

CSC3002F Networks Assignment 2022 Socket programming project.

STUDENTS: SKSVPX001, MSMJO003, GNY0SC001

Introduction

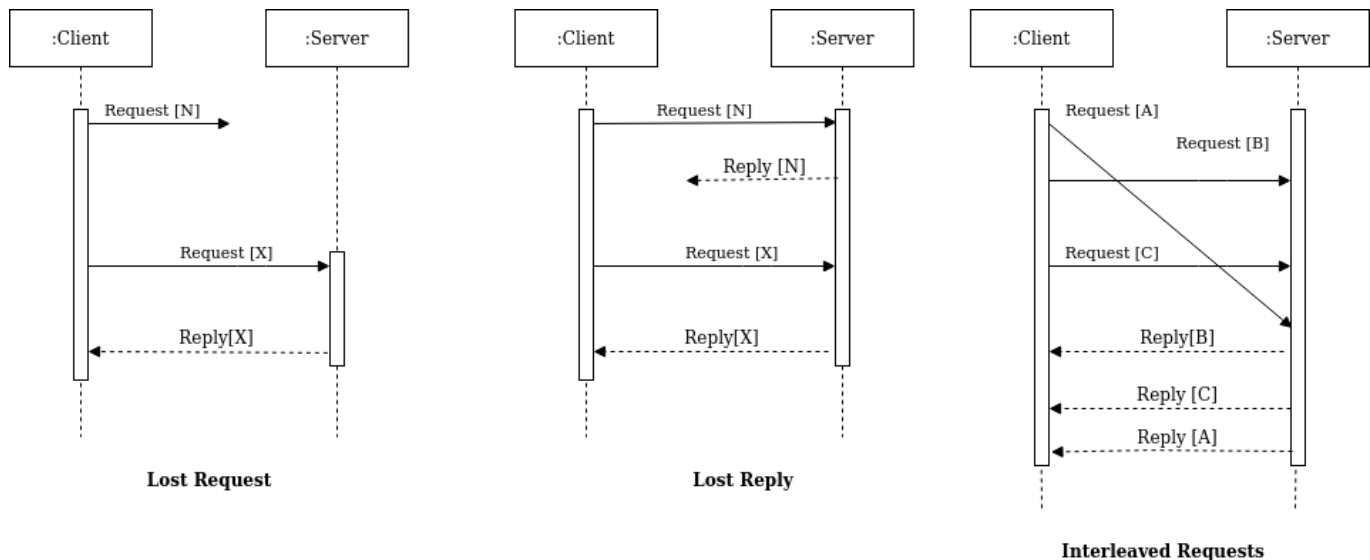
This is a design of an application layer protocol, used to implement a **Python** chat application that uses a User Datagram Protocol (UDP) at the transport layer. The application layer protocol supports the **client-server** architecture, and the implemented server manages the interaction between clients. The chat application allows multiple pairs/groups of users(*clients*) to exchange messages in real-time. As this is a network application, the different clients and the server are able to run on different hosts (computers) over a network. The client interface is a Command Line Interface (CLI).

Protocol

UDP, is a connectionless protocol, meaning it does not reserve a continuous connection a between client and server, a message can be sent to the server without first establishing a connection or an agreement to send data to the server. In fact, any data that gets sent via UDP is sent to the server whether it gets there or not. This alone makes UDP to be faster because it does not perform error checking and reliability in the data that is sends to the server. This means that unstable connections result in a terrible experience for the user (*client*).

Datagram Packets are the capsules that contain the data to be sent over the network, containing receiver IP addresses and port number, as well as containing the sender's IP address and Port number. Datagram Packets are sent through Datagram Sockets which are used to create a network connection between different computers.

Few Sequence Diagram Examples.



