Homework 3
UDP Client Pinger

PROGRAM SPECIFICATIONS

You will learn the basics of socket programming for UDP in Python. You will learn how to send and receive datagram packets using UDP sockets and also, how to set a proper socket timeout. Throughout the lab, you will gain familiarity with a Ping application and its usefulness in computing statistics such as packet loss rate.

You will first study a simple Internet ping server written in the Python and implement a corresponding client. The functionality provided by these programs is similar to the functionality provided by standard ping programs available in modern operating systems. These programs use a simpler protocol, UDP, rather than the standard Internet Control Message Protocol (ICMP) to communicate with each other. The ping protocol allows a client machine to send a packet of data to a remote machine, and have the remote machine return the data back to the client unchanged (an action referred to as echoing). Among other uses, the ping protocol allows hosts to determine round-trip times to other machines.

CODE

Below you will find the skeleton code for the UDP Ping server. I have also provided this code in the file UDPPingServer.py. You will not modify this code.

RUNNING THE SERVER

To run the UDP Ping Server

1) Open a command line terminal
2) Change directories to where UDPPingServer.py is stored
3) Run the server with python UDPPingServer.py

NOTE: The IP address of the UDP Ping Server is set to 127.0.0.1 and the port should be 9876.

THE SERVER CODE

```
# UDPPingServer.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(('127.0.0.1', 9876))

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 > 3:
		serverSocket.sendto(message, address)
```

WRITING THE CLIENT CODE

You will need to write a UDP Ping Client program named UDPPingClient.py. I have provided a skeleton of code. NOTE: This code accepts command line parameters.

```
import sys, time
from socket import *

# Get the server hostname and port as command line arguments
argv = sys.argv
host = argv[1]
port = argv[2]
timeout = 1 # in second

# Create UDP client socket
clientsocket = socket( ) #<- Code goes in the parenthesis

# Set socket timeout as 1 second
#Code start           #Code Stop

# Command line argument is a string, change the port into integer
port = int(port)

# Sequence number of the ping message
ptime = 0

# Using ptime ping for 10 times
while #Code start           #Code Stop
      ptime += 1

      # Format the message to be sent
      data = "Ping " + str(ptime) + " " + time.asctime()

      try:
      # Sent time
            RTTb = time.time()

      # Send the UDP packet with the ping message
            #Code start           #Code Stop


      # Receive the server response
            message, address =#Code start           #Code Stop

      # Received time
            RTTa = time.time()

      # Display the server response as an output. This is a print statement
            #Code start           #Code Stop

      # Round trip time is the difference between sent and received time in a print stmt
            #Code start           #Code Stop

      except:
            # Server does not response
            # Assume the packet is lost
            print ("Request timed out.")
            continue

# Close the client socket
Code start           #Code Stop
```

TO COMPLETE

1) Complete the Python code that creates a functioning UDP Client Ping program. You can copy and paste, or you can type it. My recommendation is to type it is easier then copy and pasting and trying to fix code.
   a. Most of the code is socket code. There are a few lines that are simple print statements.
   b. See the sample output below.
      i. Your output will NOT be the same as mine
      ii. You are looking at the printed messages.
      iii. Your messages will be very similar to mine
2) Run the server
3) Execute the client - capture the output of the client
4) Execute the client at least 2 more times capturing the output for each run

RUNNING THE CLIENT

To run the client

1) Ensure the UDPPingServer.py is running
2) Open a command line terminal
3) Change directories to where UDPPingClient.py is stored
4) Run the server with python UDPPingClient.py 127.0.0.1 9876

TO TURN IN

A zip file that contains

1) A PDF of your Python code and your runs
2) Your both UDPPingClient.py and UDPPingServer.py file
3) At three captures from different runs of your program

Name your zip with your last name first letter of your first name hw3.zip (Example: steinershw3.zip)

SAMPLE OUTPUT

```
$python3 ./UDPClient.py 127.0.0.1 9876
Request timed out.
Reply from 127.0.0.1: PING 2 TUE APR 27 16:49:25 2021
RTT: 0.0001437664031982422
Request timed out.
Request timed out.
Reply from 127.0.0.1: PING 5 TUE APR 27 16:49:27 2021
RTT: 0.00013208389282226562
Reply from 127.0.0.1: PING 6 TUE APR 27 16:49:27 2021
RTT: 9.703636169433594e-05
Reply from 127.0.0.1: PING 7 TUE APR 27 16:49:27 2021
RTT: 6.198883056640625e-05
Reply from 127.0.0.1: PING 8 TUE APR 27 16:49:27 2021
RTT: 5.1021575927734375e-05
Request timed out.
Reply from 127.0.0.1: PING 10 TUE APR 27 16:49:28 2021
RTT: 0.000125885009765625
```