CS2505 – Python Lab 05
21.03.2019

Let us start with the **Important** stuff first:

1. The grading for these Labs will take into account the **correctness** of your solution, the **approach** taken, and **comments**, which should be clear and concise. We will be checking carefully for plagiarism and penalties will be strictly applied.
2. If you don't understand a question, please ask Yusuf or Ahmed, the lab demonstrators, who will be only happy to help.
3. All Labs will consist of some Python programming and some questions. To maximise your Continuous Assessments marks, please answers all sections.
4. The Continuous Assessment labs are worth 20% of your final year mark for this module. Thus, each of the five labs is graded out of maximum of 4 marks. To offer students an additional means of maximising their Continuous Assessments marks, each week we will also offer an additional *Coding Assignment*, from which you can receive a maximum of 1 additional mark. This extra assignment is optional and will not affect your ability to receive the maximum 4 marks for the initial part of this lab. Note: irrespective of how many questions you attempt over the five weeks, the maximum number of marks that can be received from the Continuous Assessment aspect of this module is 20 marks.
5. We do not accept solutions which are written in Python 2 (https://pythonclock.org/). **Make sure your solutions work in Python 3**.

Your solutions for this Lab, including the solutions for the additional assignment should you decide to attempt same, must be submitted on Moodle within the specified deadline. Please note that no late submission will be accepted by Moodle. If your solution files cannot run successfully, you will lose marks. So, make sure that there is no **syntax, compilation or run-time error**. You do not need to include your name or UCC ID in the name of the submitted files (Moodle does that automatically). **Follow the file naming conventions** as mentioned in the description of assignment. ☺
We recommend (but not obligate) that you follow the official style guide for Python:
https://www.python.org/dev/peps/pep-0008/
The official Python 3.7 documentation is located here: https://docs.python.org/3.7/index.html

---

Where to get Python 3.7+?
In order to run Python 3.7 on machine in class room, type python3 (or python3.7 to be 100% sure) in Kubuntu or python in Windows. You can run your script by typing "python3.7 script.py" in Kubuntu or "python script.py" in Windows.
If you use your own machine:
- for Windows OS, go to python.org and download Python 3.7 or higher;
- Ubuntu usually has the latest Python installed by default
- for older versions of Ubuntu (for 14.04, for example) you get the latest Python version from https://launchpad.net/~fkrull/+archive/ubuntu/deadsnakes ppa:

```
sudo add-apt-repository ppa:fkrull/deadsnakes
sudo apt-get update
sudo apt-get install python3.7
```

- another very good solution is usage of conda package system(https://conda.io/docs/). Go to https://conda.io/docs/install/quick.html and read manual for your OS. With conda you can quickly install a package by typing “conda install numpy” and quickly update all installed packages by typing “conda upgrade –all” (upgrade is alias for update here).

Lab 5:

In this week’s lab, we are going to implement ‘ping’ using UDP python Sockets.

UDP Sockets:

Create a new project folder called “CS2505_lab5”, and download “UDP_ping_Server.py” from the module webpage. You will create the client python file. Look to previous Labs for details on running your client and server.

In this lab you will implement ping in python. Throughout the lab, you will gain familiarity with a Ping application and its usefulness in computing statistics such as packet loss rate.

You will 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 to other machines.

In the 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. *You do not need to modify the server code.*

Client Code:

You need to implement the following client program.

The client should send a certain 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 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 look up the Python documentation to find out how to set the timeout value on a datagram socket.

Specifically, your client program should:

1. Take command-line arguments specifying the number of pings to send and the IP address of the server. The following is an example input command format to run the client.

   ```
   UDP_ping_Client.py 10 127.0.0.1
   ```

   Where 10 is the number of pings to send and 127.0.0.1 is the address at which the ping server is running.
2. Send the ping message using UDP in the format explained below. (Note: Unlike TCP, you do not need to establish a connection first, since UDP is a connectionless protocol.)
3. Calculate the round-trip time (RTT), in milliseconds, of each packet, if server responses. Otherwise send the next ping message (after waiting for a second).
4. Calculate and report the minimum, maximum, standard deviation and average of RTTs at the end of all pings from the client. Also, calculate the packet loss rate (in percentage).
5. Print the output in the format of the standard ping program as explained below.

Message Format:
The messages in this lab are formatted in a simple way. The message that the client should send to the server is one line, consisting of ASCII characters in the following format:

Ping *sequence_number time*

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

Output Format:
To view what the output should look like, run 'ping' on your machine. Pass the IP of another machine and view the statistics offered by ping and replicate this in your python code output, e.g. (note the '–c 3' forces ping to only run for 3 packets):

```
ping -c 3 10.0.1.12
PING 10.0.1.12 (10.0.1.12): 56 data bytes
64 bytes from 10.0.1.12: icmp_seq=0 ttl=64 time=0.550 ms
Request timed out.
64 bytes from 10.0.1.12: icmp_seq=2 ttl=64 time=0.347 ms

--- 10.0.1.12 ping statistics ---
3 packets transmitted, 2 packets received, 33.33% packet loss
round-trip min/avg/max/stddev = 0.347/0.483/0.552/0.096 ms
```

64 bytes is the size of the message received from the server. *icmp_seq*, in our case, can be replaced by *udp_seq* and represents the *sequence_number* of the ping message. You can exclude the *ttl* (Time to live) field from your output format. Time is the RTT in milliseconds. If a response is not received from the server, a "Request timed out." message is displayed.

During development, you should run the UDP_ping_Server.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.

To Do:

1. Write the python file, UDP_ping_Client as outlined above. Include for exception control in your code and remember to close all sockets as required. Submit your implementation as UDP_ping_Client.py file.
2. In this lab we looked at ping and the unreliable delivery of UDP packets (packets are simply dropped). In your lectures you were introduced to reliable data transfer using UDP. Provide a brief overview, one page, including stop-and-wait, pipelining and Go-Back-N, and explain which option you think is best and why. Submit it as Overview.doc file.

Additional Coding Assignment:

There is no additional coding assignment this week.