COL334 Assignment 3

Mihir Kaskhedikar 2021CS10551
Payas Khurana 2021CS50948

October 2023

1 Week 1

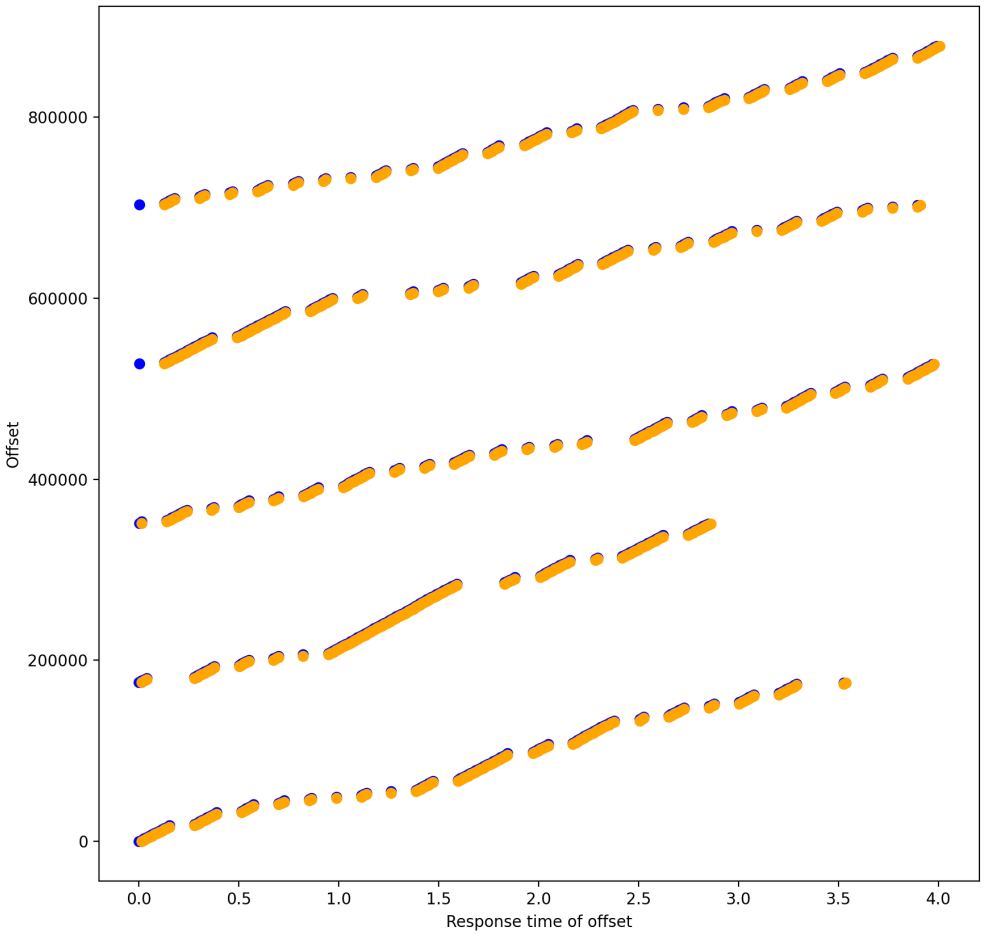
We have used **threading** to send requests parallelly to the server. Firstly we created a UDP socket and requested the number of bytes on it. After receving the `total_bytes`, we divide them almost equally in `num_threads` continous portions where `num_threads` were the number of threads we used. For each thread we created a new socket,ran a while looop and requested data from it such that the range of data requested (**[Offset,Offset+NumBytes)**) entirely lies in the portion of data assigned to this thread. NumBytes is assigned the value $\mathbf{min}(\mathbf{1448}, \mathbf{limit} - \mathbf{offset})$ where limit is the ending byte index of the data assigned to this thread. Similarly Offset is assigned the value *start* (starting byte index of data assigned) in the beginning and once we correctly receive the socket response for this data request, it is incremented by NumBytes to form the Offset in the next iteration of the while loop. If we dont receive the data, we dont change the Offset value and simply continue. In order to ensure that the response received from the socket is correct, we performed 2 important checks:

