CMPE 472 – Programming Assignment 2: UDP Pinger

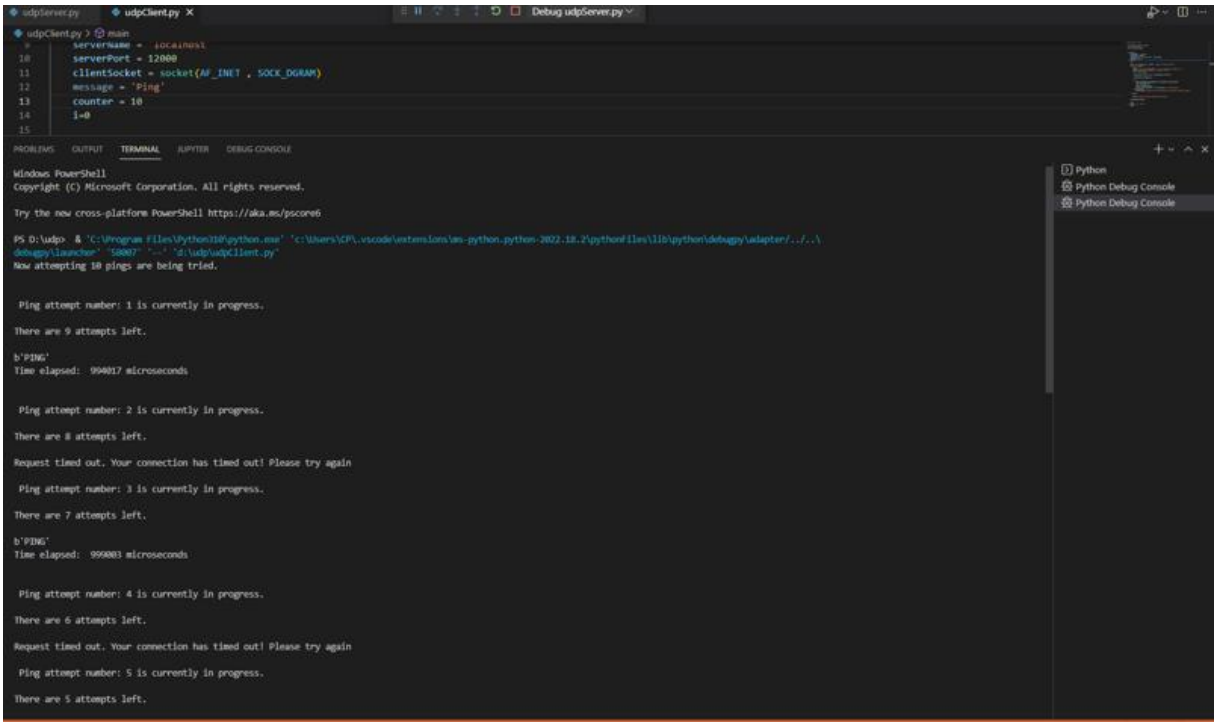
Bensu Şeker

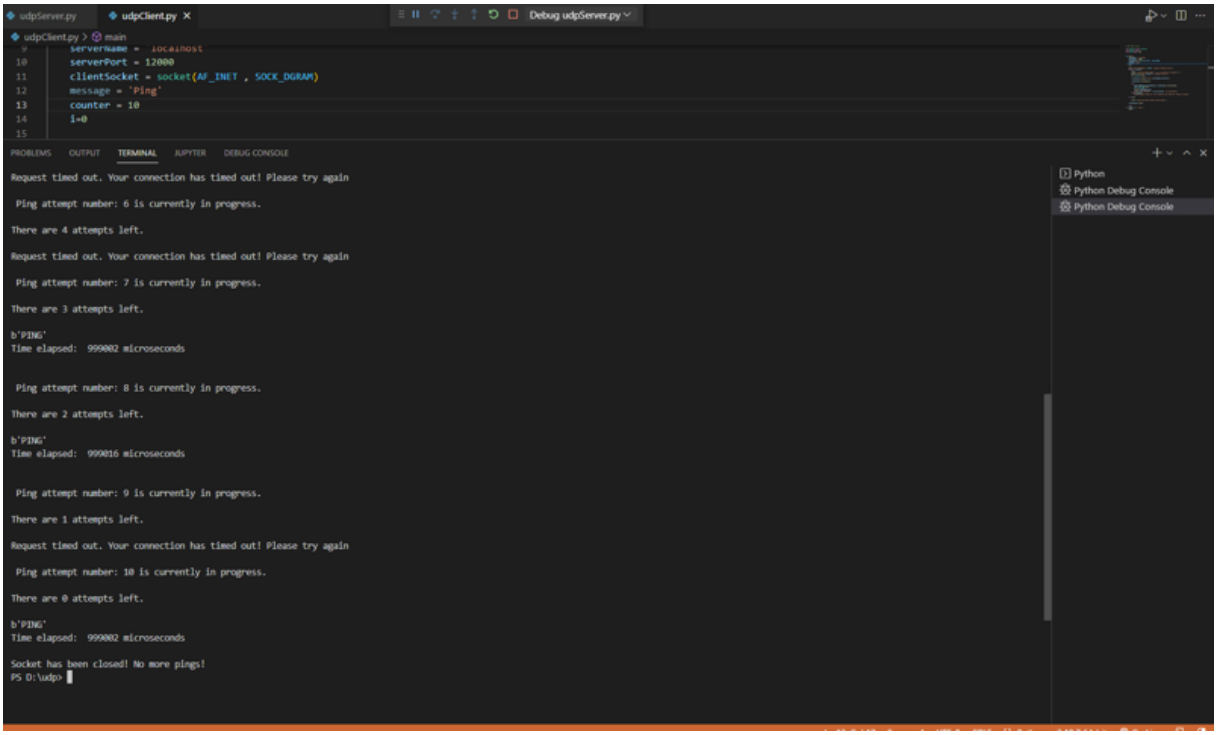
I had a hands-on introduction to Python socket programming for UDP throughout this assignment. I discovered how to use UDP sockets for datagram packet transmission and reception as well as how to establish the right socket timeout. The programs functionality is askin to what the standard ping apps found in modern operating systems can do. These programs communicate with one another using UDP, a less complicated protocol than the widely used Internet Control Message Protocol (ICMP). The ping protocol enables a client computer to broadcast a data packet to a remote computer, which subsequently transmits the data back to the client computer intact (this action is called echoing) (this action is called echoing). The ping protocol, among other things, permits hosts to provide round-trip times to other computers. Ten pings from the client should be sent to the server. A packet transmitted from the client to the server or vice versa may get lost in the network since UDP is an unstable protocol. This makes it impossible for the client to wait endlessly for a ping message's response. If no response is received after one second, my client software should presume that the packet was lost during network transmission, according to the code I provided. To learn how to change a datagram socket's timeout value, consult the Python manual.

