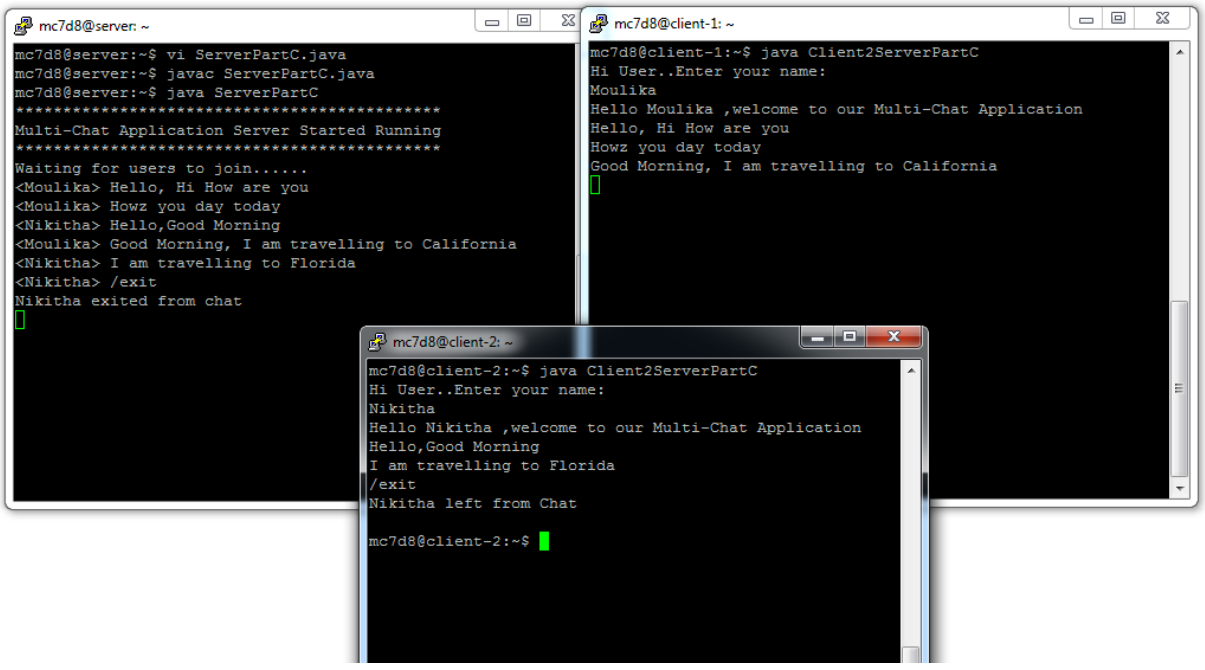
Client-1 Started Communicating with Server

3. When Client-2 in similar way as Client-1 is added. Here Client-2 name is entered is Nikitha and the chat entered in Client-1 and Client-2 are displayed on Server.



Client-2 is started and chat is displayed on Server

4. Once the client enters /exit then the client will be exited from Chat and the intimation is given to Server. Here Client-2 has exited from chat and the response is displayed on Server



