CS-421 Programming Assignment-1

ORHAN TAHİR YAVAŞCAN

21301295

ERKAN ÖNAL

21302017

Description of our protocol for peer-to-peer communication between the clients

In listen part, we were just sending HTTP POST request and then wait for messages coming from other users so in this part, there is no specific thing about our protocol. Considering listen side as server side of peer to peer communication, almost all functionality of our protocol lies in the send part of the assignment. In send part, we firstly get the userlist.txt file before doing any operations. Then parse this file to get the usernames, their IP numbers and their Port numbers. Then, we wait for user to type specific commands as listed in the assignment. As long as he does not type exit, we continue to wait for his commands.

In this protocol, listener side is understood who sent the message in a way that we are adding the username of the message sender to the string message we want to send before converting into byte array like username + ":" + msg_to_sent where msg_to_sent is the actual message user typed to send. This way is same in all other commands.

As UDP is used in this peer-to-peer communication, in the server side of the peer-to-peer, there should be a predetermined size of the byte array to capture the sent data from client side. We have made it 1024. If client's message was lower than the 1024 bytes, rest of the array was automatically filled with emty spaces so this is an unwanted behavior as output message's line was sliding. To hinder this, we, initially, get the data with 1024 bytes long byte array after then we found the size of the actual packet with method receive packet.getData() and then copy the packet.getLength() bytes of data to another array which we called realData.

Another specific information about our protocol is that we did not allow :, @, and space charecters in username and tried to cover possible wrong command types by user. For example, if he uses like a statement unicastahmet "meraba". Error message is given by our protocol because unicast statement and user to send the message should be divided by space.

How can we improve the protocol?

There are many ways to improve our protocol. For example, one of them is to be able to send and listen at the same time. In our protocol, while one is listenning other is sending. To send a message, listen side should stop the program and open it with send mode. Reverse is same for send side. Therefore; It would be much better for this protocol if both sides are able to listen and send at the same time. In addition to that, we used Udp in this peer-to-peer side so this is an unreliable protocol where no guarantee to send our message correctly. To hinder this case, we can use a reliable protocol like TCP to make sure that message is sent exactly as it is written. Another thing as an improvement to this protocol could be multithreading the program. Considering that this protocol will be used for very

large number of people. Single threaded message protocol could be a slowening factor because considering the command broadcast, you want to send the same message to 10.000 people. If you do that sequentially, you will have significant amount of delay. This is very bad. This problem can be erased by using multithreaded approach by dividing the work to number of threads.