UTS Insearch
UNIVERSITY OF TECHNOLOGY SYDNEY

IT

PROJECT B

Diploma of IT

Network Fundamentals

Subject Coordinator: Claire Pierse

Weight: 10%

DUE: Week 9 , Saturday

PROJECT B: UDP PINGER ASSIGNMENT

TASK DESCRIPTION

In this lab, 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.

OBJECTIVES

You will first study a simple Internet ping server written in 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. 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 (RTT) to other machines.

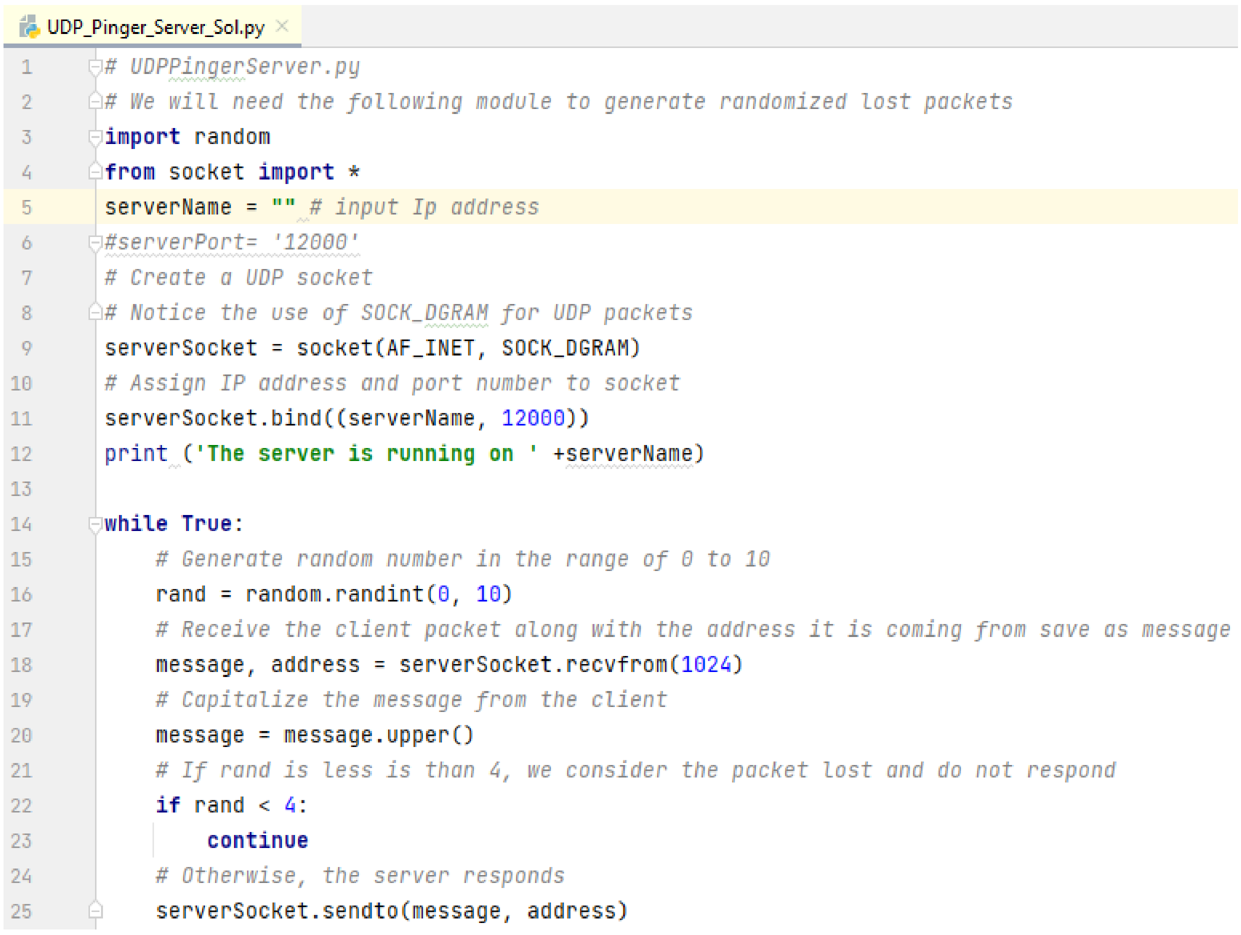
BRIEF

CODE

You are given the complete code for the Ping server. ***Your task is to write the Ping client.***

SERVER CODE

The following code fully implements a ping server. You need to compile and run this code before running your client program. **You do not need to modify this 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.



```python
# UDPPingerServer.py
# We will need the following module to generate randomized lost packets
import random
from socket import *
serverName = "" # input Ip address
#serverPort= '12000'
# 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((serverName, 12000))
print ('The server is running on ' +serverName)

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 save as message
    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)
```

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

Packet Loss

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 networks, the server in this lab injects artificial loss to simulate the effects of network packet loss. The server creates a variable randomised integer that determines whether a particular incoming packet is lost or not.

Client Code

You need to implement the following client program.

The *client should send 10 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 to 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* set the timeout value *on a datagram socket with this code*

clientsocket.settimeout(1)

Specifically, your 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 the server, if any

(3) calculate and print the round-trip time (RTT), in seconds, of each packet, if server responses

(4) otherwise, print “Request timed out”

During development, you should run the UDPPingerServer.py on your machine, and test your client by sending packets to localhost (or, 127.0.0.1). After you have fully debugged your code, you should see how your application communicates across the network with the ping server and ping client running on different machines.

Message Format

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

Ping sequence_number time

where *sequence_number* starts at 1 and progresses to 10 for each successive ping message sent by the client, and time is the time when the client sends the message.

Currently, the program calculates the round-trip time for each packet and prints it out individually. Modify this to correspond to the way the standard ping program works. *You will need to report the minimum, maximum, and average RTTs at the end of all pings from the client. In addition, calculate the packet loss rate (in percentage).*

SUBMISSION

WHAT TO SUBMIT FOR PROJECT 2:

Upload to the Canvas Project B assignment link

1) Project report. The report should explain the UDP Pinger Server python code which was supplied and, and describe the code you have written for the UDP client together with the troubleshooting required to get the program running. The suggested word count is 400-500 words, the report can be in .docx or .pdf format.

2) THE UDPPinger client code. Submit the relevant python file. (.py) make sure to include your name and student number in the filename.

3) Video - In addition to the written submission, you will create a short video using Canvas Studio. Your video should verify that your program can :
   a) send the ping message using UDP
   b) print the response message from server, if any

c) calculate and print the round-trip time (RTT), in seconds, of each packet, if server responses
d) otherwise, print “Request timed out”

EXTENSION CHALLENGE - TO ACHIEVE “HD“GRADE

1. 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 and to report one-way packet loss. The client sends a sequence number and current timestamp in the UDP packet to the server, which is listening for the Heartbeat (i.e., the UDP packets) of the client. Upon receiving the packets, the server calculates the time difference and reports any lost packets. If the Heartbeat packets are missing for some specified period of time, we can assume that the client application has stopped.

2. Implement the UDP Heartbeat (both client and server). You will need to modify the given `UDPPingerServer.py,` and your UDP ping client.

ASSESSMENT CRITERIA

CRITERIA	WEIGHT	SLOs	PLOs
System Design	**20**	**2**	**B2**
Develop and implement designs	**40**	**4**	C2
Test Solutions	**40**	**3**	B2

SLOs: subject learning outcomes
PLOs: program learning outcomes