Project 2 Report

Note: We used the select.select() function individually for each ts file at first to just receive confirmation “received” messages. When we are actually sending host queries, we use the select.select() function for both the ts files at the time. Thus, we adhered to the requirements of the project.

1. Team details: Clearly state the names and netids of your team members (there are 2 of you).
    a. Palak Mehta (pm862) and Riya Bemby (rb1080)
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.
    a. We collaborated with each other on this project and used multiple different websites as a source of explanation on socket programming relating to blocking and nonblocking sockets. For instance, we used this article from Medium (https://medium.com/vaidikkapoor/understanding-non-blocking-i-o-with-python-part-1-ec31a2e2db9b) to understand the select() function better. It helped us understand the logic of different parameters like readable, writable, and error as well as what we are meant to use for each. Along with this, our respective recitations were really helpful in understanding how to account for the “wait 5 seconds before timing out” portion since we learned about the optional 4th parameter of select(). Additionally, since we did not have a lot of prior knowledge about blocking sockets and non-blocking sockets, we used these two articles (https://subscription.packtpub.com/book/cloud-and-networking/9781786463999/1/ch01lvl1sec11/changing-a-socket-to-the-blockingnon-blocking-mode) and (https://www.scottklement.com/rpg/socktut/nonblocking.html) to learn how the “client” choose to accept data from multiple “servers”. Once we had a better idea about the difference, we were looking for examples of non-blocking sockets. This website (https://steelkiwi.com/blog/working-tcp-sockets/) really helped visually show how to implement the creation of a socket, set non-blocking, and receive data from a server. Overall, these resources helped in understand the structure of the project better and thus helped us complete the project on time.
3. Discuss how you implemented the LS functionality that tracks which TS responded to a given query or timing out if neither TS responded. Please be clear and specific.
    a. The functionality of our ls.py file is to first create a socket connection with the client.py file. Through this connection, we are able to get the total number of hosts that we need to find the IP addresses for and create an array called inputArr that has all the host names. Then we create two socket connections. The first connection is between the ls.py file and the ts1.py file. The second connection is between the ls.py file and the ts2.py file. For both sockets, we unset the blocking and received confirmation that we were correctly receiving messages from the ts1.py and ts2.py files in the ls.py file. Then we started sending the host names to the ts1.py and ts2.py at the same time. Since we only expect either the ts1.py and the ts2.py to respond–not both–our ls.py file listens

to responses from both ts files. If neither ts files respond in 5 seconds, we know that host does not have any IP address stored. Else, based on the file that was sending a message to the ls.py file, we compared the socket connection to make sure the connection was still valid. Then, we append the response containing the IP address to an array called dataArr. After we received all the responses, we created a new array called resArr that contained all the hosts from the inputArr array and append the phrase " - TIMED OUT" to each host string. This way the resArr array will arbitrarily contain a time out message for all the hosts. Then, we iterated through both the inputArr array and the dataArr array to check if there is ever a host that has an IP address stored in the dataArr array. If the dataArr array does have the host name's IP address, then we removed the Timed Out message for that host in the resArr array and added the correct, corresponding IP address for that host to the resArr array at the same index that the host was before. Finally, we send the content of the resArr array to the client that has both the host and IP address or host and a timed out message.

Through this entire process we assume that all the hosts do not have any IP addresses stored in the ts1.py and the ts2.py files. Once we send the query for the IP address from the ls.py file to the ts1.py and the ts2.py files and get a valid response, instead of the query timing out in 5 seconds, we then update the result array with the respective IP addresses of the host. This logic is able to correctly help the ls.py file communicate with the ts1.py and the ts2.py simultaneously to correctly send the IP addresses or Timed out messages to the client.

4. Is there any portion of your code that does not work as required in the description above? Please explain.
    a. All our code works correctly.
5. Did you encounter any difficulties? If so, explain.
    a. We encountered several difficulties in our project. For instance, the first thing we had an issue with was the select function and how to choose which server (ts1 or ts2) should be listened from. Initially we were only sending and receiving data between the ls.py file and the ts1.py file. Then after processing that data from ts1 in the ls.py file, we sent the hosts to ts2 and then sent the responses back to the ls.py file. However, this was not the correct implementation as we are supposed to send and receive data from ts1 and ts2 simultaneously. This was a huge challenge because we did not know that we can pass in multiple readable files as the first parameter for the select function. However, after thorough research we were able to navigate past this issue. Along with this, we were not able to understand how to run our code using different hosts and different machines. We tried a myriad of combinations for the ilab hosts, however it proved to be futile since running the files on different machines created issues in connecting the posts and the hosts correctly. To address this, we went to the TA's office hours and then started testing with the appropriate ilab hosts based on the hosts of our machine. This was a concept we were familiar with after our research on this matter, thus after a long process of trial and error we figured out how to fix this issue.

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.
    a. We learned a great deal through this project. We learned that the select function is effectively able to listen to multiple different files as long as only one of those files respond at a time, at least in our case where both TS1 and TS2 could not have the same domain name and IP listed in their datasets. This is enlightening because this project helped us better understand the structure of a socket connection and how nonblocking sockets are very effective but are also prone to errors. If multiple messages are received, then the local server can run into issues because it only expects one message in return. Also it was mesmerizing to see how we can run our code on different machines and these machines that run on different hosts can still work together. Since our ilabs are still connected, we are able to communicate with each other's machines. Overall, this project was challenging but also really amazing when we look back on the functionality of the ls.py file.