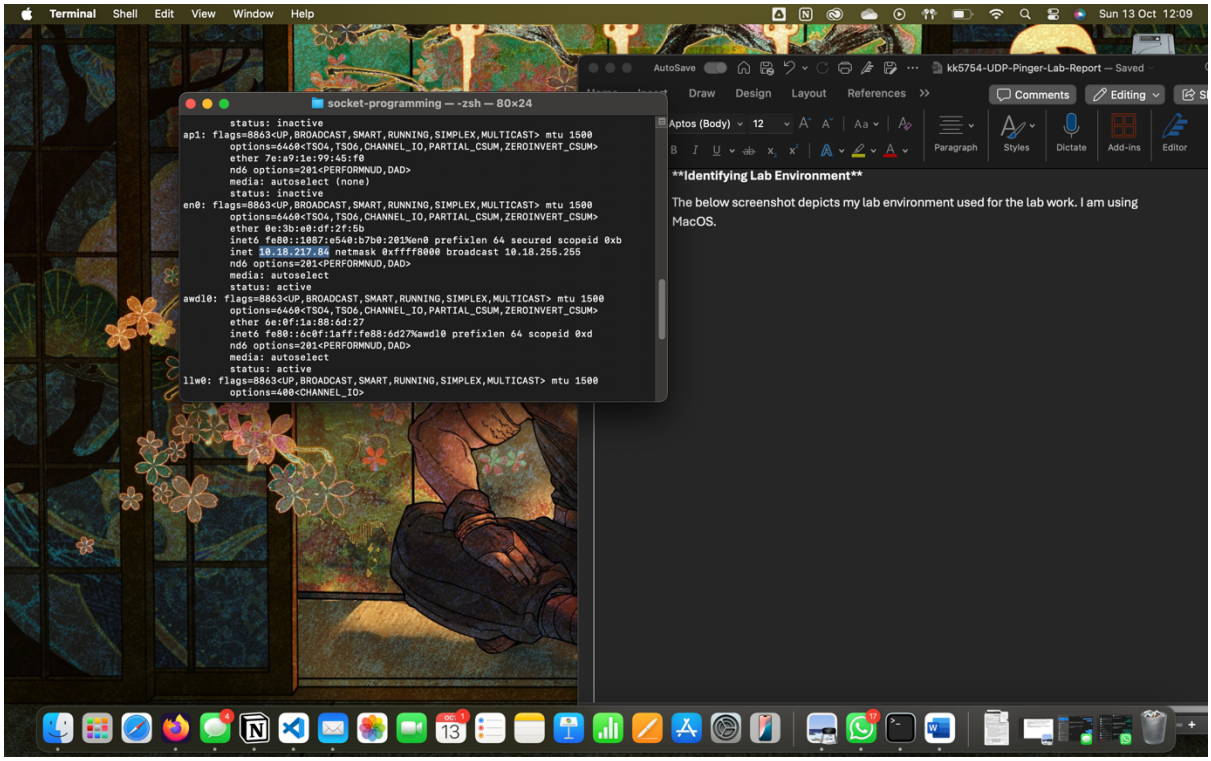
Name: Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

Computer Networking Socket Programming Lab 2: UDP Pinger Lab

Identifying Lab Environment

The below screenshot depicts my lab environment used for the lab work. I am using MacOS with Python version 3.9.11.