Client-2 exited from Chat

3.4 Question-4

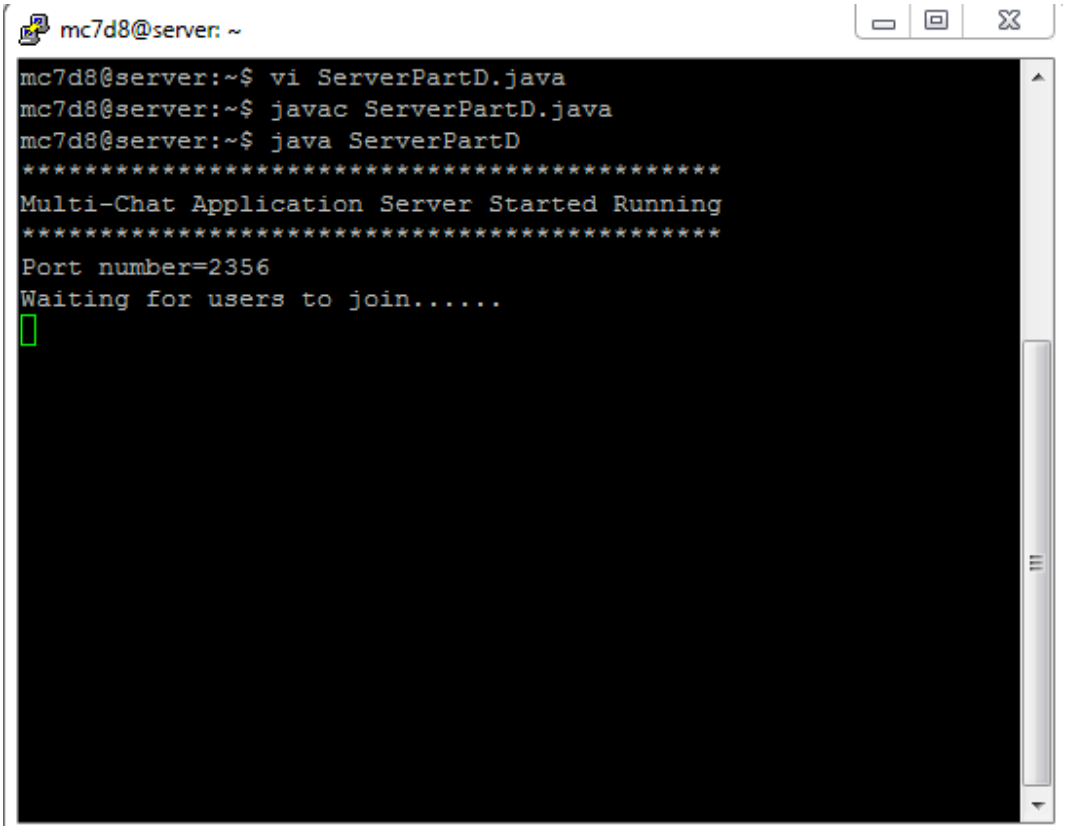
A server now echoes all the text received from any of the connected clients to all.

Solution:

Java files are created for each of the Clients and Server through which messages are received and sent back. In this query, we show that when any of the Clients sends its message to the Server, it broadcasts the message to the remaining Clients with the name of the Client specified.

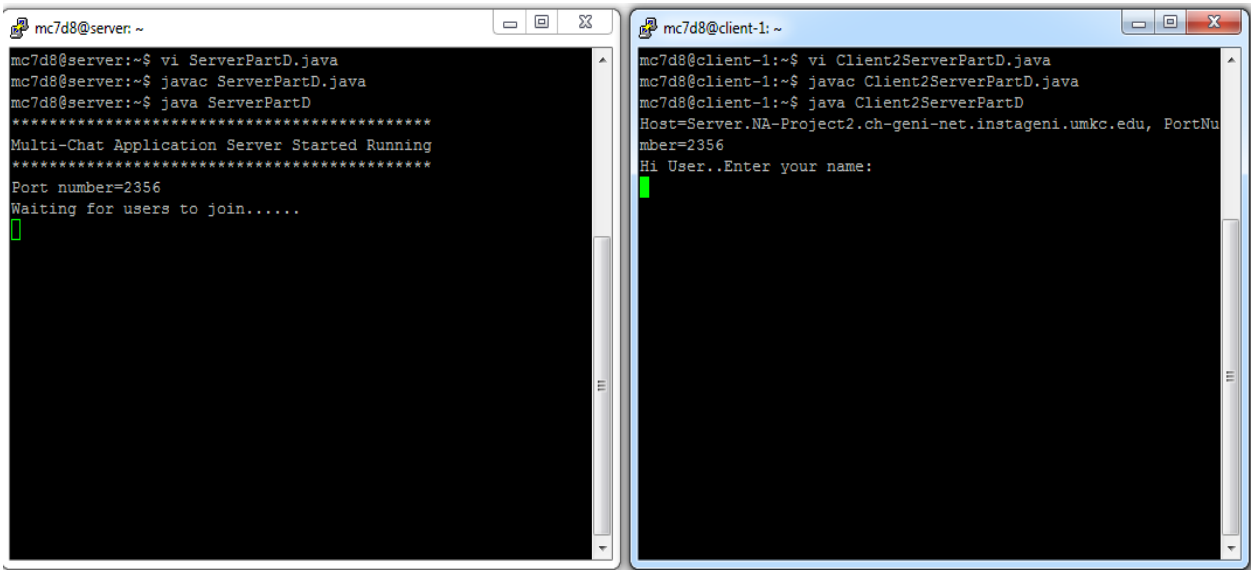
Here we are connecting to 4 Clients, whenever new Client is started to Chat Application the intimation is given to all other Clients. Also, when Client is left from Chat the exit message is sent to Server. The communication between client and server is echoed to all other servers. For example, Client-1 sends message 'hi', this message will be displayed on Clien-2, Client-3, Client-4 along with Server.

