Computer Networks and Internet 1

Programming Assignment

1 Noam Rahat 205918360

2 Operation system, code language and version

This implementation is written in Python. The specific version of Python that this implementation is written for is not specified, but it should work with any version of Python that supports the **socket** and **threading** modules.
This implementation is not tied to any specific operating system and should work on any operating system that supports Python and the necessary libraries.

3 Theoretical background

Client-server architecture is a common design pattern for distributed systems, where a client sends requests to a server and the server responds with the requested information or service. In this architecture, the server is responsible for storing and managing data, and the client is responsible for making requests and displaying the results to the user.

In the implementation provided, the server is responsible for managing the group chats and handling client requests. The client is responsible for sending requests to the server and displaying the results to the user. The client and server communicate with each other using sockets, which are a low-level network programming interface that allows programs to send and receive data over a network.

4 Technical Highlights

- Server-side:

server.py contains the code for the server-side of the chat group application. It sets up a server socket and listens for incoming connections from clients. When a client connects, the server creates a new thread to handle the client's requests and continues listening for other incoming connections.
handle_client(): This function is called for each client that connects to the server. It sends the opening message to the client, waits for the client to select an option, and then handles the selected option (either connecting to a group chat, creating a new group chat, or exiting the server).

- Client-side:

The client-side code was included in the **server.py** file as part of the handle_client() function.

To run the client-side of the chat group application, you can simply run the server.py file and connect to the server from a separate terminal window. When prompted, enter your desired action (connect to an existing group chat, create a new group chat, or exit the server). If you choose to connect to an existing group chat or create a new group chat, you will also be prompted to enter your name, group ID, and password.
Implementing a nice GUI (Graphical User Interface) will give you an extra credit of 10 points.

5 explanation about the *socket* handshake – which protocol you used, what are the commands, are they *blocking* commands.

In this implementation, we use the TCP (Transmission Control Protocol) to establish a connection between the server and client. TCP is a reliable, stream-oriented protocol that provides a connection-oriented service for transmitting data between two computers.

To establish a connection using TCP, the server and client follow a three-way handshake process:

The client sends a request to the server to establish a connection (SYN).
The server receives the request and sends a response to the client acknowledging the request (SYN+ACK).
The client receives the acknowledgement and sends a final acknowledgement back to the server (ACK).
Once the three-way handshake is complete, a connection is established between the client and server and data can be exchanged between them.

In this implementation, we use the socket library in Python to create the server and client sockets and perform the socket handshake. We use the bind() and listen() functions on the server socket to bind it to a specific IP address and port and start listening for incoming connections. On the client side, we use the connect() function to send the initial request to the server to establish a connection.

All these functions are blocking functions, which means that they will block the execution of the program until they complete. This means that the program will wait until a connection is established before continuing.

Once a connection is established, we can use the send() and recv() functions to exchange data between the client and server. These functions are also blocking functions, which

means that they will block the execution of the program until they complete. This means that the program will wait until data is received before continuing.

6 illustration pictures (usage examples) for any option in the server.

Option 1:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Loading personal and system profiles took 568ms.
PS C:\Users\noam.rahat\Documents\Noam\Computer networks> python Client1.py
Enter your name: Server
1. Connect to a group chat
2. Create a group chat
3. Exit the server
Enter your option: 1
Enter group id: 1
Enter password: 1234
You are now connected to the group chat
Hi friends
Server: Hi friends

Bar:
Wwlcome new client
```

Option 2:

```
Loading personal and system profiles took 570ms.
PS C:\Users\noam.rahat\Documents\Noam\Computer networks> python Client1.py
Enter your name: Noam
1. Connect to a group chat
2. Create a group chat
3. Exit the server
Enter your option: 2
Enter password: 1234
Your group id is 1
```

Option 3:

```
Loading personal and system profiles took 563ms.
PS C:\Users\noam.rahat\Documents\Noam\Computer networks> python .\Client1.py
Enter your name: Noam
1. Connect to a group chat
2. Create a group chat
3. Exit the server
Enter your option: 3
PS C:\Users\noam.rahat\Documents\Noam\Computer networks>
```