Aquino, Kurt Neil Miss Chesca Laguna

Choy, Matthew Seaver NETWORK – S19

Machine Problem Documentation

Milestone 1: 50%

*What programming concept must you use in order to do this?*

* Since a Client-Server type of network will only provide a one-way relationship, a Peer to Peer connection is the most applicable concept. By threading both the Client and the Server together, the program is able to send various requests (e.g. “POST”, “FOLLOW”, etc.) from our client to their servers, and then receive requests from multiple clients on our server.

*Why is an IP address needed in the command line but not in the network message?*

* An IP address is merely used to determine which socket the program wishes to connect to. It is the identifier to notify the program which sockets it needs to access. Since the intended receivers (i.e. the FOLLOWERS) already know their respective IP addresses, all the program needs to do is to open the connection of the receiver(s), send the message/request protocol, and then close the connection to stop the socket's InputStream and OutputStream.

Milestone 2: 60%

*What sockets programming function helps your program to determine the IP address of the message sender? How does it work?*

* To determine the sender’s IP or host address, we will use the socket methods socket.getInetAddress() and socket.getHostAddress(). socket.getInetAddress() returns an InetAddress object that contains the IP address of the remote machine. InetAddress.getHostAddress() returns a String object with the textual representation of that address. So, to end up with a String the program can print or use for other methods, this is how to do it.

Milestone 3: 70%

*Explain how you implemented following peers and accepting followers, where a mechanism in which the username is associated with the IP address of the peer such that when a FOLLOW request is received, the receiver can know in which IP address to reply to. As well as how the UNFOLLOW command works.*

* For receiving FOLLOWERS, we add them to a pending list which contains both their username as well as the newly created socket with the respective FOLLOWER’s IP address. When it is time to APPROVE the FOLLOW request from the sender, the program first removes their username and socket (with the IP address) from the pending list, and then add them to a list of approved sockets.
* For FOLLOWING peers, we create a new instance of a Client with our respective username as well as a new socket with the IP address of the intended peer to be followed, and then add them to a list of Clients which we are connected to. The program then sends a FOLLOW request to the client, sending the follow protocol: “FOLLOW” <ourUsername> NULL to the OutputStream and then closing the socket after the request has been sent. Before the connection is officially acknowledged, we have to be APPROVED first by the intended peer to be followed.
* When we wish to UNFOLLOW a FOLLOWED peer, we first preview a list of followed and approved Clients and then select which Client we wish to unfollow (for ease of use). The program then sends an UNFOLLOW command to the selected Client. It opens the connection then sends the protocol “UNFOLLOW” writes it to the OutputStream, closes the connection, and the finally closes the socket of the UNFOLLOWED Client itself.

Milestone 4: 75%

*How do you make sure that you are only sending posts to your followers? What is the maximum number of characters you can send in the socket?*

* As mentioned before, the program stores a list of approved Clients which it can connect to anytime when needed. For POSTING a message, the program creates new sockets with the FOLLOWERS’ IP addresses, sends the protocol “POST” <message> EOF <ourProfilePic> EOF to the OutputStream and then closes each socket after the message has been sent. Note that this does not happen all at once, it loops individually for each APPROVED FOLLOWER’s IP Address to open a connection. And as required in the MP Specification only 256 characters can be sent in the socket.

Milestone 5: 80%

*How do you know the username of your followees even when the POST message does not have an IP address or a username?*

* Help ples

Milestone 6: 85%

*The program can accept private message. The username of the sender must be displayed and indicate that it is a private message. How did you implement this?*

* Help again ples

Milestone 7: 90%

*What data type is needed to implement the sending and receiving profile pictures?*

* Since it is mentioned in the MP Specifications that the protocol for sending a profile picture is declared in the “FOLLOW” request function, the program therefore sends the entire String “FOLLOW” <ourUsername> NULL <imgFile> EOF as an array of bytes and then writes it to the OutputStream for the receiver to accept. In the same way, when the program receives FOLLOW requests with profile pictures, it reads the entire message sent as a byte array, containing the protocol function, as well as both of the senders username and profile picture.

Milestone 8: 100%

*How did you implement sending and receiving files from followees?*

* For receiving files, the program applies the same concept as receiving a FOLLOW request with the sender’s profile picture, it reads the entire file sent as an array of bytes and then outputs the file into the source code folder of the program using the function FileOutputStream().
* For sending files, the program first asks the user for the location of the file as well as the file name. The program then temporarily stores the file by using a File() data type. The program then checks whether the file is too big or not; if so, an error message will appear; if not, it then sends the file to a BufferedInputStream but coverting it first using the FileInputStream method first. The file is then concatenated with its file name and the request protocol to the form of “FILE” <filename> <file> EOF into an array of bytes. A new socket will be opened and the array of bytes is then written to the OutputStream. And finally, the socket will be closed after the file has been sent.

Milestone 9: 110%

*How did you implement displaying the profile picture of the peer who is POSTING, sending a FOLLOW request, sending a PM, etc. into the GUI?*

* For every command sent through the Socket, a function in the GUI updates the profile picture displayed according to whoever sent the latest protocol. As the POST, FOLLOW, and PM protocols all include the option of sending an image file along with the message, all the program needs to do is to set the image on the GUI based on the image sent through the socket itself.