Distributed Systems
Assignment – 2

Rishab Ramanathan – 19XJ1A0558
Ananta Srikar Puranam – 19XJ1A0507

Index :

Logic/Algorithm :

Server:

- one single server side socket, to which every client connects (using a common accessible ip and unique port)
- multi threaded program, each new client processed on a separate thread
- clients tracked by appending to a list
- messages broadcast by iterating through client list, with option of excluding a client as a parameter (in the case of broadcasting a client message to <u>every other client)</u>
- server can be shutdown using '\close' in the server terminal shell, closes the server and every connected client by broadcasting an exit command to each client, which is then processed at the client side

Client:

- socket connects to server sockets through input ip and port
- chooses stdin input stream (typing in terminal shell) or server socket for source of message
- if it is a server message, client can either shutdown by server-side close command, or display message sent from the server (usually the message sent by other clients in the room)
- if it is an stdin message, client can either shutdown by '\close' command, or send server socket the typed in message (which is then sent to every other client in the room by the server)

Code :

server.py :

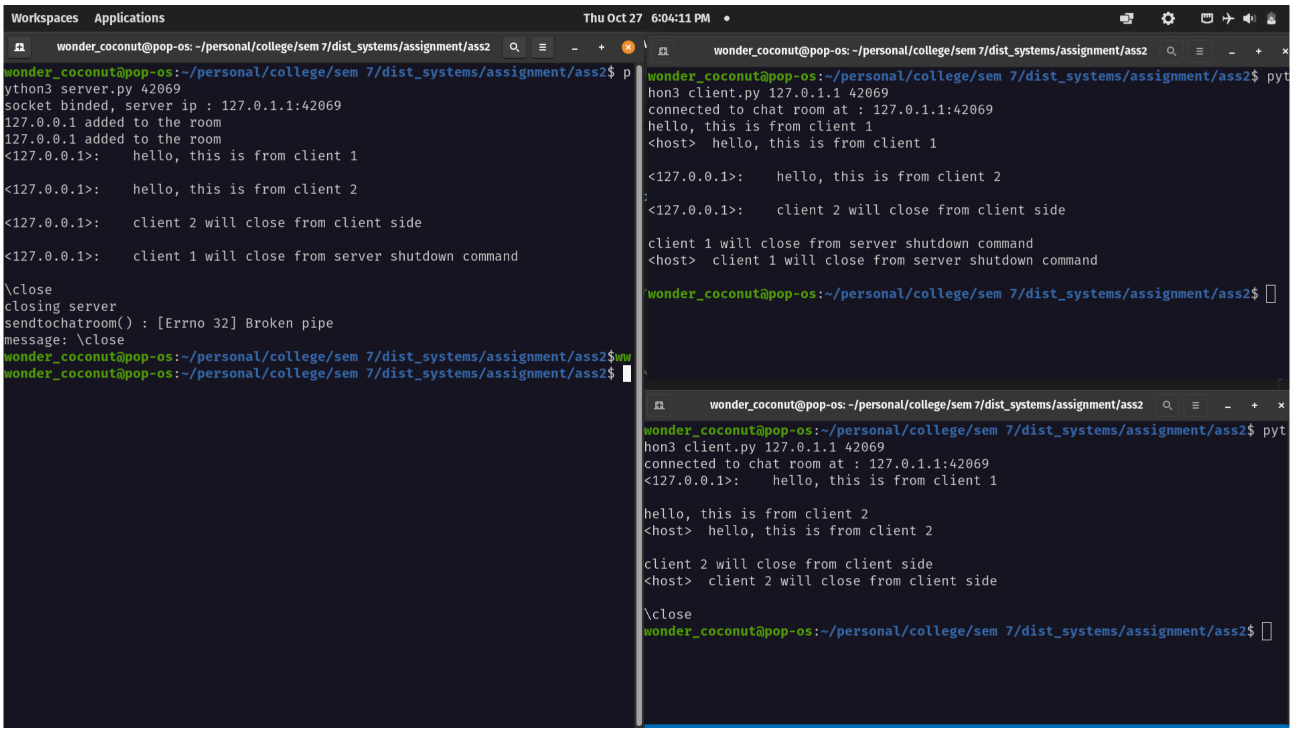
```
import socket
import sys
import _thread
import os

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

host_name = socket.gethostname()
host_ip = socket.gethostbyname(host_name) #automatically detects server ip

if (len(sys.argv) != 2) :
print("arguments syntax : <port number>")
exit()

port = int(sys.argv[1])

server.bind((host_ip, port))
print(f"socket binded, server ip : {host_ip}:{port}")

server.listen(25)

clientList = []

def closeserver(): #function to shutdown server and connected clients
input = sys.stdin.readline()
if(input == "\close\n"):
print('closing server')
server.close()
sendtochatroom('\close', None) #shutdown command to clients
os._exit(os.EX_OK)

def sendtochatroom(message, client) : #send message to every client except parameter client
for user in clientList :
if (user != client) :
try :
user.send(bytes(message , encoding='utf-8'))
except Exception as e :
print('sendtochatroom() : ' + str(e))
print('message: ' + message)
user.close()
clientList.remove(user) #dead client

def clientthread(client, addr): #new client functionality
client.send(bytes(f"connected to chat room at : {host_ip}:{port}", encoding='utf-8'))

while True:
try:
message = str(client.recv(2048), encoding= 'ascii', errors= 'ignore')
if message:
print("<" + addr[0] + ">:\t" + message)
sendtochatroom("<" + addr[0] + ">:\t" + message,client) #display message to every other connected client
except Exception as e:
print('clientthread():' + str(e))
continue
```

```
_thread.start_new_thread(closeserver, ())

while True :

client, addr = server.accept() #new client
clientList.append(client)
print(addr[0] + " added to the room")
_thread.start_new_thread(clientthread,(client,addr)) #seperate thread for each client
```

