

COL334 Assignment-3 Report

Amaiya Singhal 2021CS50598
Om Dehlan 2021CS10076

Milestone 1

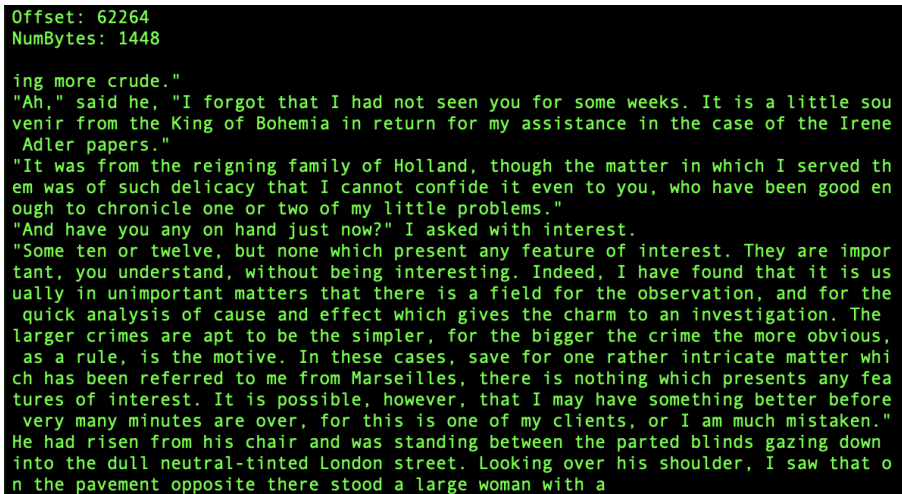
Sending and Receiving UDP Packets

- We used the `socket` library in python to create a UDP socket and to connect to the UDP server set up at `vayu.iitd.ac.in` on port 9801.
- We sent the messages to the server in the specified format:
 - Firstly the `SendSize` message to receive the total number of bytes.
 - Then the messages specifying the offset and the data size (set to 1448 bytes) to receive the specified number of bytes from the given offset (1448 bytes at a time).
- We noticed that we were able to receive a response from the server most of the times. Sometimes we did not receive a response which is expected since the server emulates a loss rate.



Size: 887493

Receiving Total Size



```
Offset: 62264
NumBytes: 1448

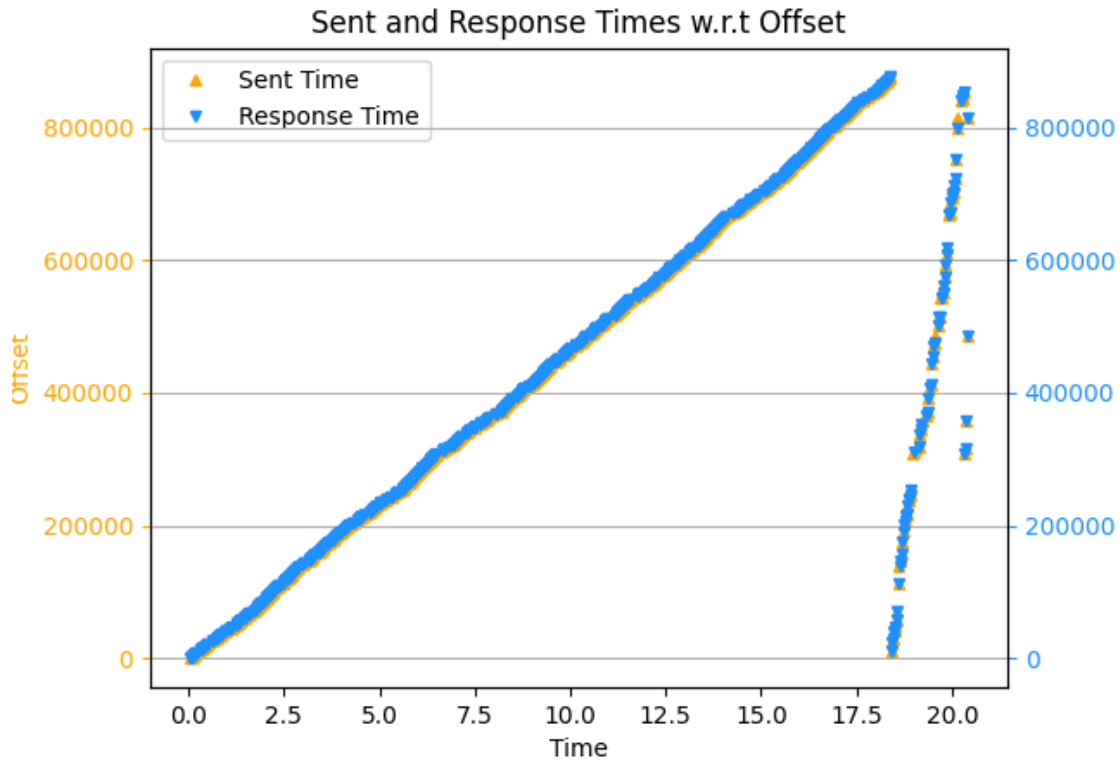
ing more crude."
"Ah," said he, "I forgot that I had not seen you for some weeks. It is a little souvenir from the King of Bohemia in return for my assistance in the case of the Irene Adler papers."
"It was from the reigning family of Holland, though the matter in which I served them was of such delicacy that I cannot confide it even to you, who have been good enough to chronicle one or two of my little problems."
"And have you any on hand just now?" I asked with interest.
"Some ten or twelve, but none which present any feature of interest. They are important, you understand, without being interesting. Indeed, I have found that it is usually in unimportant matters that there is a field for the observation, and for the quick analysis of cause and effect which gives the charm to an investigation. The larger crimes are apt to be the simpler, for the bigger the crime the more obvious, as a rule, is the motive. In these cases, save for one rather intricate matter which has been referred to me from Marseilles, there is nothing which presents any features of interest. It is possible, however, that I may have something better before very many minutes are over, for this is one of my clients, or I am much mistaken."
He had risen from his chair and was standing between the parted blinds gazing down into the dull neutral-tinted London street. Looking over his shoulder, I saw that on the pavement opposite there stood a large woman with a
```

