Assignment III

Tom Kelliher, CS 325

75 points, due Mar. 12

Introduction

(From Kurose & Ross, 4th ed.) For this assignment you will study a simple Internet ping server written in Java and implement a corresponding client. 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. (Java does not provide a straightforward means to interact with ICMP.)

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 (RTTs) to other machines.

You are given the complete code for the ping server on the class web site. Your job is to write the ping client. Study the ping server code carefully, as it will help you write your ping client. Observe that the server sits in an infinite loop listening for incoming UDP packets. When a packet comes in, the server simply sends the encapsulated data back to the client.

Packet Loss

UDP provides applications with an unreliable transport service, because messages may get lost in the network due to router queue overflows or other reasons. In contrast, TCP provides applications with a reliable transport service and takes care of any lost packets by retransmitting them until they are successfully received. Applications using UDP for communication must therefore implement any reliability they need separately in the application level (each application can implement a different policy, according to its specific needs).

Because packet loss is rare or even non-existent in typical campus networks, the server injects artificial loss to simulate the effects of network packet loss. The server has a parameter `LOSS_RATE` that determines which percentage of packets should be lost.

The server also has another parameter `AVERAGE_DELAY` that is used to simulate transmission delay from sending a packet across the Internet. You should set `AVERAGE_DELAY` to a positive value when testing your client and server on the same machine, or when machines are close by on the network. You can set `AVERAGE_DELAY` to 0 to find out the true round trip times of your packets.

Compiling and Running the Server

To compile the server, run the following shell command from the directory into which you copied or saved the server source code:

```
javac PingServer.java
```

To run the server, run the following shell command:

```
java PingServer <port>
```

where `<port>` is the port number the server is to listen on. Remember that you have to pick a port number greater than 1023, because only processes running with root (administrator) privilege can bind to ports less than 1024.

Your Job: The Client Program

You should write the client so that it sends 10 ping requests to the server, separated by approximately one second. Each message contains a payload of data that includes the keyword PING, a sequence number, and a time stamp. After sending each packet, the client waits up to one second to receive a reply. If one second goes by without a reply from the server, then the client assumes that its packet or the server's reply packet has been lost in the network. Hint: Cut and paste `PingServer.java`, rename the code `PingClient.java`, and then modify the code.

You should write the client so that it starts with the following shell command:

```
java PingClient <host> <port>
```

where `<host>` is the FQDN of the computer the server is running on and `port` is the port number it is listening to. Note that you can run the client and server either on different machines or on the same machine.

The client should send 10 pings to the server. Because UDP is an unreliable protocol, some of the packets sent to the server may be lost, some of the packets sent from server to client may be lost, or packets may be received out of order. For this reason, the client can not wait indefinitely for a reply to a ping message. You should have the client wait up to one second for a reply; if no reply is received, then the client should assume that the packet was lost during transmission across the network. You will need to research the API for `DatagramSocket` to find out how to set the timeout value on a datagram socket.

When developing your code, you should run the ping server on the same machine that the client runs on. `kingfisher.goucher.edu`, for example. After you have fully debugged your code, you should see how your application communicates across the network with a ping server run on another machine. `phoenix.goucher.edu`, for example.

Message Format

The ping messages in this lab are formatted in a simple way. Each message contains a sequence of characters terminated by a carriage return character (`'\r'`) and a line feed character (`'\n'`). The message contains the following string:

```
PING <sequence_number> <time> CRLF
```

where `<sequence_number>` starts at 0 and progresses to 9 for each successive ping message sent by the client, `<time>` is the time when the client sent the message, and `CRLF` represent the carriage return and line feed characters that terminate the line.

Client Output

For each message returned within its one second window, print its sequence number and the RTT. For each timed-out packet, prints its sequence number and the string `Timed out`. For each packet received out of sequence, print the sequence number of the received and expected messages and the string `Message received out of order`. Before terminating, your program should report the minimum, maximum and average RTTs.

Administrivia

1. Your client may be implemented in any language currently available on kingfisher, using the libraries currently available on kingfisher. If you implement your client in any language other than C or Java, you are completely on your own — I am only willing to provide assistance for programs written in one of these two languages.

   You should assume that I only know how to compile and run C and Java programs. Therefore, if you implement your client in another language, you are responsible for including complete and concise instructions for compiling and/or running your program, as part of your program's documentation.

2. Your source code must be appropriately documented. This will count as 50% of your grade. Spelling and grammar matter. See `http://people.msoe.edu/~taylor/resources/javadoc.htm` for my expectations.

3. Your client code must be emailed **as an attachment** to `kelliher[at]goucher.edu` by the beginning of class on the due date.

4. Your client code will be tested by running it on kingfisher, with the Java ping server running on phoenix. You should assume that I will set `LOSS_RATE` and `AVERAGE_DELAY` to various fairly nasty values in order to torture your client program. I will be looking for graceful behavior, and evidence of intelligent design decisions, at the limits of the client's performance.

5. I am already aware that a working `PingClient.java` is available on at least one web site. I consider merely viewing any such source code, regardless of implementation language, to be a violation of the Honor Code. In a similar vein, although I encourage you to discuss concepts with each other, I prohibit you from sharing any code with each other. Again, any violation of this policy is a violation of the Honor Code.

Extra Credit

The basic program sends a new ping immediately when it receives a reply. For 10 points of extra credit, modify the program so that it sends exactly 1 ping per second, similar to how the standard ping program works. Hint: Use the `Timer` and `TimerTask` classes in `java.util`. In order to receive the extra credit, your program's documentation must clearly indicate that this feature has been implemented. No extra credit will be given if any of the base program's functionality is missing.