1. Initially ServerPartD java file is executed, which waits for clients to join

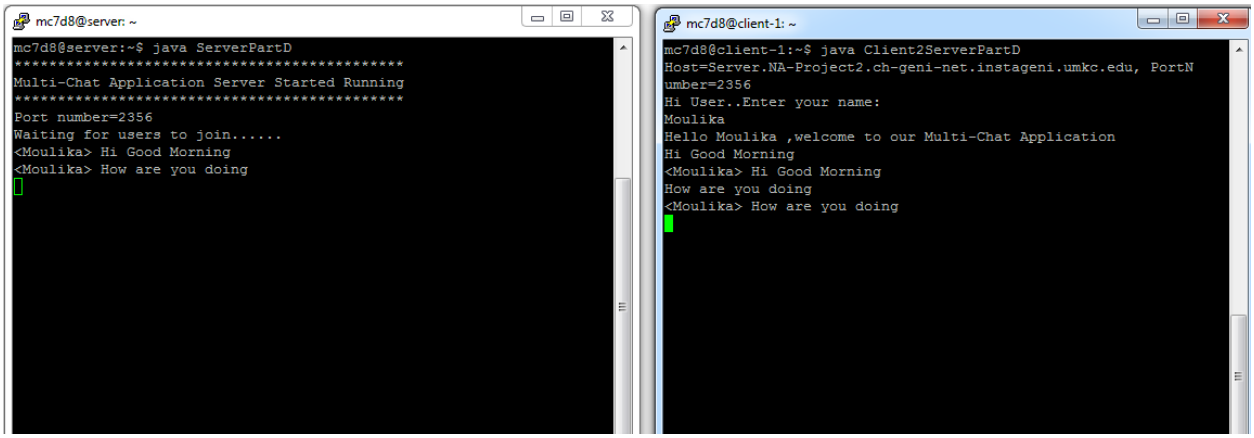


Server Starts and waits for clients

2. Next Client-1 is started, where the name of Client is entered, so that it will be easy to differentiate between clients. Here the name is entered as ‘Moulika’ and the chat entered in Client-1 is displayed on Server screen with Name of Client <Moulika>