Receiving the Specified Bytes from the Specified Offset

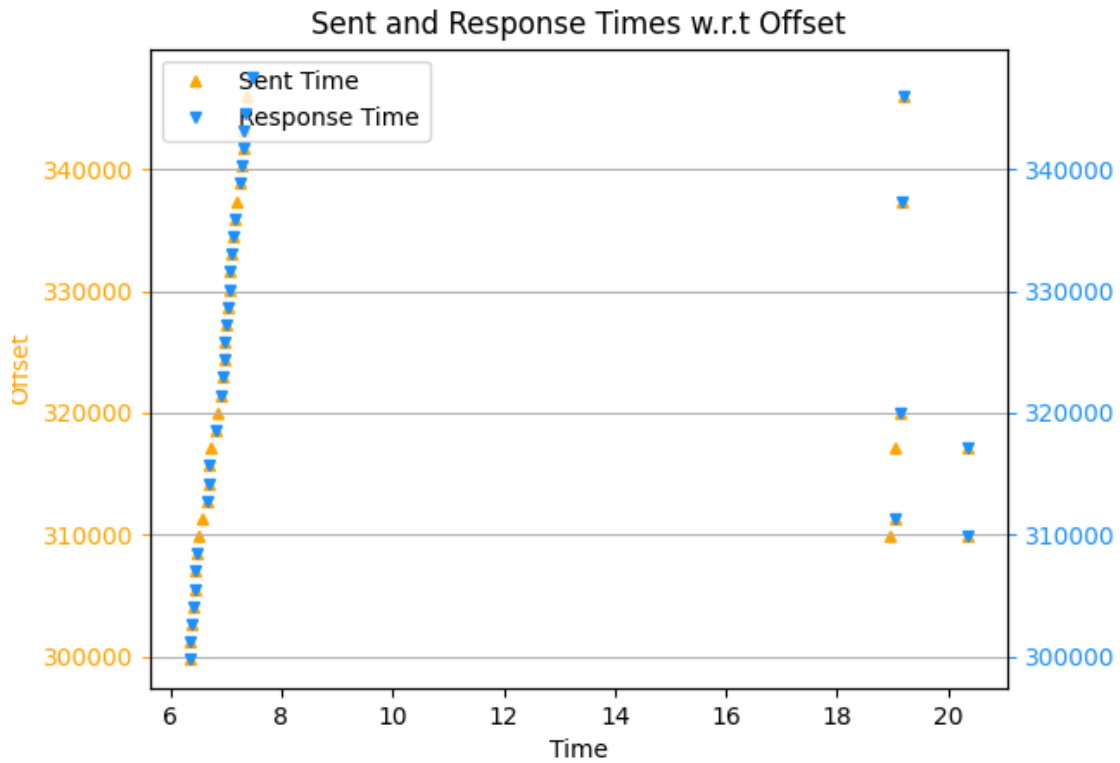
Receiving Data Reliably

- In this milestone we have implemented a simple SEND and WAIT strategy to ensure reliable reception of data.
- We send the requests one by one, each time increasing the offset by 1448 bytes.
- If a response is received, the request for the next offset is sent immediately, otherwise if a response is not received within 0.05 seconds then the next request is sent and the offset is appended to a list.
- When all the offsets (until the total size) have been requested then the offsets for which a response was not received are requested again.
- This process is repeated until we have received a response for each of the offsets (i.e. the list containing the offsets not received is empty) and we have received the entire data.
- We had stored each of the lines in a dictionary with keys as the offset number and the values as the corresponding bytes of data
- Then the MD5 hash for the entire data is calculated and submitted to the server for verification.

