Messenger

In this task, you need to build a client-server based Messaging platform. Using this platform, 2 friends should be able to communicate with each other over the network, even though they do not know each other's ip address. Each user is identified by single word name (like Saifur, Aashik etc.).

The underlying transport for the system should be UDP. You must support fully asynchronous send/receive behaviour. There should be 2 applications (i.e. 2 classes with main() function). One is the Server, the other one is the Client.

Message Format

The content of each UDP message should be as follows:

Format	Example
Via: <name of server> To: <name of intended recipient> From: <name of sender> Body: <MessageBody>	Via: CSE108_NetworkingAssignmentServer To: Aashik From: Saifur Body: Hi how are you?

The Server

The server should be started with 1 command line argument. The argument gives the server a name as follows: Java Server <server name>. In the above example, the server should have been started with:

java Server CSE108_NetworkingAssignmentServer

Once started, the server should listen for UDP packets on port 5050. When server receives a message, it first checks if its name matches the name in the "Via" field. If not, then the message is dropped. A warning message is shown in the Server console window:

"Warning: Server name mismatch. Message dropped."

If the above check passes, then it looks at the next line to identify the recipient. If server knows the ip and port for the recipient, it forwards the message to that intended user using another UDP message. If the server doesn't know the ip and port, then it prints a warning message:

"Warning: Unknown recipient. Message dropped."

The Client

The client should be started with 3 command line arguments as follows:

Java Client <user name> <listening port> <server ip> <server name>

When the client gets started, it should create a separate thread to start receiving UDP messages through the "listening port" that was specified in the command line. Observe that, while the client knows the port number of server (predefined: 5050), the server is not aware of the port number where the client is receiving messages. Therefore, the client must send the server a message letting it know the port number. See the table below for the message that should be sent to server, when the client is started with the following command line:

Java Client Saifur 12345 127.0.0.1 CSE108_NetworkingAssignmentServer

Format	Example
Via: <server name> To: <server name> From: <user name> Port: <listening port>	Via: CSE108_NetworkingAssignmentServer To: CSE108_NetworkingAssignmentServer From: Saifur Port: 12345

As such, the server can now store the ip and port information of Saifur. (Note that if the server name is incorrect, then server drops this message but no information is sent back to the client). When a message needs to be forwarded to the user, server uses this ip/port information to send a UDP message. Notice that, the above message only contains the port number, not the ip address of user. This is because the ip address is already available through getAddress() method call on the DatagramPacket that contained this message.

When a client receives a message, it first matches the "Via" field with the server name that was used to launch the client. If there is a mismatch, the message is silently dropped. Then it checks the "To" field. If the value doesn't match the user, then the message is silently dropped. Otherwise, the message is displayed.

When Saifur wants to write to Aashik, he writes in his client console:

Aashik$ Hi how are you

The data that gets transmitted to the server is:

Via: CSE108_NetworkingAssignmentServer

To: Aashik

From: Saifur

Body: Hi how are you?

When Aashik's machine receives this message, it shows in console:

```
Saifur says: Hi how are you?
```

Be Careful

While we have clearly defined the formatting of our messages, it is possible to receive UDP packets that are not conforming to our format. This is because any other application could have sent a UDP packet to our port. Therefore, we must be careful during our processing and handle exceptions as needed. (e.g. you may not find the Via field at all in the message etc. You should handle it properly. In case of the server, you should write a warning message in the console. In the client you should silently drop the message.