CS 455/555 Programming Assignment #2

Due Date: **Sunday Oct 9th, 2016 - 11:59pm**

Description:

Ping is a utility for network performance measurement used to test the reachability of a host on an Internet Protocol (IP) network and to measure the round-trip time for messages sent from the originating host to a destination computer.

The goal of this assignment is to help you become familiar with socket programming in Java or Python using UDP sockets. We'll be implementing a simple Internet ping server, and a corresponding client (client-server application) that allows a client to measure the round-trip time (RTT) to a server. The functionality provided by these programs are similar to the standard ping programs available in modern operating systems, except that they use UDP rather than Internet Control Message Protocol (ICMP) to communicate with each other.

Because we're limited to running the programs on-campus where there will be little delay and very little loss, the server will add some artificial delay and ignore some ping requests (for UDP) to simulate packet loss.

PingClient

- The ping client will accept two command-line arguments: hostname, and port.
  - hostname - the name of the server
  - port - the port the server is running on
  - Note that for ODU-CS machines, the port must be between 10001-11000. Also, these ports are only accessible to clients that are on campus.
- If any of the command-line arguments are incorrect, exit after printing an error message of the form ERR - arg x, where x is the argument number.
- The ping client will setup a UDP connection to send each message (datagrams).
- The client will send 10 messages to the server in the following format:

  PING seqno timestamp

  where PING is the word "PING", seqno is the ping sequence number (between 0-9), and timestamp is the time (in milliseconds) that the message was created and sent
  - The client will not send a new ping until the previous ping has been answered, or its timed out after 2 seconds.
- On a single line, the client will print the message that it sends to the server and the round-trip time for the ping or a * if the reply is not received in 2 second.
- After receiving the 10th reply (or after its timeout), the client will print a summary with a format similar to that displayed by the ping Unix command. Example:

  ---- PING Statistics ----
  10 packets transmitted, 7 packets received, 30% packet loss
  round-trip (ms) min/avg/max = 63/190.29/290

  Print the average RTT with up to 2 digits after the decimal.

PingServer

- The ping server is essentially an echo server (whatever data is sent will be returned).
- The ping server will accept 1 required command-line argument: port.
    - port - the port the server is running on
- The ping server will accept 1 optional command-line argument: seed.
    - seed - a long to seed the random number generator for generating delays and lost packets
- If any of the arguments are incorrect, exit after printing an error message of the form ERR - arg x, where x is the argument number.
- After a ping request has been received by the UDP server, the server will determine if it should ignore the packet or respond to it (simulating losses). Default loss rate is 25%.
- If the ping will be responded to (i.e., it is not dropped), the server will first wait for a random amount of time to simulate network delay (average 150 ms each way).
- On a single line, the server will print the IP address and port of the client, the client's ping message, and the server's action.
- The server's action will either be "not sent" if the ping was ignored or "delayed x ms" .
- The ping server will keep running until the user presses Ctrl-C.

Rules:

- Your code should run on the CS department Linux machines.
- As with all projects, you are not permitted to work with anyone else (even students not in the class) - all of the coding and documentation must be your own.
- Your program must compile (if Java/C++) and run on the CS Unix machines.
- You must write neat code and document it well. You will lose points for sloppy programs that contain little or no comments.

Testing:

A large part of your program's grade will be determined by how well it handles a set of inputs. You should test your program rigorously before submitting. Because your programs will be run and tested using a script, you must format your output exactly as I have described or you will lose points.

Since Java and Python use different random number generators, they will produce different results with the same seed. So, I'm providing separate examples for each language.

