CSUF SPRING 2021
COMPUTER COMMUNICATIONS CPSC 471 - 01/05
PROJECT #2
PROFESSOR: LINH TRINH

PROJECT SUBMISSION AND YOUR RESPONSIBILITY

You are responsible for the content of your submitted zip file. You should double-check to ensure your submitted zip file contain all required files and ensure their contents are what you intended to submit. Canvas allows you to submit as many times as you wish. The last submission is the version that will be graded. Grading will be based solely on your submission in Canvas. To ensure fairness for everyone, submissions outside Canvas will not be accepted. Canvas is setup to only accept .zip file type.

Submit **one zip file** using naming convention: **yournamep2.zip** in Canvas. The **yournamep2.zip** file shall contain the following files:

(1) PDF report file using naming convention **yournamep2.pdf**
This must be a PDF document file type, not Word file, or any other file formats.
Any file type other than PDF will not be accepted and will get zero credit for the report.
(2) Python UDP client code file **xxp21c.py**
(3) Python Heartbeat client code file **xxp22c.py**
(4) Python Heartbeat server code file **xxp22s.py**

where xx = initials (the two characters representing the first character of first and last name).

PART 1: UDP PINGER USING PYTHON

In this project, 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, 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 like the functionality provided by standard ping programs available in modern operating systems. However, 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.

You are given the complete code for the Ping server in the next subsection. Your task is to write the UDP Ping client.

SERVER CODE

The following code fully implements a ping server. You need to run this code before running your client program. **Do not modify this code**. Your client code will be graded using this server code.

In this server code, 30% of the client's packets are simulated to be lost. You should study this code carefully, as it will help you write your ping client 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(('', 45678))
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)
    # If rand is less is than 4, we consider the packet lost, do not respond
    if rand < 4:
        continue
    # Otherwise, the server responds
    serverSocket.sendto(message, address)
```

The server sits in an infinite loop listening for incoming UDP packets. When a packet comes in and if a randomized integer is greater than or equal to 4, the server simply sends it back to the client.

PACKET LOSS INJECTION

UDP provides applications with an unreliable transport service. Messages may get lost in the network due to router queue overflows, faulty hardware or some other reasons. Because packet loss is rare or even non-existent in typical campus or home networks, the server in this project

injects artificial loss to simulate the effects of network packet loss. The server creates a variable randomized integer which determines whether a particular incoming packet is lost or not.

CLIENT CODE

You task is to implement the client program as explained below.

The client should send a specified number of pings to the server. Because UDP is an unreliable protocol, a packet sent from the client to the server may be lost in the network, or vice versa. For this reason, the client cannot wait indefinitely for a reply to a ping message. You should get the client wait up to one second for a reply; if no reply is received within one second, your client program should assume that the packet was lost during transmission across the network. You will need to look up the Python documentation to find out how to set the timeout value on a datagram socket.

CLIENT CODE REQUIREMENTS

The client program should:

(1) send the ping message using UDP (Note: Unlike TCP, you do not need to establish a connection first, since UDP is a connectionless protocol.)
(2) print the response message from server, if any was received
(3) calculate and print the round-trip time (RTT), in seconds, of each packet, if the server responses
(4) otherwise, print "Request timed out"
(5) provide a summary report at the end of all pings which includes:
    a. minimum RTT in milliseconds,
    b. maximum RTT in milliseconds,
    c. average RTT in milliseconds,
    d. percentage packet loss rate

You should run the udppingserver.py on your machine and test your client by sending packets to the localhost.

PING MESSAGE FORMAT

The ping message in this lab is formatted in a simple way. The client ping message is a one line, consisting of ASCII characters in the following format:

```
seq sequence_number date_and_time
```

where sequence_number starts at 1 and progresses to total number of pings for each successive ping message sent by the client, and time is the time when the client sends the message.

Refer to the Appendix section on the last page for sample.

WHAT TO HAND IN

PDF FILE REPORT

Create a section called **Part 1 – UDP Pinger**. Include the followings:

(1) Instructions on how to run the code, i.e., command line and any applicable parameter(s)
(2) Run-time screen captures:
    a. A sequence consists of 10 pings
    b. A sequence consists of 15 pings.

    Refer to the Appendix section on the last page for sample.

(3) Python code listing:
    a. Include as text the listing of your Python code.
    Please use consolas font size 10 or equivalent monospace font. The use of these monospace font is to clearly show indentations in your code.

SUBMISSION ZIP FILE

include your Python client code file **xxp21c.py**

where xx = initials (the two characters representing the first character of first and last name).

PART 2: HEARTBEAT MONITOR USING PYTHON

Another similar application to the UDP Ping would be the UDP Heartbeat. The Heartbeat can be used to check if an application is up and running on the client side and to report one-way packet loss. The client continuously sends a message acting as a heartbeat in the UDP packet to the server, which is monitoring the heartbeat (i.e., the UDP packets) of the client. Upon receiving the packets, the server calculates the time difference. If the heartbeat packets are missing for some specified time interval, the server can assume that the client application has stopped working.

Implement the UDP Heartbeat (both client and server). You will need to modify the given udppingserver.py, and your udppingclient.py.

Use the following file naming convention:

- **xxp22c.py** for client side
- **xxp22s.py** for server side

where xx = initials (the two characters representing the first character of first and last name).

The client program sends a ping message to the server using UDP every 5 seconds.

The server program monitors if a ping is received from the client. If the ping from the client was absent for more than 10 seconds, it prints the message “No pulse after 10 seconds. Server quits”.

WHAT TO HAND IN

PDF FILE REPORT

Create a section called **Part 2 – UDP Heartbeat Monitor**. This portion of your report includes:

(1) Instructions on how to run the code, i.e., command line and any applicable parameter(s) for the client and the server programs

(2) Run-time screen capture showing:
   a. the client sends heartbeat pings to the server every 5 seconds.
   b. server prints the received heartbeat pings from the client, and the time interval.
   c. server detects absence of client heartbeat and quits.

   Refer to the Appendix section on the last page for sample.

(3) Python code listing:
   a. Include as text, the client program listing.
   b. Include as text, the server program listing.

Please use consolas font size 10 or equivalent monospace fonts. The use of these monospace font is to clearly show indentations in your code.

SUBMISSION ZIP FILE

include both your Python client and server code files **xxp22c.py** and **xxp22s.py**

where xx = initials (the two characters representing the first character of first and last name).

APPENDIX

UDP PINGER RUN-TIME SAMPLE

SEQUENCE OF 10 PINGS

```
Ping 1: host 127.0.0.1 replied: seq 1 Fri Mar 12 18:20:29 2021, RTT = 0.50 ms
Ping 2: host 127.0.0.1 replied: seq 2 Fri Mar 12 18:20:29 2021, RTT = 0.07 ms
Ping 3: timed out, message was lost
Ping 4: host 127.0.0.1 replied: seq 4 Fri Mar 12 18:20:30 2021, RTT = 0.48 ms
Ping 5: host 127.0.0.1 replied: seq 5 Fri Mar 12 18:20:30 2021, RTT = 0.36 ms
Ping 6: host 127.0.0.1 replied: seq 6 Fri Mar 12 18:20:30 2021, RTT = 0.23 ms
Ping 7: host 127.0.0.1 replied: seq 7 Fri Mar 12 18:20:30 2021, RTT = 0.18 ms
Ping 8: host 127.0.0.1 replied: seq 8 Fri Mar 12 18:20:30 2021, RTT = 0.13 ms
Ping 9: host 127.0.0.1 replied: seq 9 Fri Mar 12 18:20:30 2021, RTT = 0.13 ms
Ping 10: timed out, message was lost
Min RTT = 0.07 ms
Max RTT = 0.50 ms
Avg RTT = 0.26 ms
Packet lost = 20.00 %
```

SEQUENCE OF 15 PINGS

```
Ping 1: host 127.0.0.1 replied: seq 1 Fri Mar 12 18:17:46 2021, RTT = 0.36 ms
Ping 2: host 127.0.0.1 replied: seq 2 Fri Mar 12 18:17:46 2021, RTT = 0.06 ms
Ping 3: host 127.0.0.1 replied: seq 3 Fri Mar 12 18:17:46 2021, RTT = 0.06 ms
Ping 4: host 127.0.0.1 replied: seq 4 Fri Mar 12 18:17:46 2021, RTT = 0.06 ms
Ping 5: host 127.0.0.1 replied: seq 5 Fri Mar 12 18:17:46 2021, RTT = 0.06 ms
Ping 6: host 127.0.0.1 replied: seq 6 Fri Mar 12 18:17:46 2021, RTT = 0.05 ms
Ping 7: host 127.0.0.1 replied: seq 7 Fri Mar 12 18:17:46 2021, RTT = 0.05 ms
Ping 8: host 127.0.0.1 replied: seq 8 Fri Mar 12 18:17:46 2021, RTT = 0.05 ms
Ping 9: host 127.0.0.1 replied: seq 9 Fri Mar 12 18:17:46 2021, RTT = 0.05 ms
Ping 10: host 127.0.0.1 replied: seq 10 Fri Mar 12 18:17:46 2021, RTT = 0.05 ms
Ping 11: timed out, message was lost
Ping 12: timed out, message was lost
Ping 13: timed out, message was lost
Ping 14: host 127.0.0.1 replied: seq 14 Fri Mar 12 18:17:49 2021, RTT = 0.53 ms
Ping 15: timed out, message was lost
Min RTT = 0.05 ms
Max RTT = 0.53 ms
Avg RTT = 0.13 ms
Packet lost = 26.67 %
```

HEARTBEAT MONITOR RUN-TIME SAMPLES

CLIENT SIDE

```
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
```

SERVER SIDE

```
Server received heartbeat pulse 3 Pulse interval was 6 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
No pulse after 10 seconds. Server quits
Server stops.
```

End of Project 2