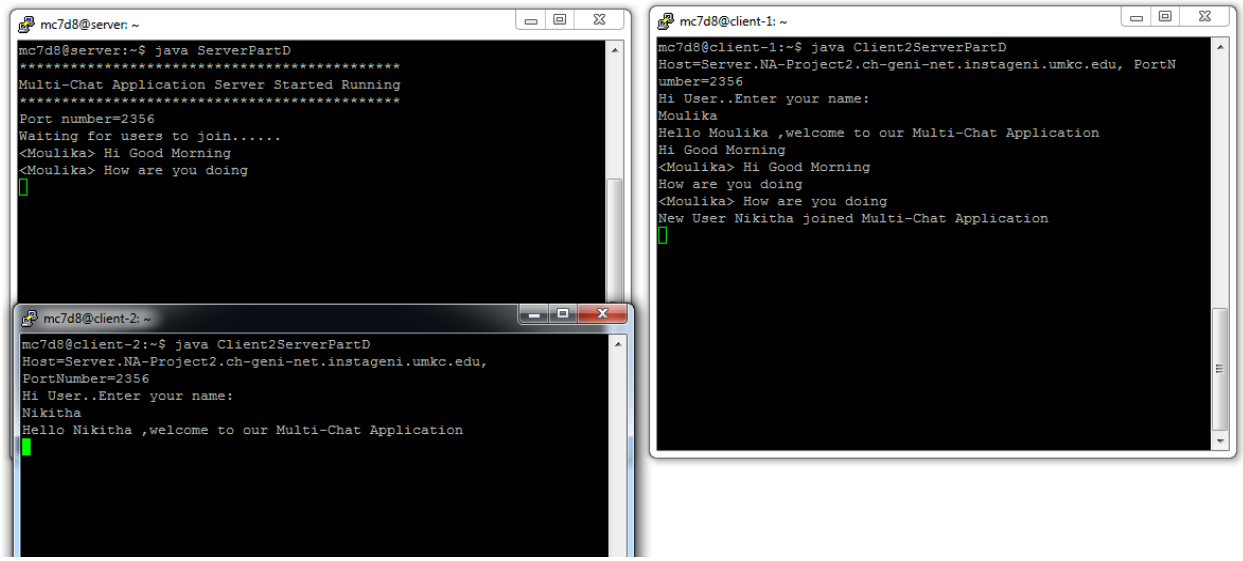


Client-1 is added into Chat

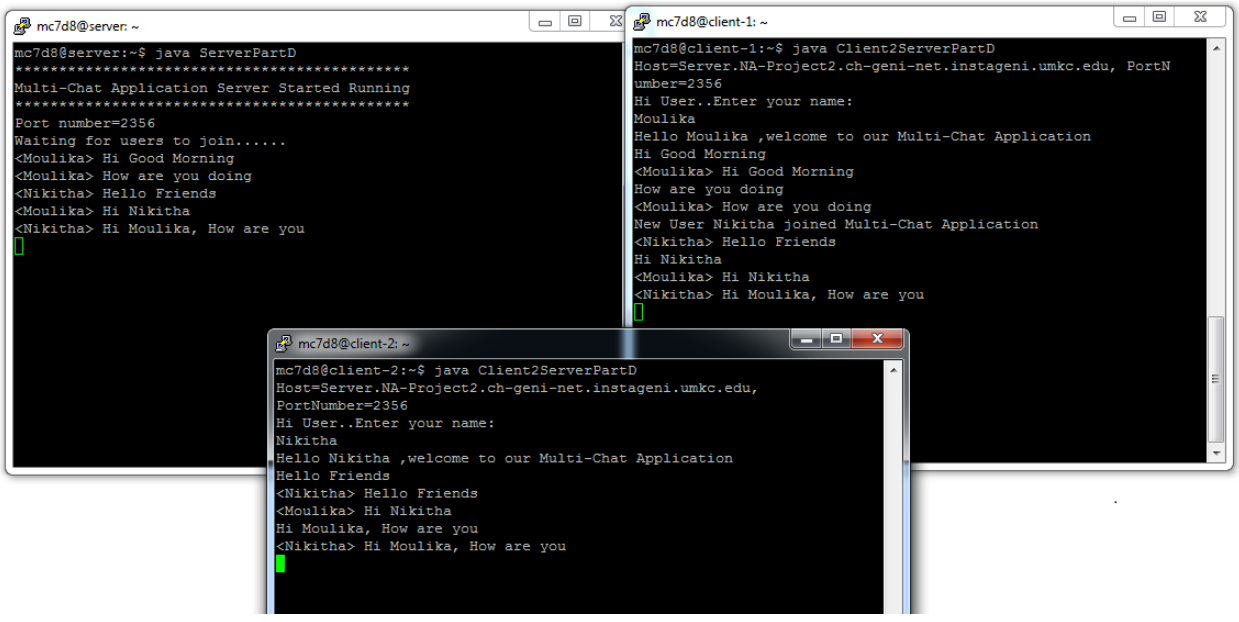


Client-1 Started Communicating with Server

3. When Client-2 is added then the New user added notification is sent to Client-1, so that they can communicate each other. Here Client-2 name is entered is Nikitha and the chat communication between Client-1 and Client-2 is displayed on Server.