Name: Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

1. UDP Pinger Lab: Client Code Completed with commented lines for explanation of code

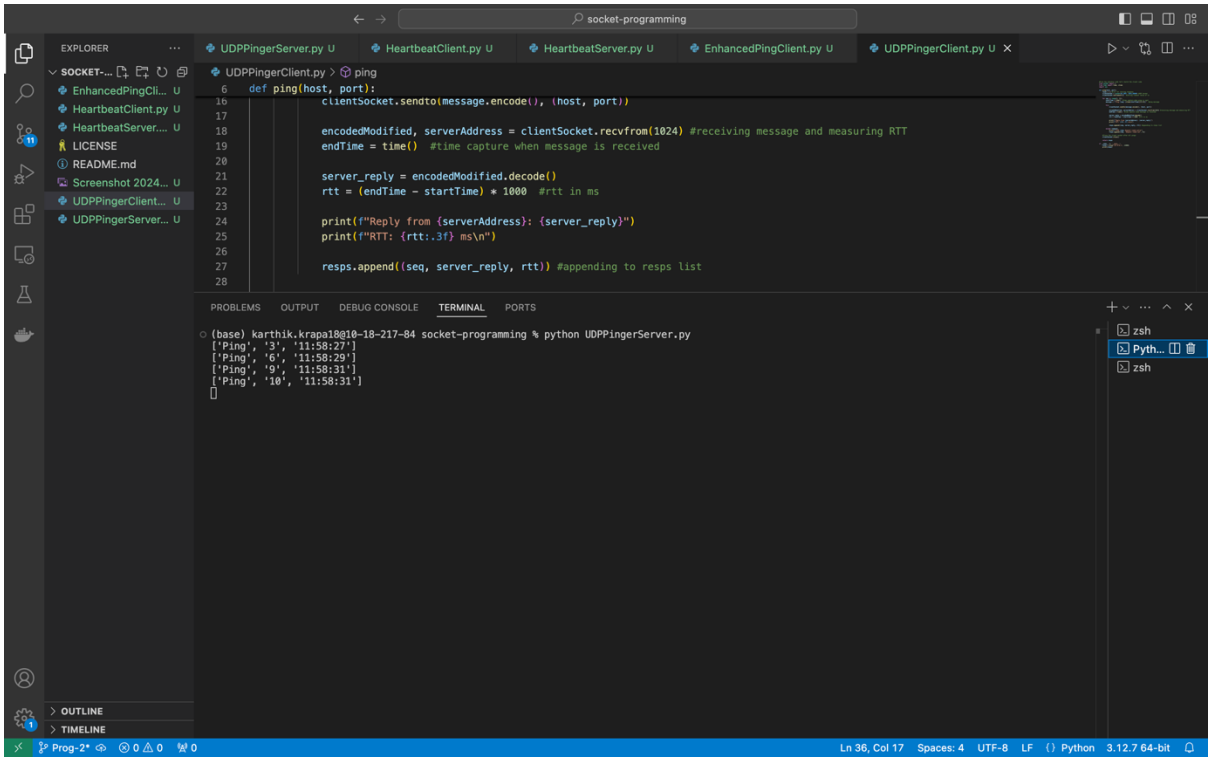
   client.py

```python
#From the skeleton code let's build the client code
from socket import *
from time import time, ctime
import sys

def ping(host, port):
    resps = [] #List for storing response
    clientSocket = socket(AF_INET, SOCK_DGRAM) #UDP Socket
    clientSocket.settimeout(1)  #setting timeout value of 1s

    for seq in range(1, 11):
        startTime = time()  #time capture when ping is sent
        message = f"Ping {seq} {ctime(startTime)[11:19]}"  #ping message

        try:
            clientSocket.sendto(message.encode(), (host, port))

            encodedModified, serverAddress = clientSocket.recvfrom(1024) #receiving
message and measuring RTT
            endTime = time()  #time capture when message is received

            server_reply = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  #rtt in ms

            print(f"Reply from {serverAddress}: {server_reply}")
            print(f"RTT: {rtt:.3f} ms\n")

            resps.append((seq, server_reply, rtt)) #appending to resps list

        except timeout:
            print(f"PING {seq} Request timed out\n")
            resps.append((seq, 'Request timed out', 0))

    #Close the client socket after all pings
    clientSocket.close()

    return resps

if __name__ == '__main__':
    resps = ping('127.0.0.1', 12000)
    print(resps)
```

