Bill Splitter

DOCUMENTATION

Rakesh Prasad | 18mcmt14
Chandrashekar TR | 18mcmt15

Introduction:

The purpose of this application is to provide an easy and fair way to resolve the problem of splitting a bill or several bills between a group of people. All the concerned parties would connect to the server using their individual PCs and enter all the places where and how much, they spent the money which can be seen by all others in their PCs. Once a person is done sending all the details of his expenditure for the group, he can ask the server to split the bill, to inform server that all his expenditures are done. This way once all the connected parties ask the server to split the bill the server does the required calculations and sends every person how much money he should pay/get from which person and all confusion, discussions and hassle of calculation can be avoided.

System Requirements:

- Java SE 6 or Higher (For supporting JavaFX) in client side PCs

- A Server PC with static IP (for usage through internet) or All clients inside same Lan as the PC running the server program.

- All concerned parties need to have their own PCs as the bill will be split only between the persons connected to the server.

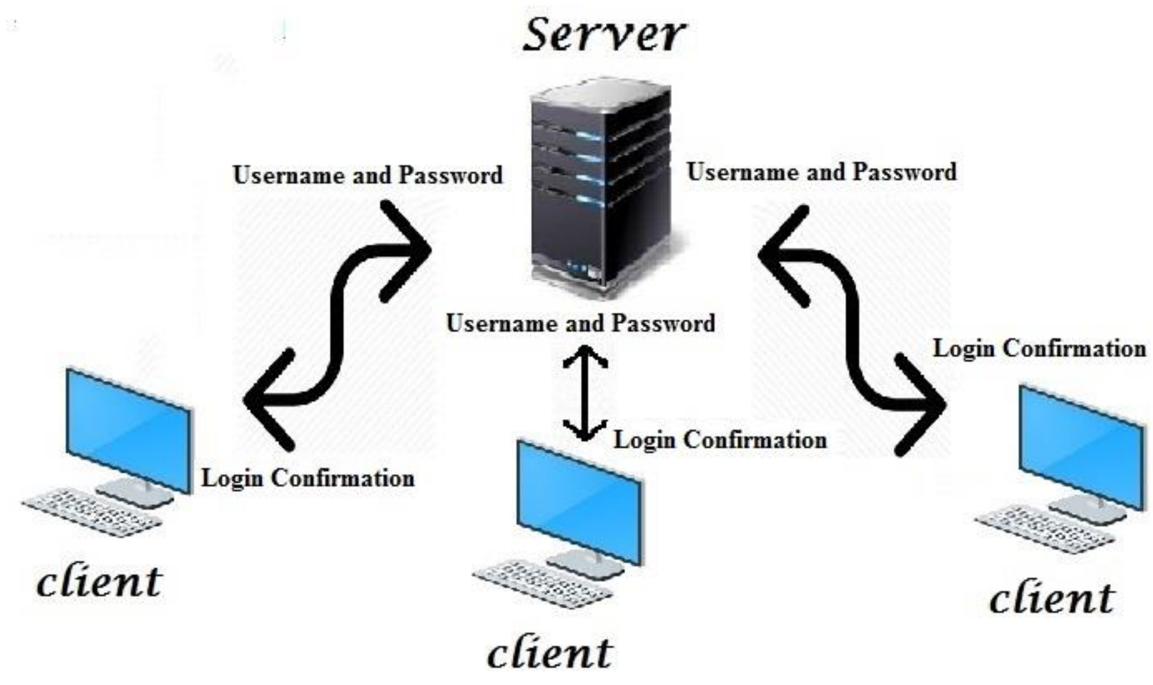
Protocols Used:

The two main communication protocols that we used in this project are:

- Point to Point UDP protocol between Server and client at the beginning and end of the program.
- Multicast UDP protocol between all the active clients and the server in most of the middle part of the application logic.

The first step, that all the clients need to do, is connect to the server within a specified time to connect to the group using their names and a prefixed key word or password.

