Harry Lum

CS 428: Computer Networks

Programming Assignment #1

The goal of programming assignment #1 is to write a client program in C++ that pings a provided server over the UDP protocol. The client program must ping the server 10 times and log the time between the ping and the server's reply. If the server does not reply in a second, the client must stop waiting for a response and print a timeout message accordingly.

In the client program, there were several design choices that were made. First, much of the code from the client program can be extracted from the server program. The code that initializes the UDP socket can be transferred over. We no longer need a client address, but setting up the server information is the same (IPv4, localhost, and port number). We need to make sure that the socket times out when waiting to receive packets, so we create a timeval struct (specified in <time.h>) and specify it in setsockopt function, which allows us to toggle an option to halt blocks caused by the recvfrom function after 1 second. We then ping the server 10 times using a for loop.

In each instance of the loop, we ping the server using the sendto function, and begin a timer. We wait for the response from the server using the function recvfrom, which returns the length of the output. Once the function gets input from the server or times out, we end the timer. If the function times out, the output is negative, so we can check the output and print a timeout message accordingly. If the output is 0 or more, that means we received a response, so we print a "Received reply" message. A delay of 1 second was added before this print statement to match the delay that would occur in the first case, just to keep the output stream at a steady pace in the case of repeated successful pings. We then null terminate the buffer before we proceed to the next ping or halt the program if there are no more pings left.

This code runs into errors when the socket is not initialized correctly, and because the current code has no method of catching this bug, unexpected output may occur at times. This can easily be remedied by checking the return value at socket initialization (if it is less than 0, then there was an error). Setting socket options and timers may also fail without warning since there are no checks for these errors, so adding appropriate code can remedy this issue.

The program is unsuccessful when running on Windows (all packets time out), but functions as expected on remote. Screenshots of sample output from remote are given on the next pages.

```
[/home/hlum1/cs428] (master)
[hlum1][remote04] $ ./client
Pinging UDP socket with 10 pings
Reply from ping 0 in 105433 ns
Reply from ping 1 in 41680 ns
Timeout from ping 2
Reply from ping 3 in 42020 ns
Reply from ping 4 in 99076 ns
Reply from ping 5 in 53325 ns
Reply from ping 6 in 43078 ns
Timeout from ping 7
Timeout from ping 8
Reply from ping 9 in 46559 ns
```

```
[/home/hlum1/cs428] (master)
[hlum1][remote04] $ ./client
Pinging UDP socket with 10 pings
Reply from ping 0 in 42819 ns
Timeout from ping 1
Reply from ping 2 in 72959 ns
Reply from ping 3 in 45542 ns
Reply from ping 4 in 63362 ns
Reply from ping 5 in 50964 ns
Reply from ping 6 in 88600 ns
Reply from ping 7 in 57230 ns
Reply from ping 8 in 54302 ns
Reply from ping 9 in 41592 ns
```

```
[/home/hlum1/cs428] (master)
[hlum1][remote04] $ ./client
Pinging UDP socket with 10 pings
Reply from ping 0 in 40539 ns
Timeout from ping 1
Reply from ping 2 in 69498 ns
Timeout from ping 3
Reply from ping 4 in 43903 ns
Timeout from ping 5
Reply from ping 6 in 41035 ns
Timeout from ping 7
Reply from ping 8 in 39744 ns
Reply from ping 9 in 38953 ns
```