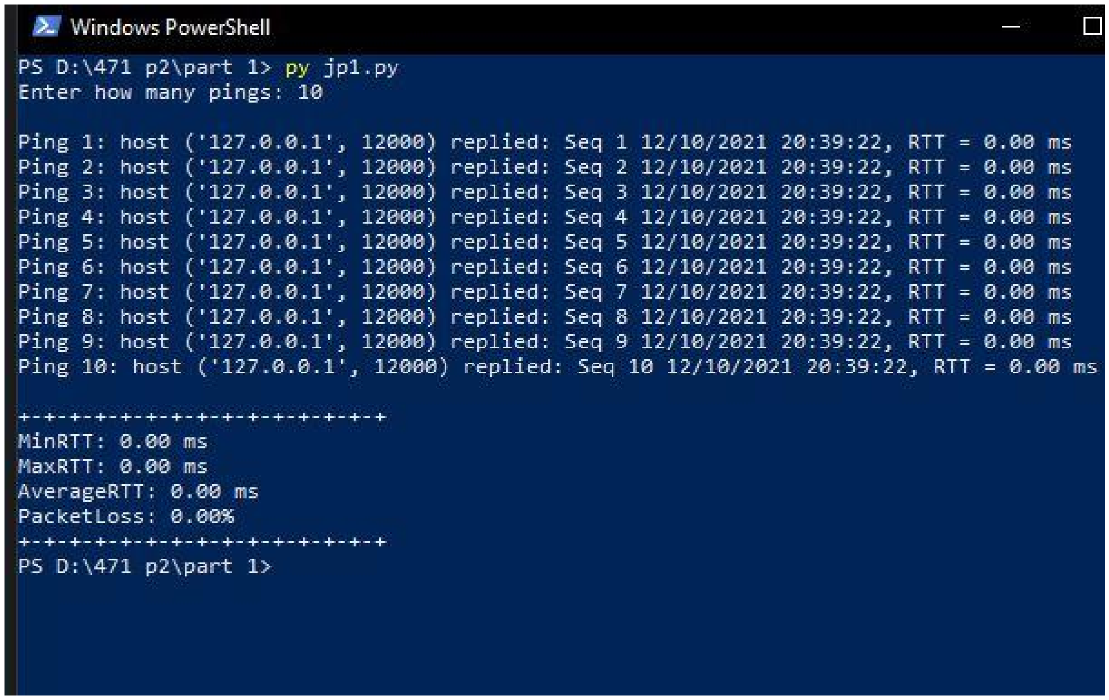
# Project 2: “Python UDP”

Joshua Popp

Using VScode and Python 3.9.7

## Part 1 - UDP Pinger with No Delay and No Loss

1) When the user runs the server and client programs in this case on the local server, the client side will ask for a certain amount of pings. When the user inputs N amount, then the client sends N requests to the server side with a message including the sequence number. The server receives the message and returns the message back to the client side.
2) In python the socket library comes with a function called timeout() which takes a value in seconds. If this value is reached without any server response then the socket experiences a timeout and closes the socket connection.
3) Navigate to the appropriate directory using cmd. With python installed on your system simply type “**py jp1.py**” or “**python jp1.py**” to launch the client side depending on the version of python and system. When done, close the terminal windows to stop the execution. *NOTE: before launching the client side run the server side first for safe measures using the same format .*