Java Examples

Example 1

```
java PingClient
Usage: java PingClient hostname port
```

java PingServer
Usage: java PingServer port [seed]

Example 2

java PingClient atria three
ERR - arg 2

java PingServer abc
ERR - arg 1

Example 3

```
atria> java PingServer 10002 123
128.82.4.244:44229> PING 0 1360792326564  ACTION: delayed 297 ms
128.82.4.244:44229> PING 1 1360792326863  ACTION: delayed 182 ms
128.82.4.244:44229> PING 2 1360792327046  ACTION: delayed 262 ms
128.82.4.244:44229> PING 3 1360792327309  ACTION: delayed 21 ms
128.82.4.244:44229> PING 4 1360792327331  ACTION: delayed 173 ms
128.82.4.244:44229> PING 5 1360792327505  ACTION: delayed 44 ms
128.82.4.244:44229> PING 6 1360792327550  ACTION: delayed 19 ms
128.82.4.244:44229> PING 7 1360792327570  ACTION: not sent
128.82.4.244:44229> PING 8 1360792328571  ACTION: not sent
128.82.4.244:44229> PING 9 1360792329573  ACTION: delayed 262 ms

sirius> java PingClient atria 10002
PING 0 1360792326564  RTT: 299 ms
PING 1 1360792326863  RTT: 183 ms
PING 2 1360792327046  RTT: 263 ms
PING 3 1360792327309  RTT: 22 ms
PING 4 1360792327331  RTT: 174 ms
PING 5 1360792327505  RTT: 45 ms
PING 6 1360792327550  RTT: 20 ms
PING 7 1360792327570  RTT: *
PING 8 1360792328571  RTT: *
PING 9 1360792329573  RTT: 263 ms
---- PING Statistics ----
10 packets transmitted, 8 packets received, 20% packet loss
round-trip (ms) min/avg/max = 20/158.62/299
```

Python Examples

Example 1

python PingClient.py
Usage: python PingClient.py hostname port

python PingServer.py
Usage: python PingServer.py port [seed]

Example 2

python PingClient.py atria three

ERR - arg 2

python PingServer.py abc
ERR - arg 1

Example 3

atria> python PingServer.py 10002 123
128.82.4.244:58353> PING 0 1361211436430 ACTION: not sent
128.82.4.244:58353> PING 1 1361211437433 ACTION: not sent
128.82.4.244:58353> PING 2 1361211438435 ACTION: delayed 32 ms
128.82.4.244:58353> PING 3 1361211438468 ACTION: delayed 11 ms
128.82.4.244:58353> PING 4 1361211438480 ACTION: delayed 99 ms
128.82.4.244:58353> PING 5 1361211438580 ACTION: delayed 47 ms
128.82.4.244:58353> PING 6 1361211438628 ACTION: delayed 100 ms
128.82.4.244:58353> PING 7 1361211438729 ACTION: not sent
128.82.4.244:58353> PING 8 1361211439731 ACTION: not sent
128.82.4.244:58353> PING 9 1361211440733 ACTION: delayed 26 ms

sirius> python PingClient.py atria 10002
PING 0 1361211436430 RTT: *
PING 1 1361211437433 RTT: *
PING 2 1361211438435 RTT: 33 ms
PING 3 1361211438468 RTT: 12 ms
PING 4 1361211438480 RTT: 100 ms
PING 5 1361211438580 RTT: 48 ms
PING 6 1361211438628 RTT: 101 ms
PING 7 1361211438729 RTT: *
PING 8 1361211439731 RTT: *
PING 9 1361211440733 RTT: 27 ms
---- PING Statistics ----
10 packets transmitted, 6 packets received, 40.0% packet loss
round-trip (ms) min/avg/max = 12/53.50/101

Notes:

- The delays the server uses should be the same with the same seed, but the timestamps in the client's ping message will not since they depend on the time the program was run.
- The RTTs that the client reports should be similar to the example, but may not be exact.

Submission Materials:

- Make sure your program compiles and executes on Dept's Linux machines.
- Create a "Readme.txt" file for your program that lists how to compile and execute the program. Include your name and your UIN as the first line in the Readme.txt.

- You must name your source programs PingClient.java/PingServer.java, PingClient.py/PingServer.py, or PingClient.cpp/PingServer.cpp.
- For Java and C++, make sure that you submit all files necessary to compile your program. But, do not submit compiled files
- Zip all files of your program and name them prog_assign_2.
- Submit through Blackboard.

