CS 330: Network Applications & Protocols

York College of Pennsylvania
Fall 2017

Programming Assignment - UDP Pinger
Due by 11:59 PM ET date in syllabus

About This Project

This lab comes directly from your textbook, so if you've been reading along you should have no problems following the concepts described within.

In this lab, you will study a simple Internet ping server (PingServer.java) written in the Java language, and implement a corresponding client. The functionality provided by these programs is similar to the standard ping programs available in modern operating systems, except that you will use UDP rather than Internet Control Message Protocol (ICMP) to communicate between the server and the client.

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 zip file provided, you will find the PingServer has already been implemented for you. Your job is to write the PingClient.

Be sure to read and understand the PingServer before attempting to write your PingClient. 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 PingServer in this lab 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.

PingServer 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.

Your Task: Creating PingClient, Part 1

You should write PingClient so that it sends fifteen (15) ping requests to the server, separated by approximately two (2) seconds. Each message contains a payload of data that includes the keyword PING, a sequence number, and a timestamp (use Java's Date). 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.

Your PingClient should take two arguments, a host and a port number where host is the name of the computer where your PingServer is running and the port number is the port number on which it is listening. See the "Compiling and Running" section below to see how to run your program. Note that you can run the client and server either on different machines or on the same machine.

The client should send 15 pings to the server. Because UDP is an unreliable protocol, some of the packets sent to the server may be lost, or some of the packets sent from server to client may be lost. For this reason, the client can not wait indefinitely for a reply to a ping message. You should have the client wait up to two (2) seconds 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 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 a PingServer run by another member of the class.

Your Task: Creating PingClient, Part 2

Extend your PingClient so that you can print out some statistics about the ping messages being sent. Your final code should print a message when a ping reply is received. After ALL ping message have been sent (and possibly all received), you should print out the RTT for each ping message that was sent. If a ping message was lost, be sure to make note of it in your output. Additionally, you should report statistics like the minimum RTT, the maximum RTT, the average RTT, how many replies were received, and how many ping messages were lost. Only include ping replies that were actually received in your computation of the average RTT. An example run from my solution is shown below. Your output should match it closely:

```
> make runClient
java -cp ./bin ycp.cs330.UDPpinger.PingClient 127.0.0.1 8081

Contacting host 127.0.0.1 at port 8081
Received PING 0
Received PING 1
Received PING 2
Received PING 3
Received PING 4
Received PING 7
Received PING 8
Received PING 9
Received PING 10
Received PING 11
Received PING 12
Received PING 13
Received PING 14
---------------------------------
PING 0: RTT = 24
PING 1: RTT = 170
PING 2: RTT = 91
PING 3: RTT = 164
PING 4: RTT = 200
PING 5  was lost
PING 6  was lost
PING 7: RTT = 76
PING 8: RTT = 19
PING 9: RTT = 22
PING 10: RTT = 90
PING 11: RTT = 123
PING 12: RTT = 87
PING 13  was lost
PING 14  was lost
----------------------------------
Min RTT: 19  Max RTT: 200  Avg RTT: 95
11 Replies, 4 Lost
```

Ping Message Format

The ping messages in this lab are formatted in a simple way.

```
        PING sequence_number time CRLF
```

Note that there is a space character (" ") after PING, after the sequence_number, AND after the time. Yes, BE SURE TO INCLUDE THE SPACE between the time and the CRLF, it will make processing the message easier.

Compiling and Running

Feel free to compile and run PingServer and PingClient from Eclipse. Keep in mind that you'll need to run BOTH PingServer and PingClient at the same time (start PingServer first). When running both, start up the PingServer first and then the PingClient.

Grading

This project will be graded on a 100 point scale as follows:

- 30 points : PingClient successfully SENDS ping messages to the PingServer
- 30 points : PingClient successfully RECEIVES ping replies from the PingServer
- 20 points : PingClient correctly processes ping replies
- 20 points : PingClient prints required statistics minRTT, maxRTT, etc.

You may also lose points for writing bad code or failing to document your code. Be sure to comment your code to better explain what it's doing.

Submission

Submit your PingClient Java code by uploading it to the project assignment in Moodle. You are responsible for making sure that your submission contains the correct file. Add your name and class as comments within the code to identify your work.