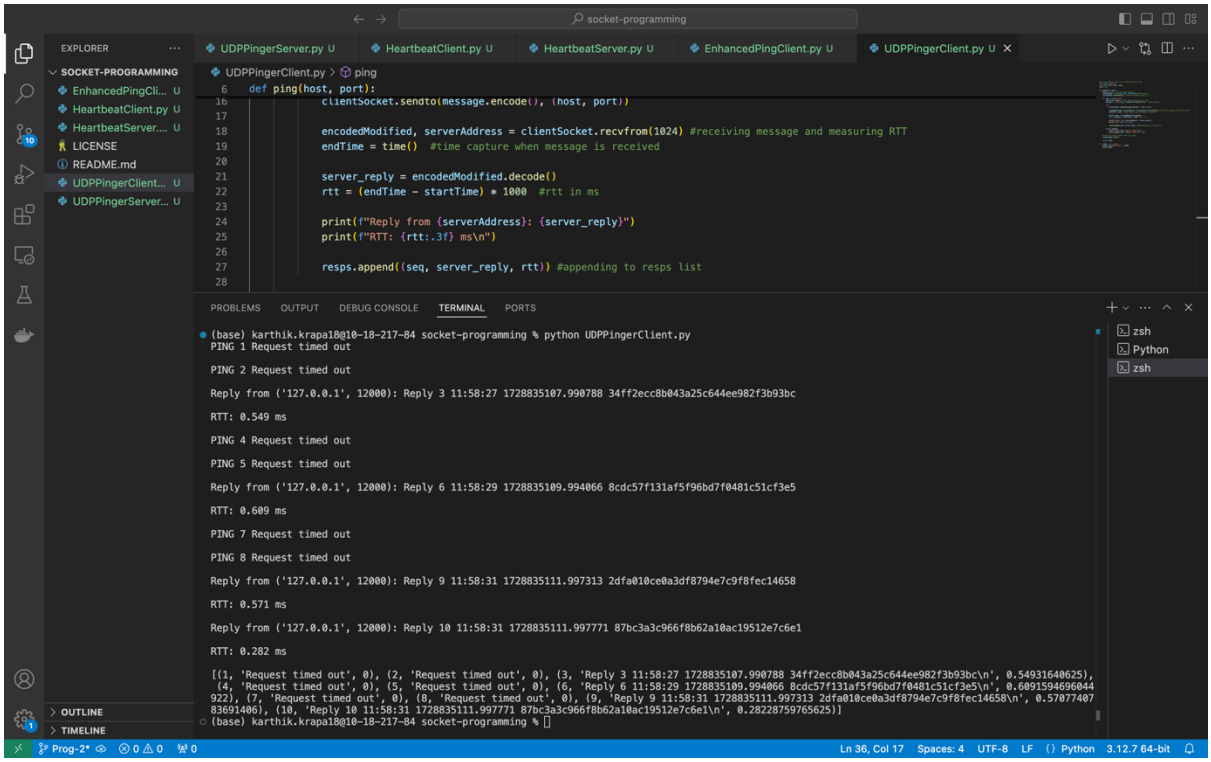
   Output:

