Text

Description automatically generated

Figure 1: ScreenShot of result

**SERVER program**

import socket

import threading

HEADER = 64

# port number can only be 0-65535

PORT = 5555

# we can either manually check the host ip using ifconfig in terminal

# but the better way to call socket.gethostname() function that automatically fetches it for us

SERVER = socket.gethostbyname(socket.gethostname())

ADDR = (SERVER, PORT)

FORMAT = 'utf-8'

# SOCK\_STREAM is for TCP

# we can change it to SOCK\_SEQPACKET for UDP if needed

# socket.AF\_INET is telling it that we are using IPV4 address

# similarly we can change it to AF\_INET6 for IPV6 addressing

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# binding ip address and port number

server.bind(ADDR)

import socket

import threading

HEADER = 64

PORT = 5555

# we can either manually check the host ip using ifconfig in terminal

# but the better way to call socket.gethostname() function that automatically fetches it for us

SERVER = socket.gethostbyname(socket.gethostname())

ADDR = (SERVER, PORT)

# SOCK\_STREAM is for TCP

# we can change it to SOCK\_SEQPACKET for UDP if needed

# socket.AF\_INET is telling it that we are using IPV4 address

# similarly we can change it to AF\_INET6 for IPV6 addressing

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind(ADDR)

# handle\_client is the main program

# we are using python socket library that has all the functions required

def handle\_client(conn, addr):

print(f"NEW CONNECTION {addr} connected.")

connected = True

while connected:

# .recv is the function call that gets the msg from connected client

msg\_length = conn.recv(HEADER).decode('utf-8')

# if recieved a msg other wise it will go on infinite loop

if msg\_length:

msg\_length = int(msg\_length)

# decode into utf-8 format to be readble by us

msg = conn.recv(msg\_length).decode('utf-8')

print(f"{addr} {msg}")

# calling the reverse fiunction to reverse the message

conn.send(f"reversed message: {reverse(msg)}".encode(FORMAT))

if msg == "DISCONNECT":

connected = False

conn.close()

def reverse(string):

return string[::-1]

# start is the function for actual call and it is threaded

def start():

server.listen()

print(f"LISTENING on {SERVER}...")

while True:

conn, addr = server.accept()

thread = threading.Thread(target=handle\_client, args=(conn, addr))

thread.start()

print(f"ACTIVE CONNECTIONS: {threading.activeCount() - 1}")

print("SERVER STARTING")

start()

**CLIENT program**

**\*Code explained in the video**

import socket

HEADER = 64

PORT = 5555

FORMAT = 'utf-8'

DISCONNECT\_MESSAGE = "!DISCONNECT"

SERVER = socket.gethostbyname(socket.gethostname())

ADDR = (SERVER, PORT)

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect(ADDR)

def send(msg):

message = msg.encode(FORMAT)

msg\_length = len(message)

send\_length = str(msg\_length).encode(FORMAT)

send\_length += b' ' \* (HEADER - len(send\_length))

client.send(send\_length)

client.send(message)

print(client.recv(2048).decode(FORMAT))

send(f"Hello from client {SERVER}")

msg = ""

while msg != "quit":

msg = input()

send(msg)

send("DISCONNECT")

**7) What are sockets and on which layer do they operate?**

Sockets operate at application layer and are bound to a PORT number. Sockets allow process to process communication between two machines. It is basically a one endpoint of a two-way communication link between two programs running on the network.

**8) Differentiate between TCP and UDP? This assignment is based on TCP or UDP? YouTube uses TCP   
or UDP?**

TCP is connection oriented which means the data will only be sent after a connection is established via 3-way handshake. It is secure and reliable and used for most applications which are not time sensitive. It ensures data is in right order and error free. Use cases are emails, recorded media, webpages, bank transactions.

UDP on the other hand is not connection oriented and does not care whether the data received is in right order and error free or not. Because of that, it is perfect for streaming data live live videos, calls etc. once the packet is lost, UDP does not ask for retransmission. Due to low overhead, it is a lot faster than TCP

This assignment was based on TCP

Youtube, contrary to what we might think, uses TCP over UDP and it makes sense because these video are recorded and it not time sensitive like watching a live sports match. Also, you need to pause and go back and watch again so in that case it makes sense to send a reliable data that is also high resolution and error free. Also, it might be because of the reason that UDP can’t be throttled. Hence TCP makes sense for regulations and better control.

**9) What will happen if I use an out-of-range port number in my code? Will I encounter error? If yes then why and if No then why?**

Yes, it will throw an error:

OverflowError: bind(): port must be 0-65535.

Moreover, port numbers should also be available for example port 80 is used by HTTP

**10) What is the maximum number of sockets that a client and a server can have?**

It depends of number of available ports. Ideally if all the ports are available, a server can serve **65535** clients simultaneously and similarly a client can have **65535** threaded connections to a server