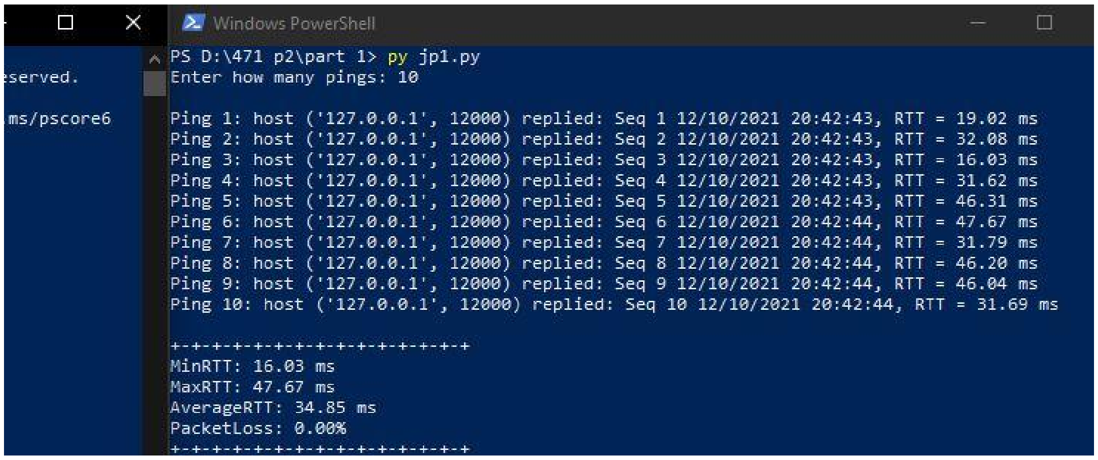
```
PS D:\471 p2\part 1> py jp1.py
Enter how many pings: 10

Ping 1: host ('127.0.0.1', 12000) replied: Seq 1 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 2: host ('127.0.0.1', 12000) replied: Seq 2 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 3: host ('127.0.0.1', 12000) replied: Seq 3 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 4: host ('127.0.0.1', 12000) replied: Seq 4 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 5: host ('127.0.0.1', 12000) replied: Seq 5 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 6: host ('127.0.0.1', 12000) replied: Seq 6 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 7: host ('127.0.0.1', 12000) replied: Seq 7 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 8: host ('127.0.0.1', 12000) replied: Seq 8 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 9: host ('127.0.0.1', 12000) replied: Seq 9 12/10/2021 20:39:22, RTT = 0.00 ms
Ping 10: host ('127.0.0.1', 12000) replied: Seq 10 12/10/2021 20:39:22, RTT = 0.00 ms

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
MinRTT: 0.00 ms
MaxRTT: 0.00 ms
AverageRTT: 0.00 ms
PacketLoss: 0.00%
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
PS D:\471 p2\part 1>
```

4) *Code:*

```
from socket import *
from datetime import datetime
from time import sleep
import time

# variables
minRTT = 0
maxRTT = 0
avgRTT = 0
totalRTT = 0
lostPackets = 0
packetLossPerc = 0
seqNum = 1
runs = input("Enter how many pings: ")
runAmount = int(runs)
print("")
# socket Info
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.connect((serverName, serverPort))

for x in range(runAmount):
    now = datetime.now()
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
    message = "Seq " + str(seqNum) + " " + dt_string
    clientSocket.sendto(message.encode(), (serverName, serverPort))
    start = time.time() * 1000
    clientSocket.settimeout(1.00000)
    try:
        data, sender_addr = clientSocket.recvfrom(1024)
        print("Ping " + str(x+1) + ": " + "host", end="")
        print(" " + str(sender_addr) + " replied: ", end="")
        print(data.decode() + ", RTT = ", end="")
        end = time.time() * 1000
        current = end - start
        RTTcurrent = format(current, '.2f')
        print(str(RTTcurrent) + " ms")
        totalRTT = totalRTT + current
        if(x == 0):
            minRTT = current
            maxRTT = current
        if (current < minRTT):
            minRTT = current
        if(current > maxRTT):
            maxRTT = current
    except:
        print("Ping " + str(x+1) + ": " + "Request timed out")
        lostPackets = lostPackets + 1
    sleep(0)
    seqNum = seqNum + 1

print("")
print("+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+")
minRTT = format(minRTT, '.2f')
print("MinRTT: " + str(minRTT) + " ms")
maxRTT = format(maxRTT, '.2f')
print("MaxRTT: " + str(maxRTT) + " ms")
avgRTT = totalRTT / runAmount
avgRTT = format(avgRTT, '.2f')
print("AverageRTT: " + str(avgRTT) + " ms")
packetLossPerc = (lostPackets / runAmount) * 100
packetLossPerc = format(packetLossPerc, '.2f')
print("PacketLoss: " + str(packetLossPerc) + "%")
print("+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+")
```