Name: Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

Running the UDPPingerServer.py



Running the client.py



2. Optional Exercise: Enhanced UDPPingerClient with RTT Statistics and Packet Loss Rate

Name: Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

EnhancedPingClient.py

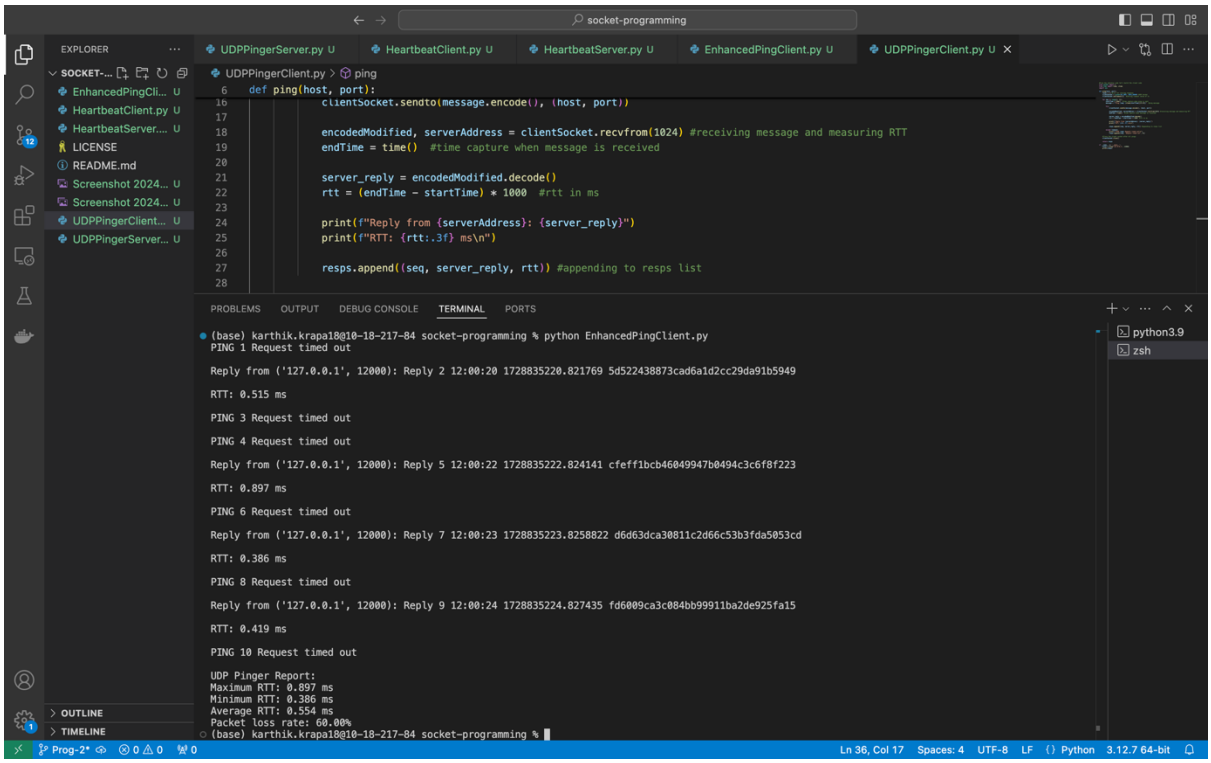
```
from socket import *
from time import time, ctime
import sys

def ping(host, port):
    rtts = []  #to store the rtts
    packet_loss_count = 0  #packet lost counter

    clientSocket = socket(AF_INET, SOCK_DGRAM)
    clientSocket.settimeout(1)

    for seq in range(1, 11):
        startTime = time()
        message = f"Ping {seq} {ctime(startTime)[11:19]}"

        try:
            clientSocket.sendto(message.encode(), (host, port))

            encodedModified, serverAddress = clientSocket.recvfrom(1024)
            endTime = time()

            server_reply = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  #rtt in ms
            rtts.append(rtt) #storing the rtt in list for getting stat report on
this

            print(f"Reply from {serverAddress}: {server_reply}")
            print(f"RTT: {rtt:.3f} ms\n")

        except timeout:
            print(f"PING {seq} Request timed out\n")
            packet_loss_count += 1

    #clising client
    clientSocket.close()

    #rtt stats
    if rtts:
        print("UDP Pinger Report:")
        print(f"Maximum RTT: {max(rtts):.3f} ms") #max value from the list
        print(f"Minimum RTT: {min(rtts):.3f} ms") #min value from the list
        print(f"Average RTT: {sum(rtts)/len(rtts):.3f} ms") #avg calculation from
the list
    else:
        print("No successful pings, all requests timed out.")

    #Calculating Packet Loss Rate
```

Name: Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

```
    total_packets = 10 #since 10 Pings were made from the client skeleton code
earlier
    packet_loss_rate = (packet_loss_count / total_packets) * 100
    print(f"Packet loss rate: {packet_loss_rate:.2f}%")

if __name__ == '__main__':
    ping('127.0.0.1', 12000)
```

Output:

Running EnhancedPingClient.py



3. Optional Exercise: UDP Heartbeat

HeartbeatServer.py

```
from socket import *
import time
import sys

def serve(port, heartbeat_timeout=5):
    serverSocket = socket(AF_INET, SOCK_DGRAM)
    serverSocket.bind(('', port))
    last_seq = -1  #to keep track of sequence number for detecting lost packets
    last_heartbeat_time = time.time()  #the last heartbeat was received time

    print("Server is ready to receive heartbeat signals.")
```

```
    while True:
        try:
            message, address = serverSocket.recvfrom(1024)
            current_time = time.time()
            heartbeat = message.decode().split() #heartbeat message decode
            seq = int(heartbeat[1])
            c_time = float(heartbeat[2])

            print(f"Received Heartbeat {seq} from {address} at {current_time}, sent
at {c_time}")

            if seq != last_seq + 1 and last_seq != -1: #checking pakect loss
                print(f"Warning: Packet loss detected! Expected seq {last_seq + 1},
but got {seq}.")

            if current_time - last_heartbeat_time > heartbeat_timeout: #checking
heartbeat timeout
                print("Warning: Heartbeat timeout. Client may have stopped.")

            #Updating last sequence number and heartbeat time
            last_seq = seq
            last_heartbeat_time = current_time

        except KeyboardInterrupt:
            serverSocket.close()
            sys.exit()

if __name__ == '__main__':
    serve(12000)
```