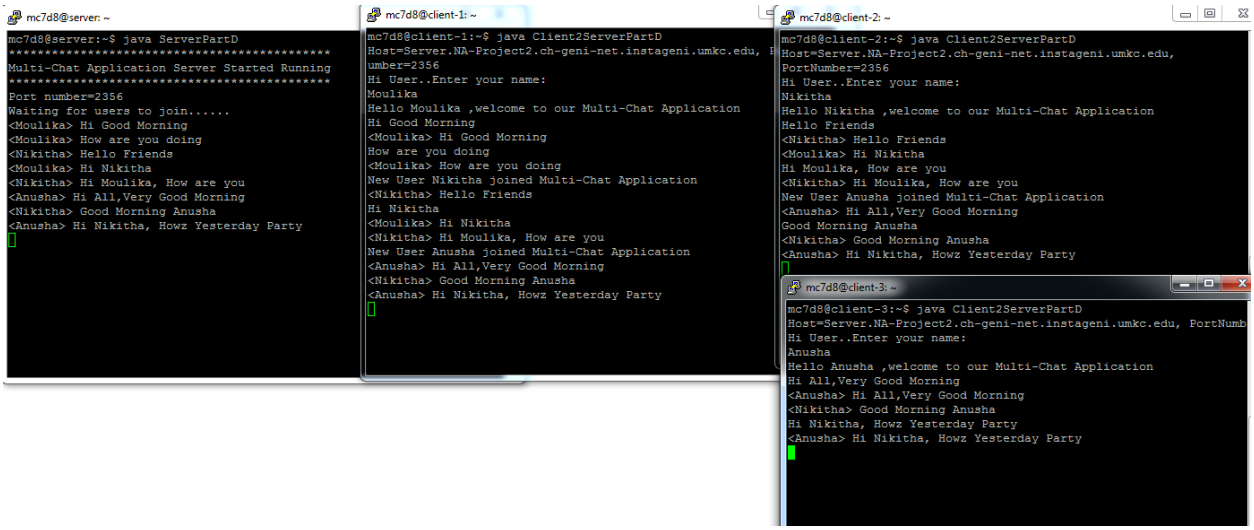


Client-2 is added and intimation sent to Client-1

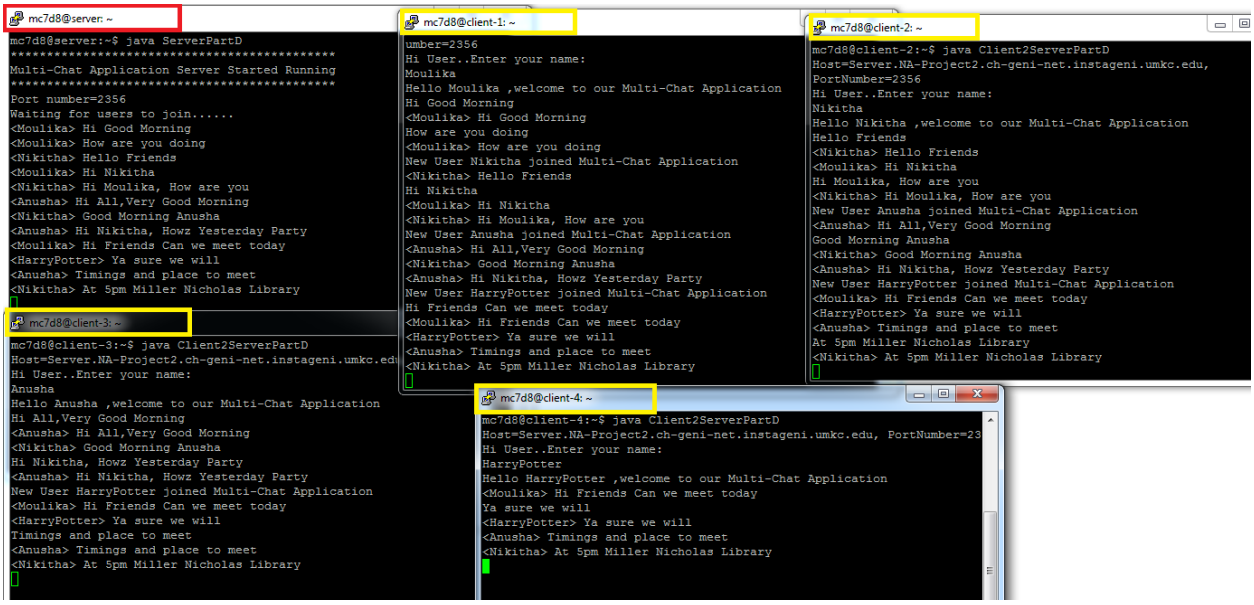


Client-1 and Client-1 are communicating with each other

4. Similarly, other two clients Client-3 and Client-4 are also added into Chat Application. So, four clients group chat and the same is displayed on Server.

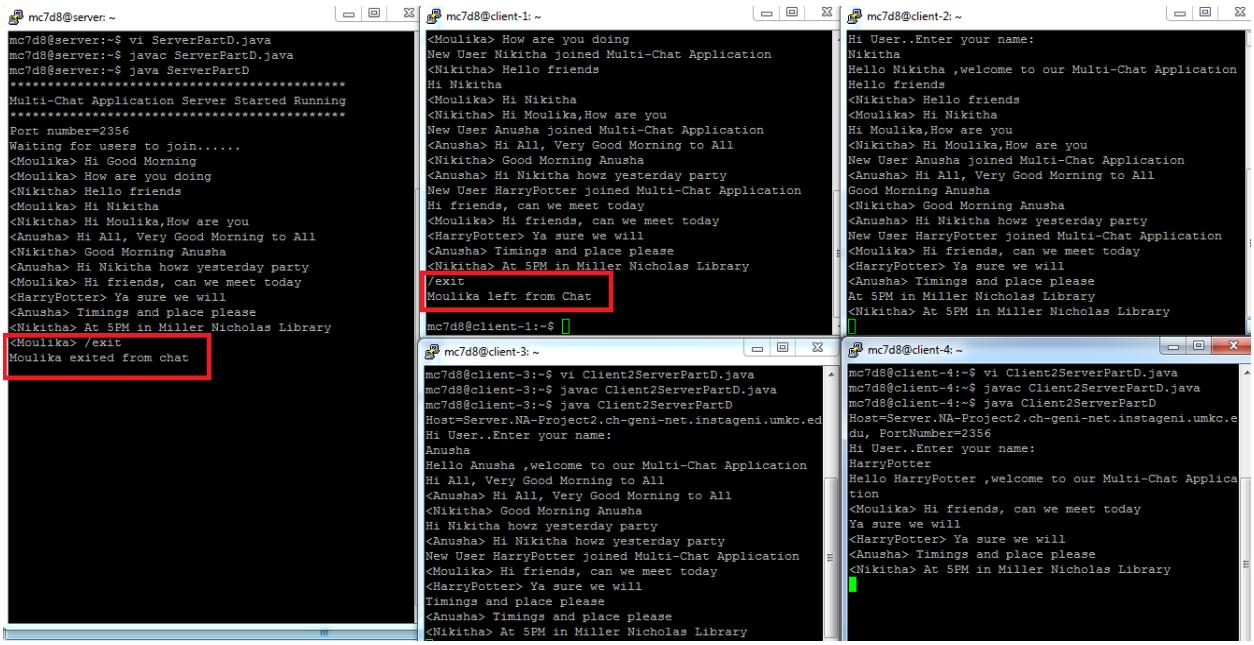


Client-3 is added and started communicating with other clients



Client-4 is added and group chat between all client

5. Once the client enters /exit then the client will be exited from Chat and the intimation is given to Server. Here Client-1 has exited from chat and the response is displayed on Server



Client-1 Exited from Chat

4. References

http://portal.geni.net/secure/dashboard.php

http://stackoverflow.com/

http://www.putty.org/