- I use UDP to deliver the ping message.

- I print the server's response message, if any.

where time is the moment the client transmits the message, and sequence number begins at 1 and increases by 10 for each subsequent ping message delivered by the client.


udpServer.py
udpClient.py
Debug udpServer.py
udpClient.py > main
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET , SOCK_DGRAM)
message = 'Ping'
counter = 10
i=0
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS D:\udp> & 'C:\Program Files\Python310\python.exe' 'c:\Users\CP\.vscode\extensions\ms-python.python-2022.18.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '58007' '--' 'd:\udp\udpClient.py'
Now attempting 10 pings are being tried.
Ping attempt number: 1 is currently in progress.
There are 9 attempts left.
b'PING'
Time elapsed: 994017 microseconds
Ping attempt number: 2 is currently in progress.
There are 8 attempts left.
Request timed out. Your connection has timed out! Please try again
Ping attempt number: 3 is currently in progress.
There are 7 attempts left.
b'PING'
Time elapsed: 999083 microseconds
Ping attempt number: 4 is currently in progress.
There are 6 attempts left.
Request timed out. Your connection has timed out! Please try again
Ping attempt number: 5 is currently in progress.
There are 5 attempts left.
Python
Python Debug Console
Python Debug Console


udpServer.py
udpClient.py
Debug udpServer.py
udpClient.py > main
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET , SOCK_DGRAM)
message = 'Ping'
counter = 10
i=0
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
Request timed out. Your connection has timed out! Please try again
Ping attempt number: 6 is currently in progress.
There are 4 attempts left.
Request timed out. Your connection has timed out! Please try again
Ping attempt number: 7 is currently in progress.
There are 3 attempts left.
b'PING'
Time elapsed: 999002 microseconds
Ping attempt number: 8 is currently in progress.
There are 2 attempts left.
b'PING'
Time elapsed: 999016 microseconds
Ping attempt number: 9 is currently in progress.
There are 1 attempts left.
Request timed out. Your connection has timed out! Please try again
Ping attempt number: 10 is currently in progress.
There are 0 attempts left.
b'PING'
Time elapsed: 999002 microseconds
Socket has been closed! No more pings!
PS D:\udp>
Python
Python Debug Console
Python Debug Console

udpServer.py Code:

```
# UDPPingerServer.py
# We will need the following module to generate randomized lost packets
import random
from socket import *

# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets

serverSocket = socket(AF_INET, SOCK_DGRAM)

# Assign IP address and port number to socket

serverSocket.bind(('', 12000))
print("Ready to serve")

while True:
    # Generate random number in the range of 0 to 10
    rand = random.randint(0, 10)
    # Receive the client packet along with the address it is coming from
    message, address = serverSocket.recvfrom(1024)
    # Capitalize the message from the client
    message = message.upper()
     #If rand is less is than 4, we consider the packet lost and do not
respond

    if rand <4:
        continue
        # Otherwise, the server responds
    serverSocket.sendto(message, address)
```

udpClient.py Code:

```
# UDP Pinger Client

from datetime import datetime
from socket import *
from time import time


def main():
    serverName = 'localhost'
    serverPort = 12000
    clientSocket = socket(AF_INET , SOCK_DGRAM)
    message = 'Ping'
    counter = 10
    i=0
```

```
    print ('Now attempting', counter, 'pings are being tried.\n')
    while i < counter:
        i+=1
        print ('\n Ping attempt number:', i, 'is currently in progress.\n ')
        print ('There are', counter-i ,'attempts left.\n')
        dt = datetime.now()

        clientSocket.sendto(b'Ping',(serverName,serverPort))

        clientSocket.settimeout(1)

        try:
            modifiedMessage,serverAddress= clientSocket.recvfrom(1024)
            dt2 = datetime.now()
            et = dt - dt2;
            print (modifiedMessage)
            print ('Time elapsed: ', et.microseconds, 'microseconds\n')
        except timeout:
            print('Request timed out. Your connection has timed out! Please
try again')

    if i==10:

        print ('Socket has been closed! No more pings!')

    clientSocket.close()

    pass
if __name__ == '__main__':
    main()
```