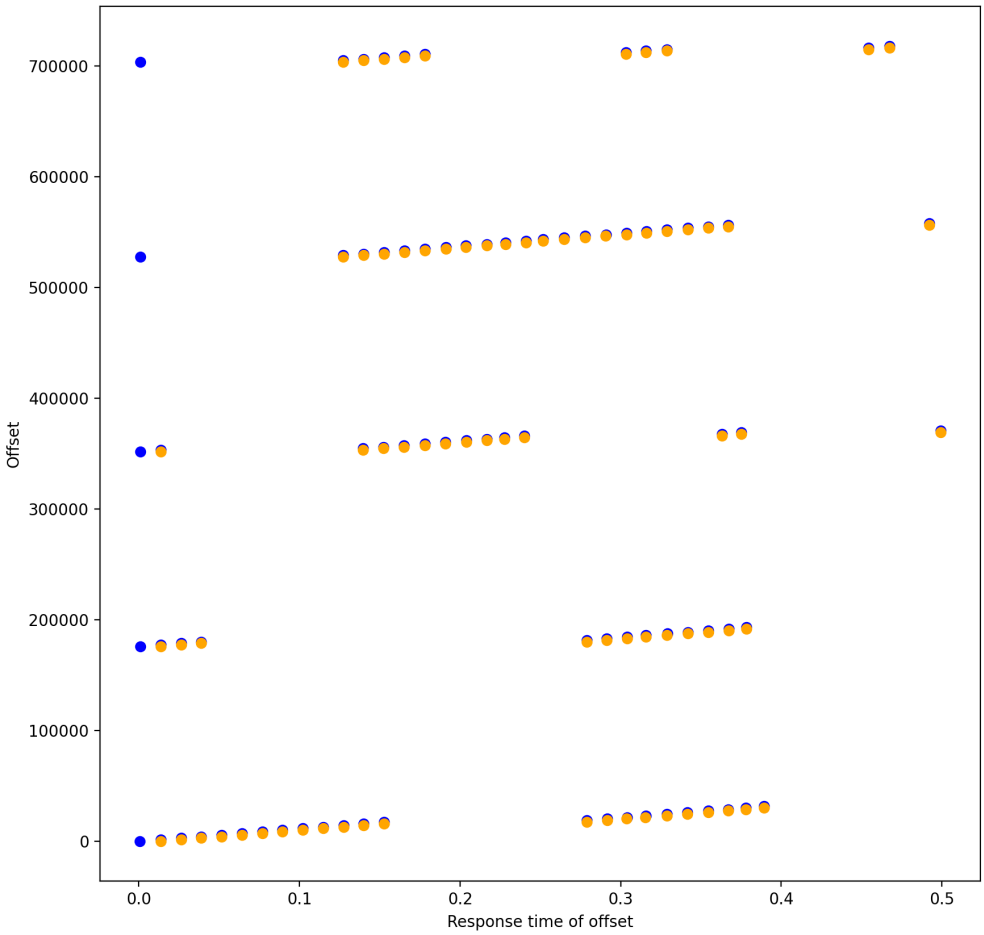
- Checking the length of actual data received (that is the response received after removing the "Offset:" and "NumBytes:" lines) is equal to NumBytes
- Checking the value mentioned in the "Offset:" field of response matches with the Offset value that we requested the socket to send.

If the response is received correctly, we append it to a string corresponding to the thread and change the Offset value. We have implemented 2 features to avoid token loss:

- In each thread, before we make a request to the socket we do **time.sleep(0.01)**.
- We use the **try-except** syntax to send or receive anything from the socket everytime with a cutoff of $0.1s$.

After completion of all threads we concatenate the strings of the threads and send it to the server. Below is the offset vs response time graph, when `num_threads` $= 5$. We were only able to test our code on local server (Vayu was not responding). The first image gives a complete picture of the request and reception times for all offsets (blue dots specify the request times and orange dots, the reception times, since they would be very close to each other for a particular offset and the offset after it, orange dots almost cover the blue ones). The second image gives a magnified image of the request and reception times between $0s$ and $0.1s$. From both the images one can clearly deduce that 5 threads were used for sending requests.


800000
600000
400000
200000
0
Offset
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Response time of offset


700000
600000
500000
400000
300000
200000
100000
0
Offset
0.0
0.1
0.2
0.3
0.4
0.5
Response time of offset