Baptiste Saliba

CS\_428

Programming Assignment 1

**Design Decisions:**

In terms of design decisions, most of my code seems pretty standard. I create a new socket for the client using the same specs as the server (UDP, IPv4), I connect to the server using the local host address and port that we used in the server code, and I send/receive a packet.

For the case where a packet is lost, I tried two different approaches, but only the second approach worked. My first approach was the bad one. It consisted of something like this:  
  
 A screen shot of a computer

Description automatically generated

It assumed that read would return -1 if it couldn’t get a reply which in that case it would work. However, I later realized that the sockopts are set so as to indefinitely wait if nothing is received, so this approach didn’t work.

My second approach is the one that worked. I used the timeval struct to create a timeout object set to 1.0s. I then set the sockopt SO\_RCVTIMEO, to the new timeout object I had created. This made it so that when receiving a packet back, the socket would timeout if nothing was received after 1.0s.   
  
A close up of a screen

Description automatically generated

Remote vs. Local:

On remote everything works fine. As for local I had to make one very minor change where in the server code I had to change the MSG\_CONFIRM flag to a 0. Other than that, it was exactly the same.

**Output:**

Remote

A screenshot of a cell phone

Description automatically generated

Local

A screenshot of a cell phone

Description automatically generated

**Possible Edge Cases:**

One case that I thought might cause an error was if the client sends a packet, but no response comes in within a second and as it sends the second response it gets the response from the server which would cause a faulty RTT. However, considering the average RTT it seems unlikely that any packet would be sent in more than a few hundred microseconds. Additionally, I’d have to read up on the socket read() method but I’m pretty sure it accounts for that case. The code could break as you try to create a socket but that’s unlikely.  
To improve the program you could definitely make the 10 packets send in parallel to improve the speed. You could also add error checking for setting up the socket.