

Alison Lee
Programming Assignment 1
2021/06/24

DESIGN DECISIONS

I found that most of the code provided for the server could be copied over to the client code. However, there were a few differences. I did not copy over the `bind()` as it is not necessary for the client side code. Inside the for loop, the order of the `recvfrom()` and `sendto()` is the reverse of that of the server code. Furthermore, the address parameter of `recvfrom()` and `sendto()` is changed from 'cliaddr' to 'servaddr' to reflect the fact that the client sends pings to the server and receives responses from the server.

Some additions I made were `setsockopt()`, and for the timing of the ping-and-response, I used the C++ chrono library. The option parameter of `setsockopt()` was a `timeval` variable that was set to exactly 1.0 seconds. This sets the socket to timeout at that specified amount of time.

Although I wrote my code in a Windows environment, I wrote for it to run in a Linux environment. As a result, I had to make significant changes to run my code in my Microsoft Visual Studio working environment (rather than in remote). These changes are:

1. Using `<ws2tcpip.h>` and `<winsock2.h>` to replace `<sys/socket.h>`, `<arpa/inet.h>`, and `<netinet/in.h>` due to the missing POSIX headers in Visual Studio. I also removed `<sys/types.h>`.
2. Initialization of winsock with `WSAStartup()`.
3. Making my socket timeout variable a `DWORD` instead of a `timeval`
4. Adding `-lws2_32` to the Makefile
5. Changing `MSG_CONFIRM` to 0 in `sendto()` and `recvfrom()` in both the server and client files, since there was no Windows-equivalent flag.

Despite the changes I made, I consistently got a RTT of 0 ms for all 10 pings (shown in screenshot), and could not find a way to fix this. My code was fully functional in Linux, however.

In the next page are the screenshots of execution in the remote session, and in Visual Studio's Command Prompt.

Linux:

```
alee889@remote.cs.binghamton.edu
alee889@remote05:~/cs428/project1$ ./udp_ping_server &
[1] 2193871
alee889@remote05:~/cs428/project1$ ./udp_ping_client
0: RTT: 0.094418 milliseconds
1: Timeout occurred (1023.45 milliseconds elapsed)
2: RTT: 0.106878 milliseconds
3: RTT: 0.039584 milliseconds
4: Timeout occurred (1023.76 milliseconds elapsed)
5: Timeout occurred (1023.97 milliseconds elapsed)
6: Timeout occurred (1023.94 milliseconds elapsed)
7: RTT: 0.107852 milliseconds
8: RTT: 0.040001 milliseconds
9: RTT: 0.033419 milliseconds
alee889@remote05:~/cs428/project1$

alee889@remote05:~/cs428/project1$ ./udp_ping_client
0: RTT: 0.079912 milliseconds
1: RTT: 0.051072 milliseconds
2: RTT: 0.036877 milliseconds
3: RTT: 0.056296 milliseconds
4: RTT: 0.032299 milliseconds
5: RTT: 0.031005 milliseconds
6: RTT: 0.032376 milliseconds
7: RTT: 0.036894 milliseconds
8: RTT: 0.034012 milliseconds
9: RTT: 0.030279 milliseconds
alee889@remote05:~/cs428/project1$
```

Windows:

```
Developer Powershell
C:\Users\syzygy\OneDrive\Desktop\CS428\project1-windows>udp_ping_client.exe
0: RTT: 0 milliseconds
1: RTT: 0 milliseconds
2: RTT: 0 milliseconds
3: RTT: 0 milliseconds
4: RTT: 0 milliseconds
5: RTT: 0 milliseconds
6: RTT: 0 milliseconds
7: RTT: 0 milliseconds
8: RTT: 0 milliseconds
9: RTT: 0 milliseconds
```

ERROR CASES

The code can run into errors if socket initialization was unsuccessful. In Windows, this error appears as `INVALID_SOCKET`. Another error can come from unsuccessfully setting the timeout in `setsockopt()`. This error can be caught through `if(setsockopt(...) < 0).` As

mentioned, I also got errors with my Windows code, likely due to the many changes I made for the code to be able to run in Windows/Visual Studio.