HeartbeatClient.py

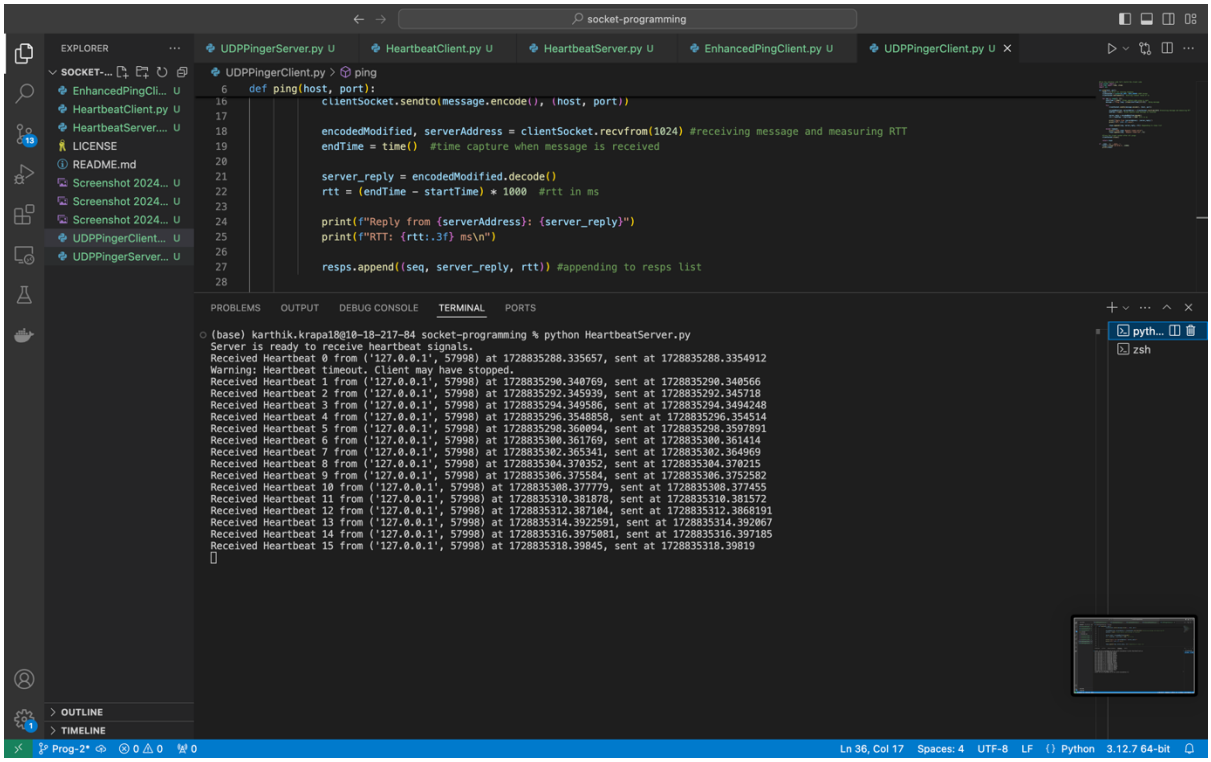
```
from socket import *
import time
import sys

def heartbeat(host, port, interval=2):
    seq = 0  #Sequence number for heartbeat messages

    clientSocket = socket(AF_INET, SOCK_DGRAM)

    while True:
        try:
            current_time = time.time()  #Current time
            message = f"Heartbeat {seq} {current_time}"

            #Sending heartbeat to server
            clientSocket.sendto(message.encode(), (host, port))
```

Name: Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

```
            print(f"Sent Heartbeat {seq} at {current_time}")
            seq += 1

            #Wait for the specified interval before sending the next heartbeat
            time.sleep(interval)

        except KeyboardInterrupt:
            print("Stopping the heartbeat client.")
            clientSocket.close()
            sys.exit()

if __name__ == '__main__':
    heartbeat('127.0.0.1', 12000)
```

Output:

Running HeartbeatServer.py



Running HeartbeatClient.py

Name: Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754