## Part 2 - UDP Pinger with Delays

1) The operation of the ping server is identical to the process from part 1; except for that I included a random number generator from 10 to 40 that returns a number which is then multiplied by 1000 to mimic milliseconds. Before the server responds a sleep function is triggered with the given value to create RTT delays.
2) Navigate to the appropriate directory using cmd. With python installed on your system simply type "**py jp2.py**" or "**python jp2.py**" to launch the client side depending on the version of python and system. When done, close the terminal windows to stop the execution. *NOTE: before launching the client side run the server side first for safe measures using the same format .*



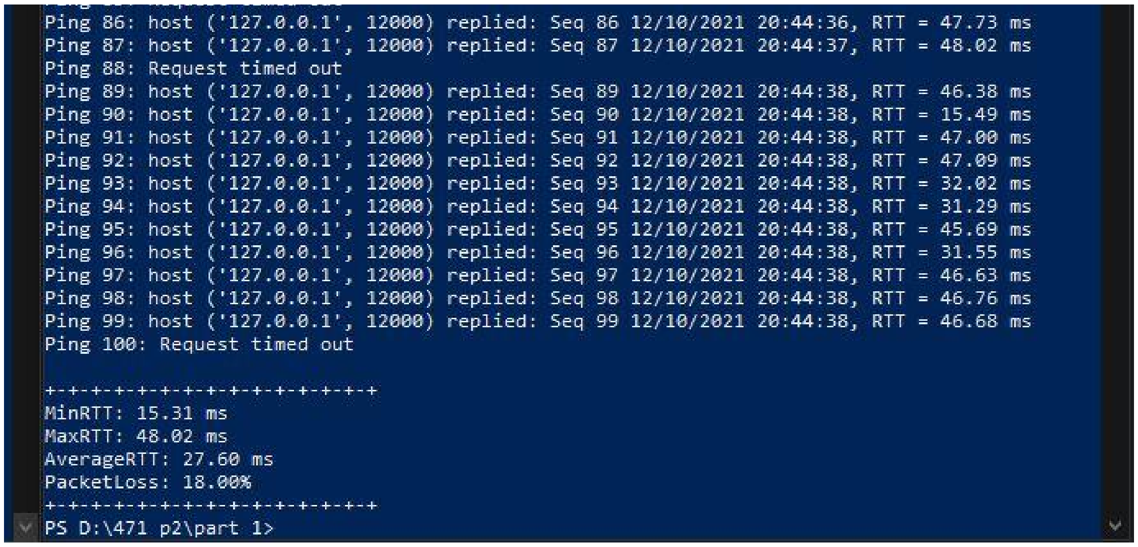
```
PS D:\471 p2\part 1> py jp1.py
Enter how many pings: 10

Ping 1: host ('127.0.0.1', 12000) replied: Seq 1 12/10/2021 20:42:43, RTT = 19.02 ms
Ping 2: host ('127.0.0.1', 12000) replied: Seq 2 12/10/2021 20:42:43, RTT = 32.08 ms
Ping 3: host ('127.0.0.1', 12000) replied: Seq 3 12/10/2021 20:42:43, RTT = 16.03 ms
Ping 4: host ('127.0.0.1', 12000) replied: Seq 4 12/10/2021 20:42:43, RTT = 31.62 ms
Ping 5: host ('127.0.0.1', 12000) replied: Seq 5 12/10/2021 20:42:43, RTT = 46.31 ms
Ping 6: host ('127.0.0.1', 12000) replied: Seq 6 12/10/2021 20:42:44, RTT = 47.67 ms
Ping 7: host ('127.0.0.1', 12000) replied: Seq 7 12/10/2021 20:42:44, RTT = 31.79 ms
Ping 8: host ('127.0.0.1', 12000) replied: Seq 8 12/10/2021 20:42:44, RTT = 46.20 ms
Ping 9: host ('127.0.0.1', 12000) replied: Seq 9 12/10/2021 20:42:44, RTT = 46.04 ms
Ping 10: host ('127.0.0.1', 12000) replied: Seq 10 12/10/2021 20:42:44, RTT = 31.69 ms

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
MinRTT: 16.03 ms
MaxRTT: 47.67 ms
AverageRTT: 34.85 ms
PacketLoss: 0.00%
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3) *Code:*