Graphs

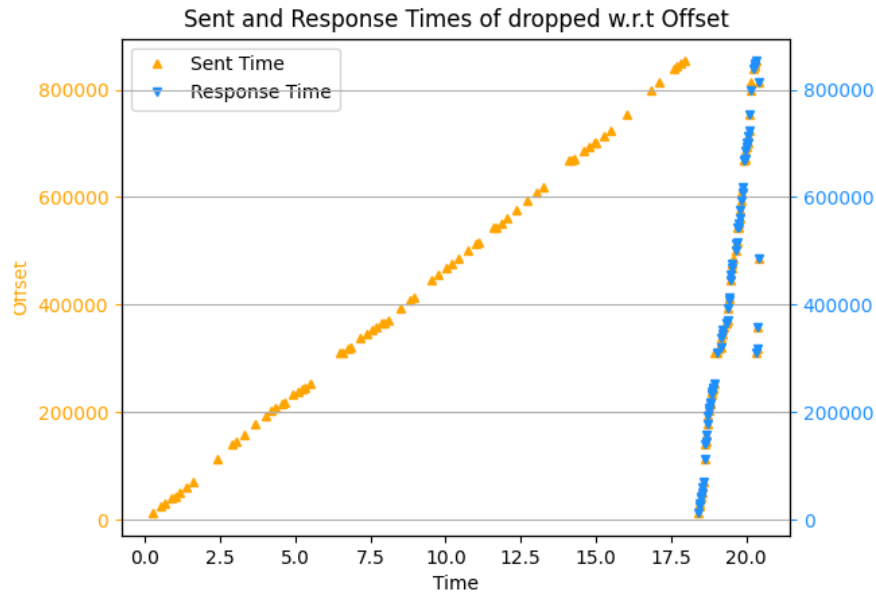


- The above graph shows the **Sequence Number Trace** for the requests sent and the responses received.
- All the requests are monotonically sent (in orange) and the responses are received for most of them (in blue). Due to the scale of this graph, it is not very clearly which responses are not received.
- Then the requests are sent again for the offsets for which we did not receive a response.
- This process had to be repeated twice until all the data was received and the MD5 hash could be submitted for verification.

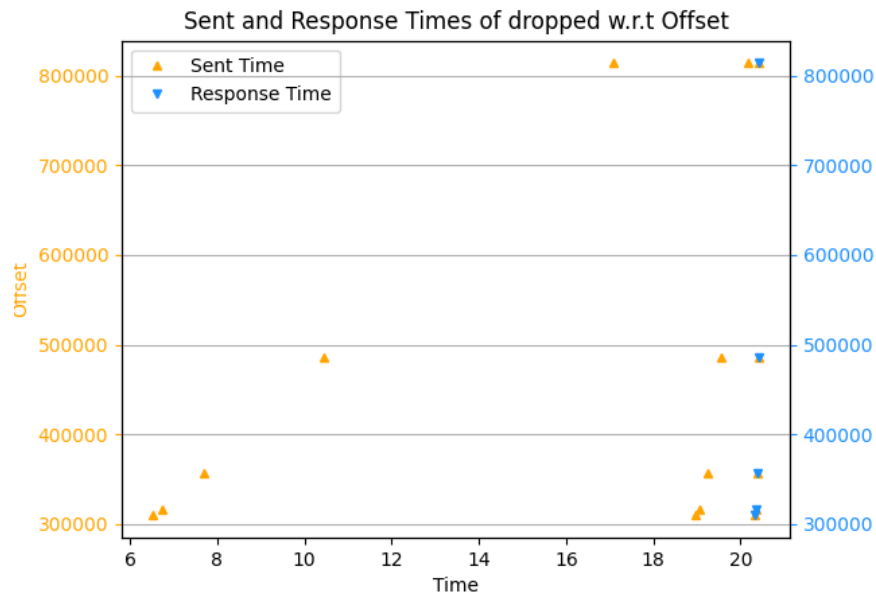


- The above graph shows a zoomed in version of the Sequence Number Trace.
- Here we can see that for some of the requests a response was not received (Orange triangle present but no corresponding Blue triangle indicating that a response was not received).
- Even in the second iteration some of the packets were dropped and finally in the third iteration all the remaining data was finally received.

Some More Graphs



- The graph in the left shows all the packets that were dropped in the first run. (There is no blue triangle for any of them indicating that no response was received).
- In the next iteration all of these requests (except 3) were responded to and finally in the third and the last run, the remaining data was received.



- The graph in the right shows the same information for the requests that were dropped both in the first and the second iterations.
- There were 5 such requests which were finally responded to in the third iteration.