Submitting Assignments using Blackboard

Instructions from the official Blackboard 9.1 help pages (with screenshots)

My step-by-step instructions:

- Log in to the course Blackboard page
- Click the **Programming Assignments** link in the left sidebar
- Select the current assignment
- Under **Assignment Submission**, there is an **Attach File** section. Click **Browse My Computer**. Your web browser will open a window in which you can select the file to attach.
- Once you have successfully attached a file, its name should appear in the **Attached files** list.
- You may attach as many files as needed.
- When you've finished adding all of the files that are needed for the assignment, click **Submit** at the bottom of the page. **Important:** If you click Save as Draft instead of Submit, your assignment will not be turned in.
- After you've clicked **Submit**, an assignment receipt page will display. Click **OK**.

Important: *If you submit your assignment and later realize you have made a mistake, Blackboard will allow you to re-submit the assignment. The time of your last attempt will be counted as your submission date. Re-submissions submitted after the assignment deadline are not guaranteed to be graded. If I have started grading your assignment, I will not grade another version submitted after the due date.*

Notes/FAQ

- Where can I find information about Java classes?
  Java Class Reference

- Where can I find information about Python functions?
  Python Standard Library Reference

- Can I use the linux.cs.odu.edu alias when running these programs?
  No. You must use an actual machine name (atria or sirius).

- How can I get the current time in milliseconds?

Java: See System.currentTimeMillis()
Python: See time.time()

- How can I set a timeout value on reading from a datagram socket?
  Java: See the setSoTimeout() function in the DatagramSocket class.
  Python: See the settimeout() function in the socket library.

- How can I format a double to have a certain number of digits after the decimal?
  Java: See the DecimalFormat class.
  Python: See the str.format() function.

- How can I generate a random loss pattern and artificial delay?
  Java:
  - Use the java.util.Random class (so put import java.util.* at the beginning of your program).

  - Between your class PingServer { and public static void main lines, insert:
    ```
    private static final double LOSS_RATE = 0.25;
    private static final int AVERAGE_DELAY = 150; // milliseconds
    ```

  - To set the random number generator seed and create the Random object (after you have determined if the user has given you a seed argument), use
    ```
    // Create random number generator for use in simulating packet loss
    and network delay.
    Random random;
    if (seed == 0) {
        random = new Random();
    } else {
        random = new Random(seed);
    }
    ```
    Note: The Random object should be created only once in the server program. Do not create new Random objects for each connection or for each packet received.

  - To determine whether to reply to a ping, use
    ```
    // Decide whether to reply, or simulate packet loss.
    if (random.nextDouble() < LOSS_RATE)
    ```
    - If this evaluates to true, print 'not sent' and don't send the reply.
    - If this evaluates to false, send the reply.

  - To add delay, use
    ```
    // Simulate network delay.
    delay = (int) (random.nextDouble() * 2 * AVERAGE_DELAY);
    Thread.sleep(delay);
    ```
    Don't forget to print out the delay amount.

  Python:
  - Use random (so put import random at the beginning of your program).

- Create the following global constants:

```
LOSS_RATE = 0.25
AVERAGE_DELAY = 150 # milliseconds
```

- To set the random number generator seed (after you have determined if the user has given you a seed argument):

```
# Create random number generator for use in simulating packet loss
and network delay.
random.seed(seed)
```

  Notes:
  - If seed == 0, the seed will be set based on the current time of day. This will produce different values each time you run the program.
  - random.seed() should be called only once in the server program. Do not reset the seed for each connection or for each packet received.

- To determine whether to reply to a ping, use

```
# Decide whether to reply, or simulate packet loss.
if random.random() < LOSS_RATE
```

  - If this evaluates to true, print 'not sent' and don't send the reply.
  - If this evaluates to false, send the reply.

- To add delay, use

```
# Simulate network delay.
delay = (random.random() * 2 * AVERAGE_DELAY) # sleep time in
ms
time.sleep(delay/1000.0) # accepts time in secs
```

  Don't forget to print out the delay amount.