```
# udppingserver_no_loss.py
from socket import *
import random
from time import sleep
# Create a UDP socket
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Assign IP address and port number to socket
serverSocket.bind(('localhost', 12000))
while True:
    # Receive the client packet along with the address it is coming from
    message, address = serverSocket.recvfrom(1024)
    val = random.randint(10, 40)
    sleepVal = val/1000
    sleep(sleepVal)
    # # The server responds
    serverSocket.sendto(message, address)
```

## Part 3 - UDP Pinger with Delays and Packet Losses

1) The operation of the ping server is identical to the process from part 2; except for that I included a random number generator from 0 to 100 that returns a number which is then checked in an if statement to see if it is below 20. If it returns true then the server sleeps exactly 1 second which causes a timeout on the client side (packet lost).
2) Navigate to the appropriate directory using cmd. With python installed on your system simply type "**py jp3.py**" or "**python jp3.py**" to launch the client side depending on the version of python and system. When done, close the terminal windows to stop the execution. *NOTE: before launching the client side run the server side first for safe measures using the same format .*



```
Ping 86: host ('127.0.0.1', 12000) replied: Seq 86 12/10/2021 20:44:36, RTT = 47.73 ms
Ping 87: host ('127.0.0.1', 12000) replied: Seq 87 12/10/2021 20:44:37, RTT = 48.02 ms
Ping 88: Request timed out
Ping 89: host ('127.0.0.1', 12000) replied: Seq 89 12/10/2021 20:44:38, RTT = 46.38 ms
Ping 90: host ('127.0.0.1', 12000) replied: Seq 90 12/10/2021 20:44:38, RTT = 15.49 ms
Ping 91: host ('127.0.0.1', 12000) replied: Seq 91 12/10/2021 20:44:38, RTT = 47.00 ms
Ping 92: host ('127.0.0.1', 12000) replied: Seq 92 12/10/2021 20:44:38, RTT = 47.09 ms
Ping 93: host ('127.0.0.1', 12000) replied: Seq 93 12/10/2021 20:44:38, RTT = 32.02 ms
Ping 94: host ('127.0.0.1', 12000) replied: Seq 94 12/10/2021 20:44:38, RTT = 31.29 ms
Ping 95: host ('127.0.0.1', 12000) replied: Seq 95 12/10/2021 20:44:38, RTT = 45.69 ms
Ping 96: host ('127.0.0.1', 12000) replied: Seq 96 12/10/2021 20:44:38, RTT = 31.55 ms
Ping 97: host ('127.0.0.1', 12000) replied: Seq 97 12/10/2021 20:44:38, RTT = 46.63 ms
Ping 98: host ('127.0.0.1', 12000) replied: Seq 98 12/10/2021 20:44:38, RTT = 46.76 ms
Ping 99: host ('127.0.0.1', 12000) replied: Seq 99 12/10/2021 20:44:38, RTT = 46.68 ms
Ping 100: Request timed out

