Internet Technology Project 1 Report
Warmup: Socket programming 101

1. Team details: Clearly state the names and NetIDs of your team members (there are 2 of you).

Wesley Zhou (wgz4) and Richard Li (rl902)

2. Collaboration: Who did you collaborate with on this project? What resources and references did you consult? Please also specify on what aspect of the project you collaborated or consulted.

This was a joint collaboration between Wesley and Richard through the use of GitHub. Our specific code is labeled and signed but a brief overview of the division of work is as follows: Richard did steps 1-4, experimenting with sleeps, separating the programs and creating the initial code for string manipulation and transfer.
Wesley then completed Step 5, creating the program to read strings from a file and translate them onto a new file through the sending of data from client to server.

Our consulted references were as follows:
Python Socket Library Documentation: https://docs.python.org/3/library/socket.html
A GeeksForGeeks article: https://www.geeksforgeeks.org/socket-programming-python/
And the Recitation guidance

3. What did you observe after running step 2 above? Can you explain why you see what you see?

```
rl902@ilab2:~/Desktop/IT-Proj1$ python3 proj.py
[S]: Server socket created
[S]: Server host name is ilab2.cs.rutgers.edu
[S]: Server IP address is 128.6.13.3
[C]: Client socket created
[S]: Got a connection request from a client at ('128.6.13.3', 53416)
[C]: Data received from server: Welcome to CS 352!
Done.
```

Figure 1: the terminal output after running proj.py without modifications

As you can see the program runs normally without any resulting confusing output. I also noticed that between the initial 3 server outputs and the first client response, there was a noticeable pause. This same was observed right before the “Done.” statement was printed.

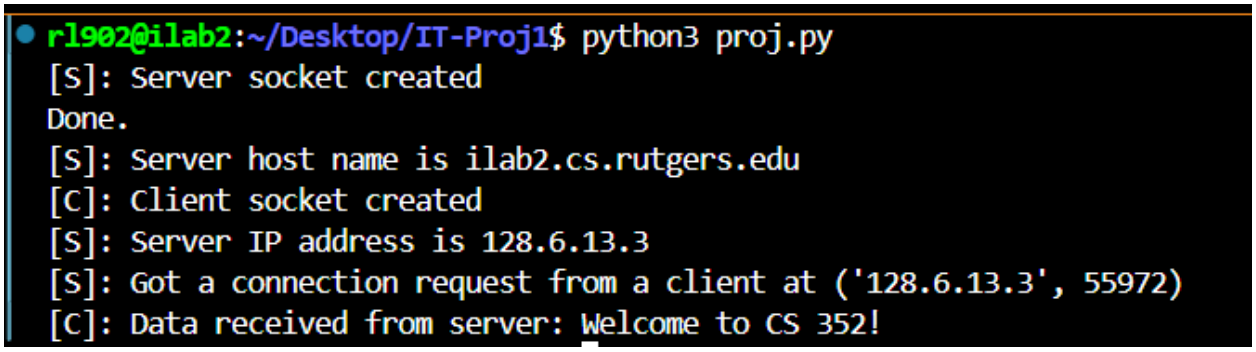

Figure 2: the terminal output after running the altered proj.py without sleep statements.

There is a clear jumbling of outputs as both programs run the print statements the moment they are called upon in the compile time of the program. The program took less than a second to finish with no noticeable pauses.

This is easily explained by taking a look at what the sleep arguments are doing. The first pauses for a random period of time after initiating the server program and the second pauses for a time unit of 5 after the client program is run. By doing so it allows the server thread to finish initializing and setting up its information before the client thread begins creating its own socket. Then once all exchanges are finished (after an arbitrary unit of 5 time has passed), the program announces its conclusion. We can clearly see that without sleep, the main function announces its completion before the client has even begun its socket initialization. This highlights the importance of allotting proper time for threads to reach completion before printing out declaratory statements in the main function.

4. Is there any portion of your code that does not work as required in the description above? Please explain.

All of our code works as intended.

5. Did you encounter any difficulties? If so, explain.

Our main difficulties with the project actually lay in GitHub rather than the programming itself. Both me and Wesley are pretty adept with python and after a bit of research to understand the socket library the code solution was simple. However, I have only just recently begun utilizing Git to its fullest extent last semester, and Wesley is only beginning to learn now.

6. What did you learn from working on this project? Add any interesting observations not otherwise covered in the questions above. Be specific and technical in your response.

Beyond a refresher on python syntax and an introduction to the socket library, most of our gained knowledge in this project was learning how to collaborate on Github. As mentioned

in the last question, our skill with Git bordered on novice, and utilizing this project as a motivator both me and Wesley gained key insight on the ins and outs of commits and branches on Git. In regards to other class material, I think the server client relationship mirrored the actions of a TCP handshake (albeit at a much higher level) and deepened our understanding of the inner workings of primitive networks.