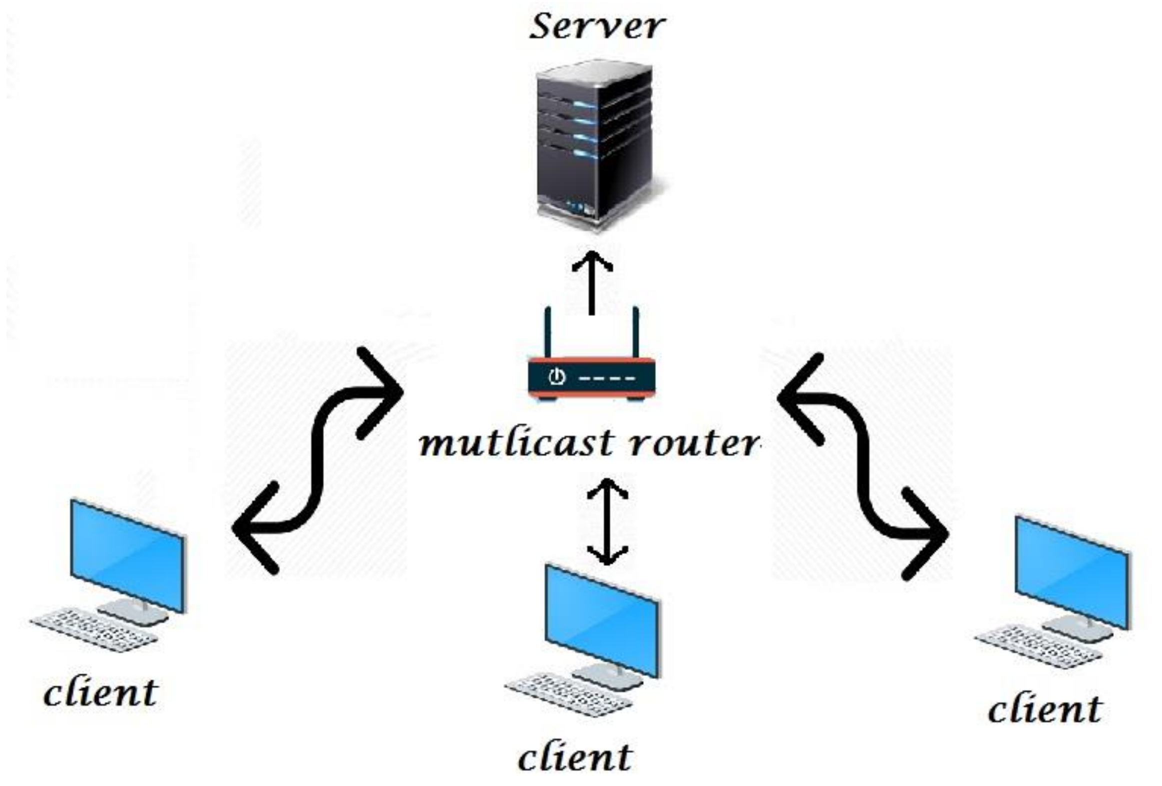
To connect to the server and to receive successful login confirmation, we use the point to point UDP where all the client logs into the server individually and then the server sends them the login confirmation, once the prefixed password is checked and found to be in order.



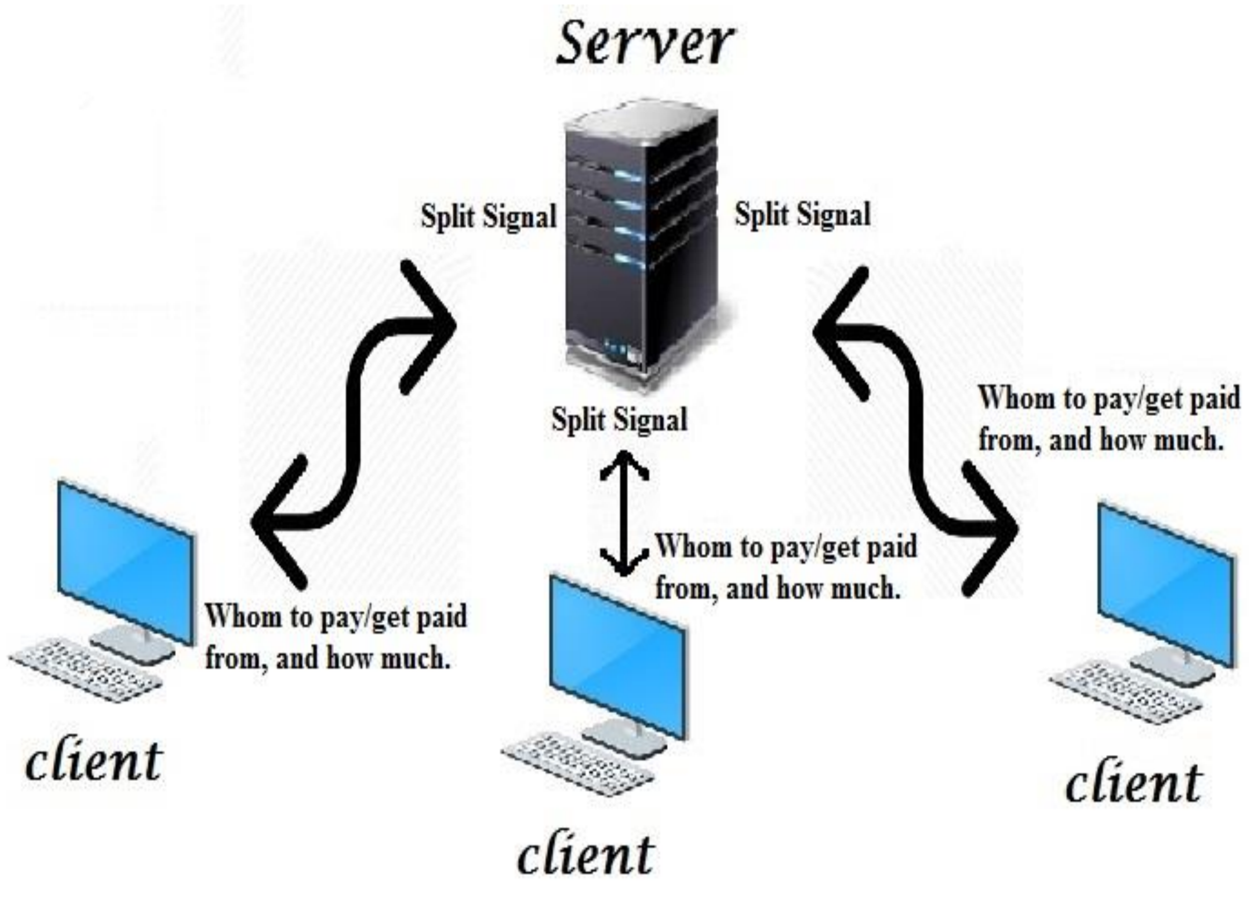
After every single clients receives the login confirmation and the specified time limit ends, all the clients and the server joins the multicast network to listen to all the messages sent between the clients and record the data of who spent how much to calculate the bill amount each of the client needs to pay at the end. This Server acts as snoop and just listens the messages being sent on this multicast network but never sends its own message here.