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
MinRTT: 15.31 ms
MaxRTT: 48.02 ms
AverageRTT: 27.60 ms
PacketLoss: 18.00%
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
PS D:\471 p2\part 1>
```

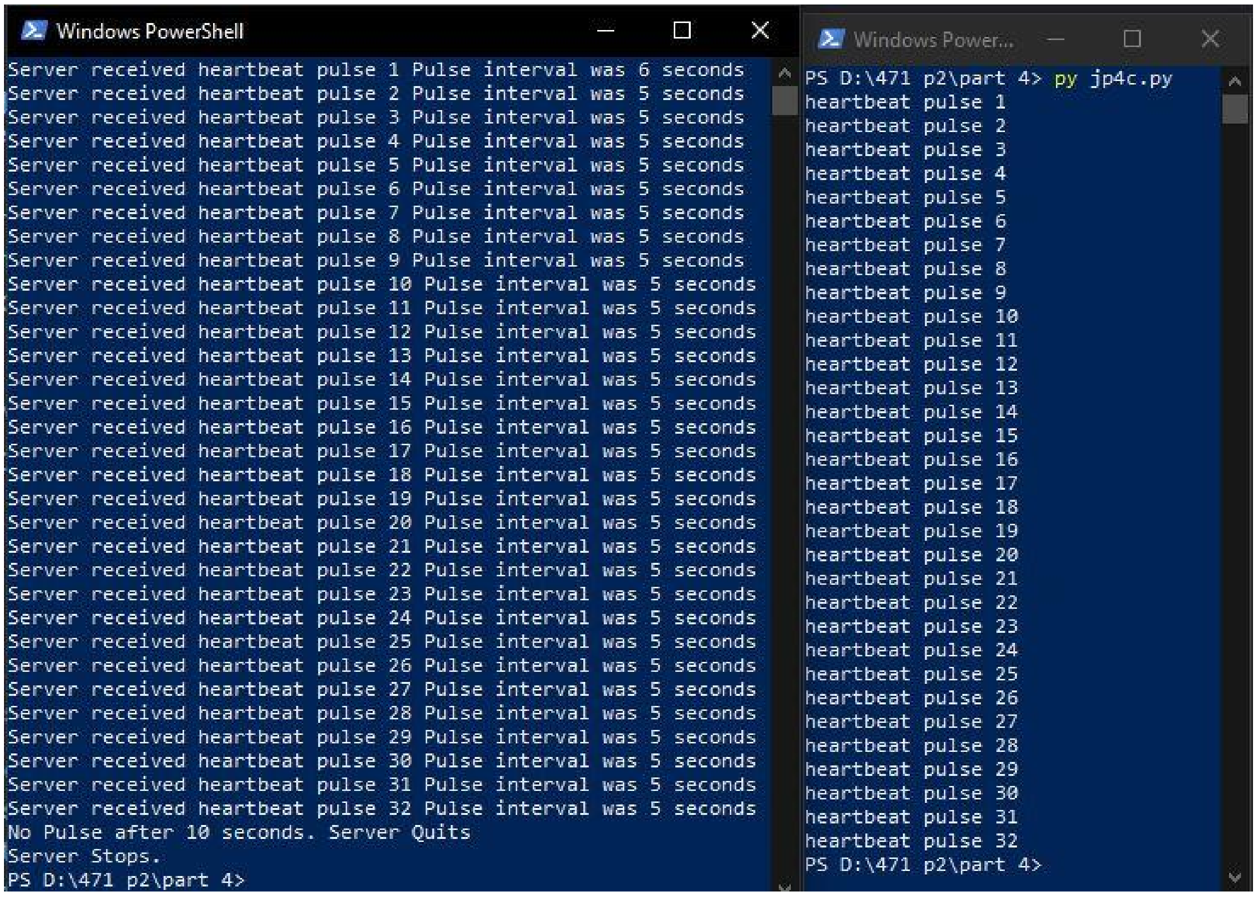
3) *Code:*

```
# udppingserver_no_loss.py
from socket import *
import random
from time import sleep
# Create a UDP socket
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Assign IP address and port number to socket
serverSocket.bind(('localhost', 12000))
while True:
    # Receive the client packet along with the address it is coming from
    message, address = serverSocket.recvfrom(1024)
    val = random.randint(10, 40)
    drop = random.randint(0, 100)
    if drop < 20:
        sleep(1)
    else:
        sleepVal = val/1000
        sleep(sleepVal)
```

```
# The server responds
serverSocket.sendto(message, address)
```

## Part 4 - UDP Heartbeat Monitor

***1)*** Navigate to the appropriate directory using cmd. With python installed on your system simply type “**py jp4s.py**” or “**python jp4s.py**” to launch the server side depending on the version of python and system. Repeat this step in a new terminal however and type “**py jp4c.py**” or “**python jp4c.py**” to launch the client side. When the client times out it will close automatically and so will the server. *NOTE: before launching the client side run the server side first for safe measures.*



