CS352 - Project 1 Report

Arun Felix and Bhavya Patel

February 17, 2025

1 Team Details

- Arun Felix (netid: ajf277)
- Bhavya Patel (netid: bsp75)

2 Collaboration

We used the Python documentation, and Bhavya focused on the first half. I did the last half and developed testcases.

3 Observations after step 2

In step two, we removed the time.sleep(5) statement. Notice that the program will not behave consistently.

- First, multithreading is inconsistent, and the behavior of multithreaded programs is inconsistent.
  - The main function can finish before the threads finish.
  - We can fix this using the join function, so ensure the program stops only after the client and server threads are complete.
- Second, the server might not be able to bind to a particular port number since it takes time to free the port number.
  - This is after running the program consistently.

4 Difficulties

We didn't face any difficulties.

5 Design decisions

- Because waiting for the port to free up is cumbersome, we implemented a function for the server to wait until the port becomes available.

6 Testing

- We put all tests in the tests folder.
- We created a tests.py file with the client and server functions. We call the server and client functions with an input file, an output file, and an expected output file.
- We tested a couple of corner cases, like having an empty file, which behaves as intended, giving an empty output file.

7 What'd we learn?

- This project helped us understand why we need sockets and what binding and listening do.
  - Systems Programming taught sockets, but it was slightly complex, requiring us to use multithreading explicitly.
  - The Python socket interface provided many abstractions over C's socket interface.
- We also learned about the netstat utility on UNIX systems.