Computer Networks and Programming (ECE 5650)

Project 1

Winter 2017

Deadline: Tuesday, February 14 by 11:59 PM. No late submissions will be accepted.

Instructions:

- **Team Work**: you may work individually or with one other student. **If you work with another student, you must submit only one copy with both your names.**
- Your program must be in Python 2.7.x and must use socket programming for all networking activities.
- If you have questions, please contact the **GTA** during his office hours. You may also contact him by e-mail if you cannot make it to his office hours. His contact information is included in the online syllabus.
- **You must submit all the following files**:
  - Python program(s) in .py extension
  - **Professional report in pdf format, including a copy of each source code, testing procedure used to verify the correctness of the program, and the screenshots and their explanations. Make sure that you include your name(s) on the report.**
- **File Naming:** the filenames must contain your last name(s): yourlastname.properExtension (if you work alone) or yourlastname1-yourlastname2.properExtension (if you work with another student).
- **Required details in the beginning of the program(s):** The beginning of the python program must include the team names and must indicate whether your program is interpreted correctly and whether it produces the correct results. If it does not produce the correct results, you must provide the necessary details.
- **Comments and Documentation:** the program must be well documented and commented.
- Do NOT put "#" in the file name(s).
- Upload your file(s) using to this Assignment page. Make sure that you attach the files and hit the "Submit" button.
- **To verify your submission**, go to grading center and make sure that there is "!" in the entry for this assignment.
- **Assessed Penalties:**

Situation	Penalty
Late submission	No accepted
Plagiarism or disallowed collaboration	At minimum negative grade
Not using socket programming for all networking	Not accepted
Not using Python 2.7.x	Not accepted
Failure to include the full report with **thoughtful screenshots** verifying the program(s) work correctly	Up to 10%
Failure to include all required information at the beginning of the program(s)	Up to 10%
Failure to have a well commented and documented program(s)	Up to 10%

Policy on Cheating, Fabrication, and Plagiarism:

Cheating, fabrication, plagiarism, and helping others to commit these acts are all forms of academic dishonesty, and they are wrong. Academic misconduct will result in at least failing the course. Therefore, avoid all appearance of improper behavior! Students who witness cheating should report the incident to the instructor as soon as possible.

Assignment

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 project, 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 then 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 to other machines.

You are given the complete code for the Ping server below. 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, 40% 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.

```
# 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))


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 than 5, packet is considered lost and do not respond
if rand < 5:
    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 capitalizes 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 randomized integer, which determines whether a particular incoming packet is lost or not.

Client Code
You are required to implement the following client program. The client should send 12 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.

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 server, if any,
3. calculate and print the round trip time (RTT), in seconds, of each packet, if the server responds; otherwise, print "Request timed out", and
4. report the minimum, maximum, median, and mean RTTs at the end of all pings from the client.

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 must be formatted as follows. The client message is one line, consisting of ASCII characters in the following format:

Ping time sequence_number

where sequence_number starts at 1 and progresses to 12 for each successive ping message sent by the client, and time is the time when the client sends the message.

What to Submit

- You must submit the complete UDP Pinger python code.
- You must also submit a **detailed report,** including a copy of the source code, testing procedure used to verify the correctness of the program, and the screenshots and their explanations. The screenshots of the command lines of both the client and server must be included, verifying that your code runs correctly. You must provide **thoughtful screenshots** to prove that your code works as expected.

Good Luck!