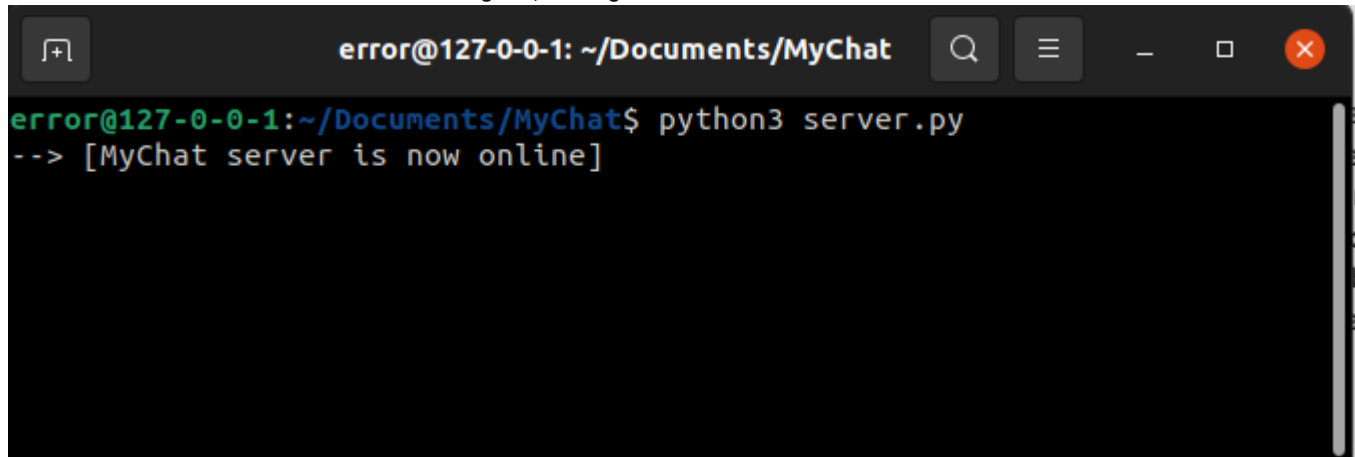
Protocol Design

The protocol design is such that before users (*clients*) can start communication with each other, a connection request to the server has to be established first, meaning that a server should be online and waiting for messages. After successful connection, users connected to server may exchange messages freely while online.

Server

The Server file, contains a Server class that runs and hosts the server. The server class has two sockets, first to receive messages and requests from clients, the second is responsible for sending messages to different clients connected to the Server depending on the message *type*. An unlimited number of clients can connect to the server and all be accepted, provided that they are on the same network. After the server is executed and it is online, it will display a message on the command line screen confirming that it is online, only after it is online it will be able to receive, process, forward or send requests and messages to and from clients. Afterwards, it stores the information of the clients in a `database.txt` file.

Below is a screenshot of the server after being run, waiting for clients to connect.

A screenshot of a terminal window with a dark background. The title bar at the top reads "error@127-0-0-1: ~/Documents/MyChat". The terminal shows the command "python3 server.py" being executed, followed by the output "[MyChat server is now online]". The window has standard Linux window controls (minimize, maximize, close) on the right side.

```
error@127-0-0-1: ~/Documents/MyChat$ python3 server.py
--> [MyChat server is now online]
```

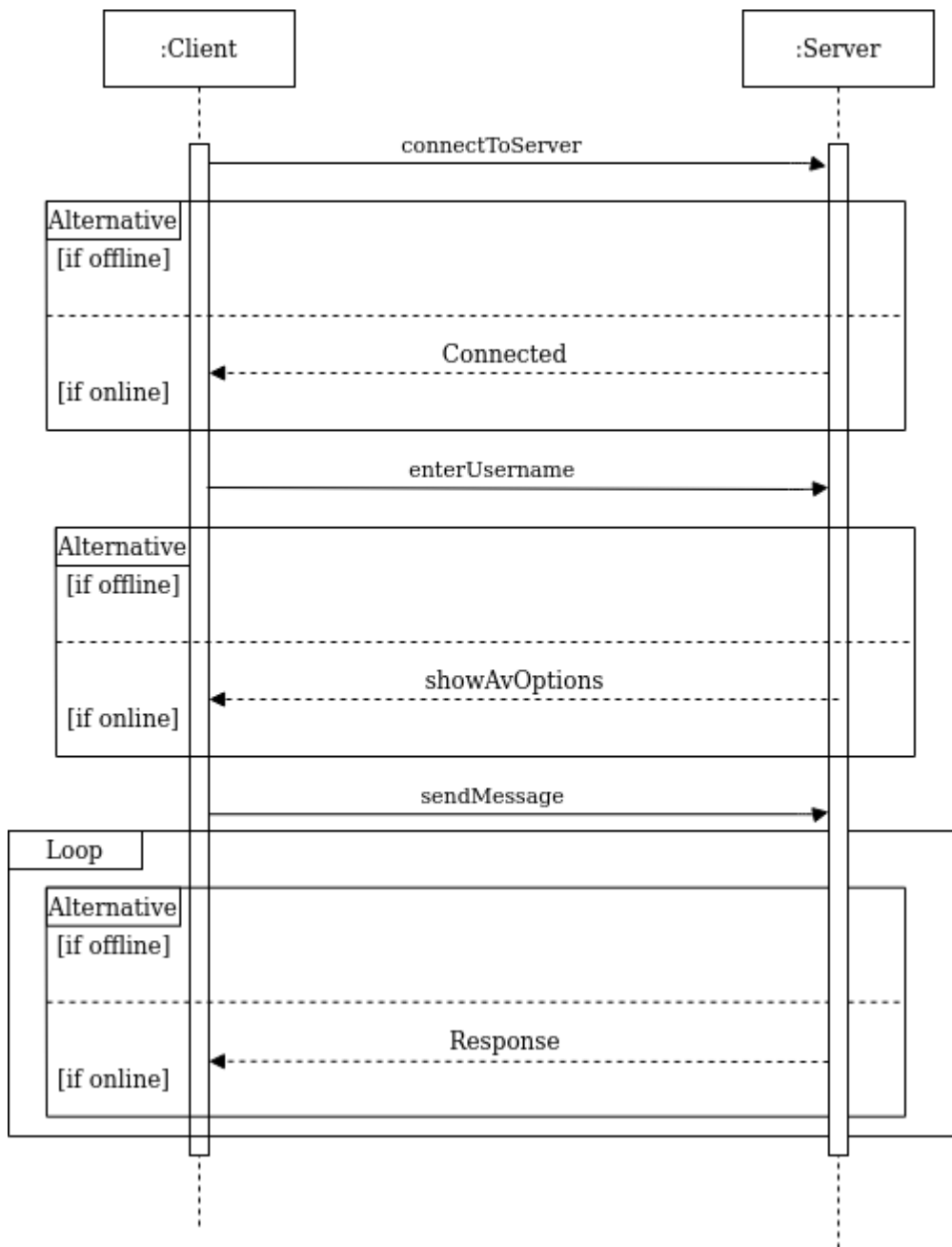
Below is a screenshot of the server accepting clients.

