CSCE 5580: Computer Networks

Programming Assignment 1

Due: 11:59 PM on Monday, February 17, 2020

PROGRAM DESCRIPTION:

In this assignment, you will write two complete C programs to support a client/server model using Linux sockets for a UDP "ping" utility, similar to the ping utility already available on our CSE machines.

- **Server**
  - The server program will be called with one command-line argument, the port number being used, such as `./pingsvr 8001`. If the user calls the server program with too few or too many arguments, you will print out a usage statement and terminate the program.
  - The server will set up a UDP socket on the Internet (i.e., INET) domain and then wait in an infinite loop listening for incoming UDP packets, specifically `PING` messages from a client. Your server should be able to support multiple clients at the same time (though no extra work is expected to support this requirement).
  - *Packet Loss*

    UDP provides applications with an unreliable transport service. Messages may get lost in the network due to a variety of reasons. Since packet loss is rare or even non-existent in typical campus networks, including UNT's, the server in this project will inject artificial loss to simulate the effects of network packet loss. The server will simulate 30% packet loss through generation of a seeded, randomized integer that will determine whether a particular incoming `PING` message is lost or not.
  - When a `PING` message comes in from a client and if the packet is not lost, the server will print the client message to the terminal and then send a `PONG` message back to the client. If the packet is determined to be lost, the server will print an appropriate message to the terminal and simply "eat" the message by not responding to the client.
  - The server will remain "always on" until a user enters `Ctrl-C` (`^C`) to send an interrupt signal to the program to terminate.
- **Client**
  - The client program will be called with two command-line arguments, the hostname of the server and the port number being used, such as `./pingcli cse06 8001`. If the user calls the client program with too few or too many arguments, you will print out a usage statement and terminate the program.

- The client will send 10 automated `PING` messages to the server using a UDP socket, where automated means the message is built in the code, not entered from the keyboard. 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.
- Specifically, for each of the 10 `PING` messages, your client program should:
  - send the `PING` message using the UDP socket and print a status message;
  - if the response message is received from the server, calculate and print the round trip time (RTT) in milliseconds for each message; otherwise, print a status message that it timed out.
- After all of the PING messages have been sent (and responses received or timed out), the client program should report the following and then terminate:
  - the number of messages sent, the number of messages received, and the message loss rate (as a percentage);
  - the minimum, maximum, and average RTTs for all of the `PING` messages in milliseconds.

Your program should run on the INET domain using SOCK_DGRAM (i.e., UDP) sockets so that the server and the client execute on a different CSE machine.

Given the randomness of what messages get dropped, it could be that less than or greater than 30% of the messages are dropped. Notice, however, in the SAMPLE OUTPUT that the server did indeed receive 10 `PING` messages, but "dropped" a few of the messages through our *Packet Loss* simulation.

You will also need to make sure you are able to handle any error cases.

SAMPLE OUTPUT (user input shown in **bold**)**:**

==> SERVER on cse06

```
$ ./pingsvr
usage: ./pingsvr <port>
$ ./pingsvr 8001
[server]: ready to accept data...
[client]: PING
[server]: dropped packet
[client]: PING
[client]: PING
[server]: dropped packet
[client]: PING
[client]: PING
[client]: PING
```

```
[client]: PING
[server]: dropped packet
[client]: PING
[client]: PING
[server]: dropped packet
[client]: PING
^C
```

==> CLIENT on cse05

```
$ ./pingcli
usage : ./pingcli <hostname> <port>
$ ./pingcli cse06
usage : ./pingcli <hostname> <port>
$ ./pingcli cse06 8001
 1: Sent... Timed Out
 2: Sent... RTT=0.650000 ms
 3: Sent... Timed Out
 4: Sent... RTT=0.359000 ms
 5: Sent... RTT=0.213000 ms
 6: Sent... RTT=0.191000 ms
 7: Sent... Timed Out
 8: Sent... RTT=0.346000 ms
 9: Sent... Timed Out
10: Sent... RTT=0.368000 ms
10 pkts xmited,  6 pkts rcvd, 40% pkt loss
min: 0.191000 ms, max: 0.650000 ms, avg: 0.246167 ms
```

REQUIREMENTS:

- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, course section, date, and brief description), comments for each variable, and commented blocks of code.
- Your programs should be named "**`project1svr.c`**" and "**`project1cli.c`**", without the quotes, for the server and client code, respectively.
- Your program will be graded based largely on whether it works correctly on the CSE machines (e.g., `cse01`, `cse02`, …, `cse06`), so you should make sure that your program compiles and runs on a CSE machine.
- Please pay attention to the **SAMPLE OUTPUT** for how this program is expected to work. If you have any questions about this, please contact your instructor or TAs, assigned to this course to ensure you understand these directions.
- This is an individual programming assignment that must be the sole work of the individual student. Any instance of academic dishonesty will result in a grade of "F" for the course, along with a report filed into the Academic Integrity Database.

SUBMISSION:

- You will electronically submit your two C source code files, `project1svr.c` and `project1cli.c`, to the **Project 1** dropbox in Canvas by the due date.