client.py :

```
import socket
import select
import sys

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

if (len(sys.argv) != 3) :
print("arguments syntax : <server ip> <port number>")
exit()

server_ip = str(sys.argv[1])
port = int(sys.argv[2])
server.connect((server_ip, port))

while True:

socketlist = [sys.stdin,server]

readsocket, writesocket, errorsocket = select.select(socketlist, [], []) #server socket or terminal input for message

for socket in readsocket:
if (socket == server) : #server message incoming
message = str(socket.recv(2048), encoding= 'ascii', errors= 'ignore')
if(message == '\close'):
server.close()
exit()
print(message)

else : #client message outgoing
message = sys.stdin.readline()
if(message == '\close\n'):
server.close()
exit()
server.send(bytes(message, 'utf-8'))
print('<host>\t' + message)
```

Output Screenshot :


Workspaces
Applications
Thu Oct 27 6:04:11 PM
wonder_coconut@pop-os: ~/personal/college/sem 7/dist_systems/assignment/ass2
wonder_coconut@pop-os: ~/personal/college/sem 7/dist_systems/assignment/ass2
wonder_coconut@pop-os: ~/personal/college/sem 7/dist_systems/assignment/ass2

```
wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignment/ass2$ p
ython3 server.py 42069
socket binded, server ip : 127.0.1.1:42069
127.0.0.1 added to the room
127.0.0.1 added to the room
<127.0.0.1>:    hello, this is from client 1

<127.0.0.1>:    hello, this is from client 2

<127.0.0.1>:    client 2 will close from client side

<127.0.0.1>:    client 1 will close from server shutdown command

\close
closing server
sendtochatroom() : [Errno 32] Broken pipe
message: \close
wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignment/ass2$ww
wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignment/ass2$
```

```
wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignment/ass2$ pyt
hon3 client.py 127.0.1.1 42069
connected to chat room at : 127.0.1.1:42069
hello, this is from client 1
<host>  hello, this is from client 1

<127.0.0.1>:    hello, this is from client 2

<127.0.0.1>:    client 2 will close from client side

client 1 will close from server shutdown command
<host>  client 1 will close from server shutdown command

wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignment/ass2$
```

```
wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignment/ass2$ pyt
hon3 client.py 127.0.1.1 42069
connected to chat room at : 127.0.1.1:42069
<127.0.0.1>:    hello, this is from client 1

hello, this is from client 2
<host>  hello, this is from client 2

client 2 will close from client side
<host>  client 2 will close from client side

\close
wonder_coconut@pop-os:~/personal/college/sem 7/dist_systems/assignment/ass2$
```

Github link (contains readme) :

Github repository for distributed systems assigment 2

or plaintext link :

https://github.com/wonder-coconut/dist_system_assignment/tree/master/ass2