Report for Project #2: Asynchronous sockets

1. Team Members:

   Yuchen Wei (yw780) and Tiange Zhang (tz196)

2. Collaboration:

   We use GitHub to collaborate between the two of us, updating and communicating the parts we have completed and the changes and improvements we have made to the code. The primary reference materials we used in the process were from the lectures we received, the previous Python programming courses we participated in, and our experience in learning Python independently. We divided server.py and ls.py of the project, as well as the ts1&2.py and the report, into two people's jobs to complete and share the problems encountered in the process.

3. Discuss how you implemented the LS functionality that tracks which TS responded to a given query or timing out if neither TS responded.

   In our project, ls.py is connected to both the socket of client.py and the socket of ts1&2.py at the same time. The LS communicates with the Client using socket.listen and socket.accpet since there would only be one client connected to the LS in this project. However, we have to use selector to listen to responses from TS1 and TS2. Whoever gives the response first would be heard, and if neither respond, the selector would timeout after 5.0 seconds. After the Client sends the query to the LS, the LS will simultaneously transfer the query content to TS1 and Ts2. Then, through the select function of the socket, we give the response back to the client if we get one from any TS, and a "TIMED OUT" response if we don't get a response.

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

   After comparing the results of our tests with the samples included in the project and the description, we believe that our code fulfills all the requirements of the description and can be reflected by the tests.

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

   After realizing the complete function of the four files, we found that the LS and TSs sockets won't close properly. The client socket closes immediately after sending all queries, but the LS does not know it and keeps accepting data as well as the TSs. The LS eventually raises a "BronkenPipeError [Errno 32]" error, and the TSs raise a "ConnectionResetError [Errno 54] Connection reset by peer" error. We tried to find out if the client could send out some form of signal that indicates the servers to close their sockets but failed. Eventually, Yuchen sends an "EOF" string from the client to the LS that indicates that the client has finished sending query, and send it to the TSs as well. Then they will break the infinite while loop and close the socket properly.bil

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.

   In the process of completing this project, we have a clearer understanding of the establishment and connection between server and client in sockets. We learned the basics of how balancing load forms of distributed servers work and how they achieve balancing load. The creation of the load-balancing server as a relay also showed me how to quickly traverse all the balancing load services to satisfy the client's queries. We've also got to know the how to listen to multiple endpoints at the same time while implementing the LS using selector.