Client

The Client class is in the chat file, this is file to be run by all the users that want to run use the Chat Application. A client has one socket that is responsible for sending and receiving messages or request to and from the server. After running the chat file, the client firstly reaches out to try and establish a connection with the Server before any messages can be sent. If 15 seconds pass the without establishing the connection, the client is notified that it could not reach the server (*this could be because of either, server is offline, wrong server address or data is not flowing through the network*). If so, try to re-connect to the Server, If the mentioned possible possibilities are fixed, the client will receive a confirmation from server that it is connected.

```
while not clientObj.isConnected:
    clientObj.connectToServer()
    count = count+1
    if count > 75:
        print("[couldn't connect to server]")
        exit()
    print("*",end = "")
    time.sleep(0.2)
    sys.stdout.flush()
```

After the reaching the server, the client receives available options and what commands to use in order for server to process requests and messages, each and every request sent to the server is recorded in a file.



Features

Message Transfer

Message structure: comprises of the header and body. The header structure is known to both the client and server, contains fields that describe the actual data in the message. Some of the fields/information contained in the header include the message type, the command, and recipient information.

For example, in the chat(client) program

```

message1 = {'key':'LOGOUT','username':self.username} # requestLogout()
message2 = {'key':'GET_DATA','find':usernameToGet,'username':self.username} # getData()
message3 = {'key':'REQUEST_JOIN_GROUP','username':self.username} # requestToJoinChat()
message4 = {'key':'CONNECT','isConnected':self.isConnected,'username':self.username, 'isStach':False} #
connectToServer()
  