```
Server received heartbeat pulse 1 Pulse interval was 6 seconds
Server received heartbeat pulse 2 Pulse interval was 5 seconds
Server received heartbeat pulse 3 Pulse interval was 5 seconds
Server received heartbeat pulse 4 Pulse interval was 5 seconds
Server received heartbeat pulse 5 Pulse interval was 5 seconds
Server received heartbeat pulse 6 Pulse interval was 5 seconds
Server received heartbeat pulse 7 Pulse interval was 5 seconds
Server received heartbeat pulse 8 Pulse interval was 5 seconds
Server received heartbeat pulse 9 Pulse interval was 5 seconds
Server received heartbeat pulse 10 Pulse interval was 5 seconds
Server received heartbeat pulse 11 Pulse interval was 5 seconds
Server received heartbeat pulse 12 Pulse interval was 5 seconds
Server received heartbeat pulse 13 Pulse interval was 5 seconds
Server received heartbeat pulse 14 Pulse interval was 5 seconds
Server received heartbeat pulse 15 Pulse interval was 5 seconds
Server received heartbeat pulse 16 Pulse interval was 5 seconds
Server received heartbeat pulse 17 Pulse interval was 5 seconds
Server received heartbeat pulse 18 Pulse interval was 5 seconds
Server received heartbeat pulse 19 Pulse interval was 5 seconds
Server received heartbeat pulse 20 Pulse interval was 5 seconds
Server received heartbeat pulse 21 Pulse interval was 5 seconds
Server received heartbeat pulse 22 Pulse interval was 5 seconds
Server received heartbeat pulse 23 Pulse interval was 5 seconds
Server received heartbeat pulse 24 Pulse interval was 5 seconds
Server received heartbeat pulse 25 Pulse interval was 5 seconds
Server received heartbeat pulse 26 Pulse interval was 5 seconds
Server received heartbeat pulse 27 Pulse interval was 5 seconds
Server received heartbeat pulse 28 Pulse interval was 5 seconds
Server received heartbeat pulse 29 Pulse interval was 5 seconds
Server received heartbeat pulse 30 Pulse interval was 5 seconds
Server received heartbeat pulse 31 Pulse interval was 5 seconds
Server received heartbeat pulse 32 Pulse interval was 5 seconds
No Pulse after 10 seconds. Server Quits
Server Stops.
PS D:\471 p2\part 4>
```

```
PS D:\471 p2\part 4> py jp4c.py
heartbeat pulse 1
heartbeat pulse 2
heartbeat pulse 3
heartbeat pulse 4
heartbeat pulse 5
heartbeat pulse 6
heartbeat pulse 7
heartbeat pulse 8
heartbeat pulse 9
heartbeat pulse 10
heartbeat pulse 11
heartbeat pulse 12
heartbeat pulse 13
heartbeat pulse 14
heartbeat pulse 15
heartbeat pulse 16
heartbeat pulse 17
heartbeat pulse 18
heartbeat pulse 19
heartbeat pulse 20
heartbeat pulse 21
heartbeat pulse 22
heartbeat pulse 23
heartbeat pulse 24
heartbeat pulse 25
heartbeat pulse 26
heartbeat pulse 27
heartbeat pulse 28
heartbeat pulse 29
heartbeat pulse 30
heartbeat pulse 31
heartbeat pulse 32
PS D:\471 p2\part 4>
```

*2)*

*3) Code:*

***a) Client Code:***

```
from socket import *
from datetime import datetime
from time import sleep
import time
import random

num = 0

# socket Info
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.connect((serverName, serverPort))

while True:
    val = random.randint(0, 100)
    if val < 5:
        break
    else:
        sleepVal = 5  # send after 5 seconds
        num = num + 1
        message = "heartbeat pulse " + str(num)
        clientSocket.sendto(message.encode(), (serverName, serverPort))
        clientSocket.settimeout(1.00000)
        try:
            print(message)
        except:
            print("Request timed out")
    sleep(sleepVal)

clientSocket.close()
```

***b) Server Code:***

```
# udppingserver_no_loss.py
from socket import *
from time import sleep
import time
# Create a UDP socket
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Assign IP address and port number to socket
serverSocket.bind(('localhost', 12000))
```

```
num = 0

while True:
    start = time.time()
    # Receive the client packet along with the address it is coming from
    serverSocket.settimeout(10.00000)
    try:
        message, address = serverSocket.recvfrom(1024)
        end = time.time()
        timeVal = end - start
        timeVal = int(timeVal)
        print("Server received " + message.decode() +
              " Pulse interval was " + str(timeVal) + " seconds")
    except:
        print("No Pulse after 10 seconds. Server Quits")
        break
print("Server Stops.")
serverSocket.close()
```