All the clients send their own expenditure to the multicast network and each of them receives the message so that they can themselves see and why they need to pay and how much.

This is the major part of the program Logic and the communications during this time occurs using Multicast UDP. For proper facilitation of this part the network requires a router which supports Multicast functionality (This is trivial as almost all the routers these days have multicast functionality inbuilt.)



The final Part of the program Logic happens when a client has finished sending all his expenditure and is ready to submit his data for calculation of the bill split. The client would send the message to split the bill to the server directly using point to point UDP communication and when all of the active clients sends their respective split message, the server does the calculation and sends private message to all the people how much their share of the bill is and to whom they need to pay it.


Server
Split Signal
Split Signal
Split Signal
Whom to pay/get paid from, and how much.
Whom to pay/get paid from, and how much.
Whom to pay/get paid from, and how much.
client
client
client

Classes and their description:

➢ Client Side :-

- BSclient – This is the main () class of the client side. It initiates the GUI of the client side by using the launch () function. It also calls the SendRecv.InitMulticast () function which initiates the multicast socket.

- SendRecv – This class is the contains all the functions used for communication between the client and server(Unicast) and all other clients (Multicast). It has five functions,

  ➲ InitMulticast (): It initiates the multicast socket.
  ➲ sendtogroup (expense ob): It sends the object ob to the multicast group.
  ➲ recievemulticast (): It receives multicast messages coming to the multicast socket.
  ➲ connectToServer (String user, String pass): It is the starting function of the communication. It sends the clients username and group's password to the server and waits for its acknowledgement. Once the credentials are verified the client is allowed to connect to the multicast group and communicate with all other active clients.
  ➲ recieveresults (): It is the last function of the communication. After split bill button is pressed, this function is called to receive the results from the server.

- FXMLDocumentController – This class has all the functions which controls the JavaFX GUI components. The initialize function is called on creation of the FX scene. All other functions are called as and when their respective events occur.

- Readhread – Runnable class which keeps the receive multicast function in separate thread so that the whole program is not stalled while waiting for multicast messages.
- expense – It's the class which gets serialized to send the expense details in the multicast group.

➢ Server Side :-

- SendRcv – It is the main class of the server side. It calls all the other threads required for the program to run. The main function at first waits for the clients (the ones with valid group password only) to connect to it and once all the clients are connected it will join the multicast group itself and starts the Thread responsible for receiving the multicast messages.
- ReadThread – Runnable class which keeps the receive multicast function in separate thread so that the whole program is not stalled while waiting for multicast messages.
- split – After all clients send the split message the server calls this class to split the bill. Once the bill is split this function sends the all the connected clients their respective part of the bill using unicast communication.
- expense – It's the class which is needed deserialize the expense details received from the multicast group.

Future Expansions:

- Currently due to the time constraints of this project we have only implemented the server capable of just handling one group at a time. In future we can extend this to create server capable of handling multiple groups at a time.
- We might also add more functionalities to this project like allowing users the option to reject other users claim that they spent money on something they didn't and get group consensus to resolve the dispute that arises during the split if any.