```

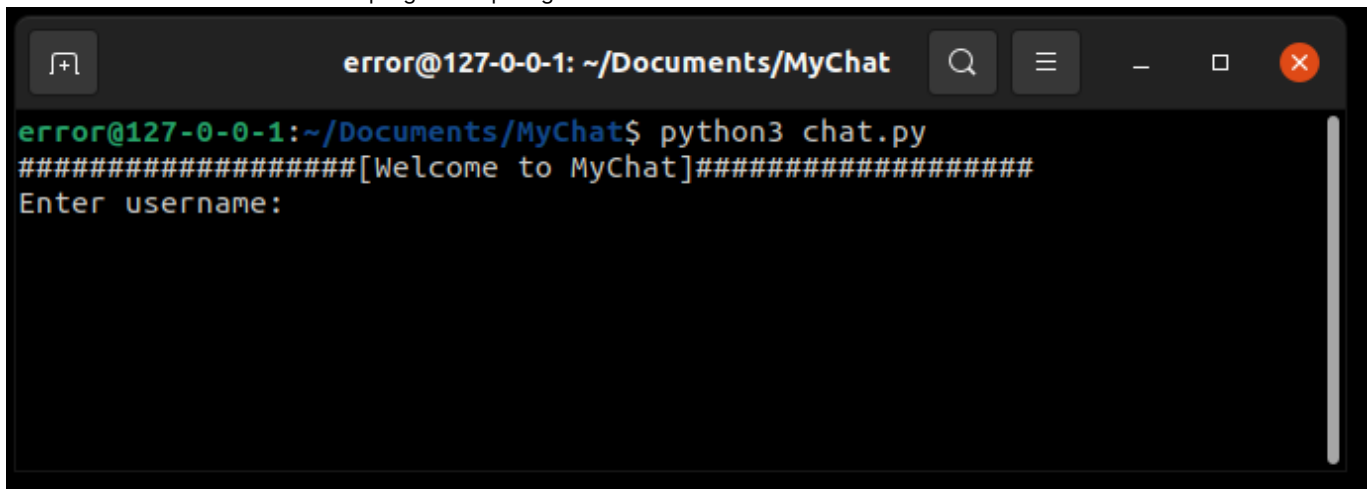
Registration

Before anything can be done on the client side after executing the program.After registration, the server is reached out to check

weather it online or offline.

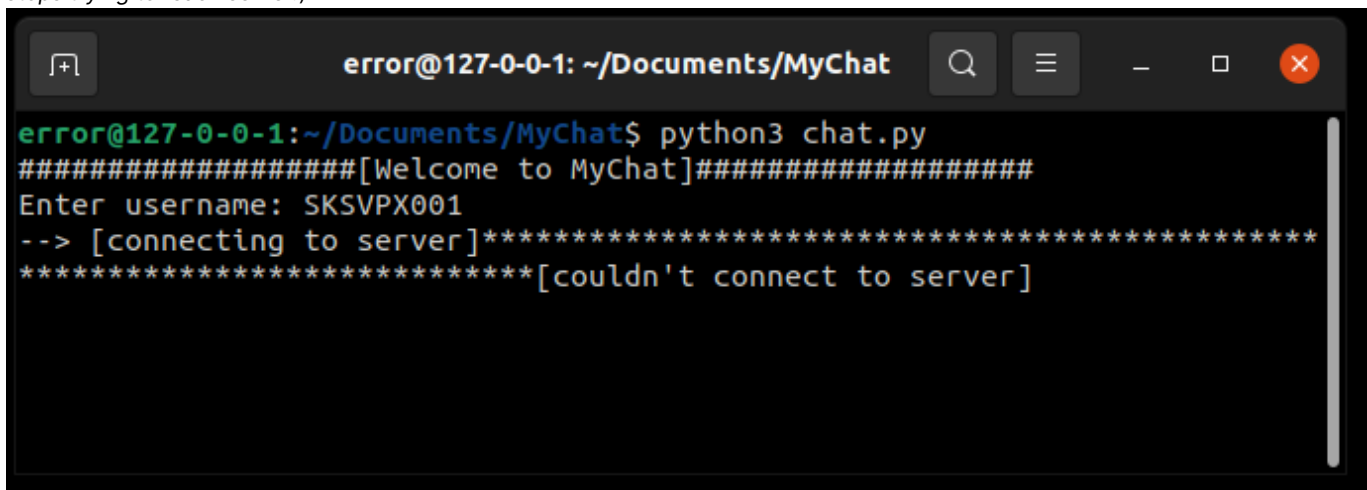
Explanation: A username is required in order to be identified by the server and distinguished from other clients that are also connected to the server.

Below is a screenshot of the client program requiring username after execution.



```
error@127-0-0-1: ~/Documents/MyChat
error@127-0-0-1:~/Documents/MyChat$ python3 chat.py
#####[Welcome to MyChat]#####
Enter username:
```

Below is a screenshot of the client reaching out to server. (note: in this example the server is offline, so after 15 seconds the client stops trying to reach server.)



```
error@127-0-0-1: ~/Documents/MyChat
error@127-0-0-1:~/Documents/MyChat$ python3 chat.py
#####[Welcome to MyChat]#####
Enter username: SKSVPX001
--> [connecting to server]*****
*****[couldn't connect to server]
```

Online and offline feature + Group Chat + One on One Chat

Online and Offline: When users log in, they can automatically see who connected to the server, and who of those is online and offline.

Group Chat: When user wants to communicate the message to all the online users, they can type "J + group chat name"

One on One: To send messages to use a specific users that has connected to the server (logged in or not)

NOTE: After server is executed users logged in this sequence , SKSPVX001, GNY0SC001, MSMJO003

```
error@127-0-0-1: ~/Documents/MyChat(Updated)$ python3 chat.py
#####[Welcome to MyChat]#####
Enter username: SKSPVX001
--> [connecting to server]**[connected to server]
#####[you: SKSPVX001]#####
--> [available group chats]
1. MyChat

--> [type 'JOIN' + 'group chat name' and press ENTER to join the group chat, 'CHAT' + 'user
name' and press ENTER to chat to a user, 'R' to refresh and 'E' to exit]
[]

error@127-0-0-1: ~/Documents/MyChat(Updated)$ python3 chat.py
#####[Welcome to MyChat]#####
Enter username: GNY0SC001
--> [connecting to server]**[connected to server]
#####[you: GNY0SC001]#####
--> [available group chats]
1. MyChat

--> [available users]
1. SKSPVX001 (online)

--> [type 'JOIN' + 'group chat name' and press ENTER to join the group chat, 'CHAT' + 'username' and press ENTER
to chat to a user, 'R' to refresh and 'E' to exit]
[]

error@127-0-0-1: ~/Documents/MyChat(Updated)$ python3 chat.py
#####[Welcome to MyChat]#####
Enter username: MSMJO003
--> [connecting to server]**[connected to server]
#####[you: MSMJO003]#####
--> [available group chats]
1. MyChat

--> [available users]
1. SKSPVX001 (online)
2. GNY0SC001 (online)

--> [type 'JOIN' + 'group chat name' and press ENTER to join the group chat, 'CHAT' + 'usern
ame' and press ENTER to chat to a user, 'R' to refresh and 'E' to exit]
[]

error@127-0-0-1: ~/Documents/MyChat(Updated)$ python3 server.py
--> [MyChat server is now online]
--> [type 'E' and press ENTER to turn server off]
```

Real Time + Message Confirmation (sent and received)

RealTime: When message is sent weather in a group chat or one-on-one chat, they appear on the receivers end screen without them doing anything.

Message Confirmation: When message is sent on the group, verification is done and a message is displayed on the screen to confirm. On a one-on-one chat it goes a step further and show if it was also received .

```
error@127-0-0-1: ~/Documents/MyChat(Updated)$ python3 chat.py
#####[Welcome to MyChat]#####
Enter username: SKSPVX001
--> [connecting to server]**[connected to server]
#####[you: SKSPVX001]#####
--> [available group chats]
1. MyChat

--> [available users]
1. SKSPVX001 (online)

--> [type 'JOIN' + 'group chat name' and press ENTER to join the group chat, 'CHAT' + 'user
name' and press ENTER to chat to a user, 'R' to refresh and 'E' to exit]
JOIN MyChat
#####[GROUP CHAT]#####
--> [Access granted]
--> [you joined 'MyChat' group]
--> [type message and press ENTER or type 'B' to return to menu]
hey My Name is Prince
--> [Message sent - 08:33:46]

error@127-0-0-1: ~/Documents/MyChat(Updated)$ python3 chat.py
#####[Welcome to MyChat]#####
Enter username: GNY0SC001
--> [connecting to server]**[connected to server]
#####[you: GNY0SC001]#####
--> [available group chats]
1. MyChat

--> [available users]
1. SKSPVX001 (online)

--> [type 'JOIN' + 'group chat name' and press ENTER to join the group chat, 'CHAT' + 'user
name' and press ENTER to chat to a user, 'R' to refresh and 'E' to exit]
JOIN MyChat
#####[GROUP CHAT]#####
--> [Access granted]
--> [you joined 'MyChat' group]
--> [type message and press ENTER or type 'B' to return to menu]
>> SKSPVX001: hey My Name is Prince

error@127-0-0-1: ~/Documents/MyChat(Updated)$ python3 chat.py
#####[Welcome to MyChat]#####
Enter username: MSMJO003
--> [connecting to server]**[connected to server]
#####[you: MSMJO003]#####
--> [available group chats]
1. MyChat

--> [available users]
1. SKSPVX001 (online)
2. GNY0SC001 (online)

--> [type 'JOIN' + 'group chat name' and press ENTER to join the group chat, 'CHAT' + 'usern
ame' and press ENTER to chat to a user, 'R' to refresh and 'E' to exit]
JOIN MyChat
#####[GROUP CHAT]#####
--> [Access granted]
--> [you joined 'MyChat' group]
--> [type message and press ENTER or type 'B' to return to menu]
>> SKSPVX001: hey My Name is Prince

error@127-0-0-1: ~/Documents/MyChat(Updated)$ python3 server.py
--> [MyChat server is now online]
--> [type 'E' and press ENTER to turn server off]
```

Message retrieval: If a user A was is online and joins group chat and becomes offline, and while other users (B and C) continue to communicate through the group, when the user come back online, all the messages that user A did not see, will be retrieved and displayed on user A's CLI.

NOTE:

After using server and clients, if you want to use it again, remove all the contents in the `database.txt` file, but DO NOT delete it.