COL334 Assignment 2
Aryan Dua

20/9/22

Part 1

The summary of Ports for both the parts is given as follows:

1. N UDP receiving ports for clients, one for each
2. N UDP sending ports for clients, one for each
3. N UDP receiving ports for server
4. N UDP sending ports for server
5. N TCP ports for communication from Client to Server
6. N TCP ports for communication from Server to Client
7. 1 Port for communicating the end of the program from the client to the server.

Part 2

Essentially just switched the file sending sockets from TCP to UDP, and the control signal sockets from UDP to TCP.

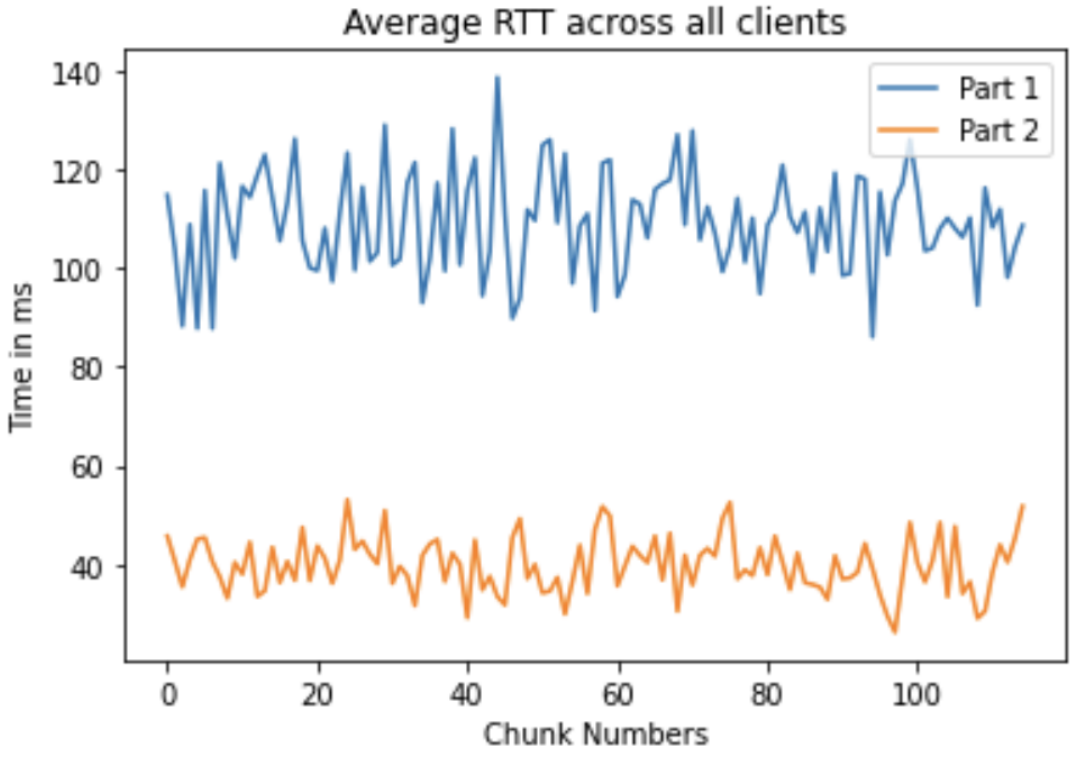
Please note::: The shell script does not work. Please run individually, on 2 different terminals.

Analysis

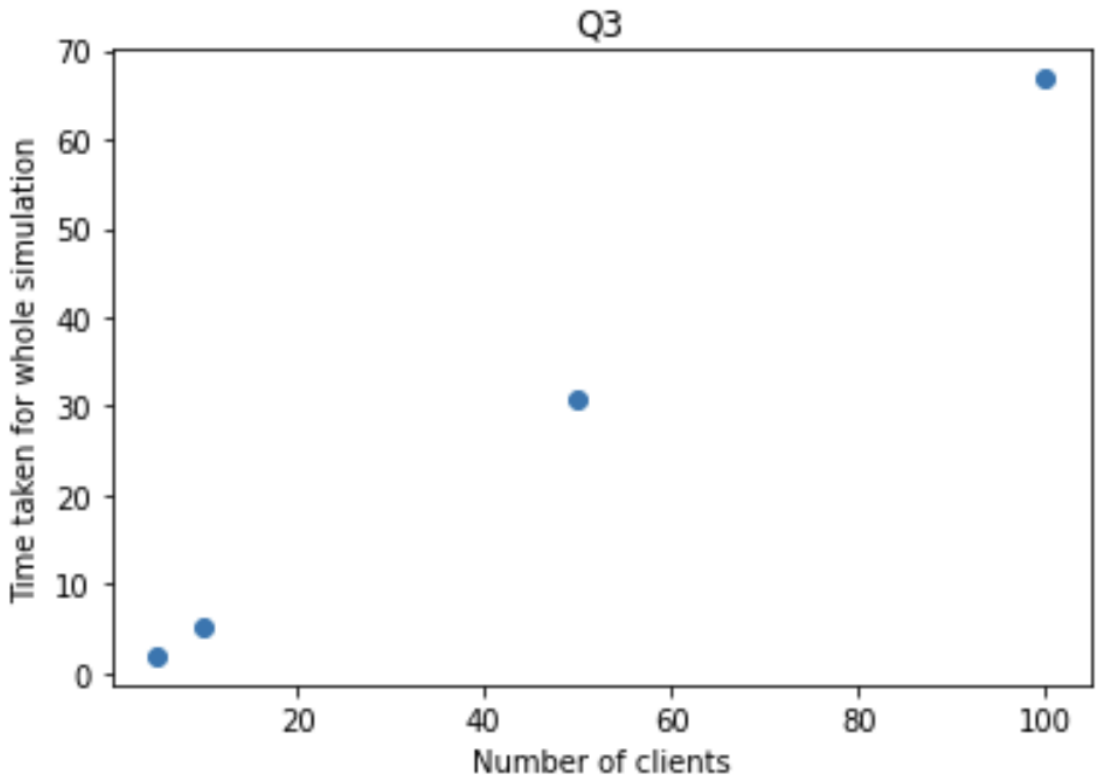
1. a) Average RTT for all chunks (Part 1) = 129ms
   b) Average RTT for all chunks (Part 2) = 43ms
   The Average RTT of Part 1 is higher, and this was expected because it uses TCP for the transfer of larger files, which happens for a longer duration of time. UDP is a much faster protocol than TCP because of the lack of acknowledgments required. All we need in UDP is an open port. With UDP, there is a seamless transfer of data between the client and server whereas, with TCP, there are checks and handshakes, which consume time.

2.

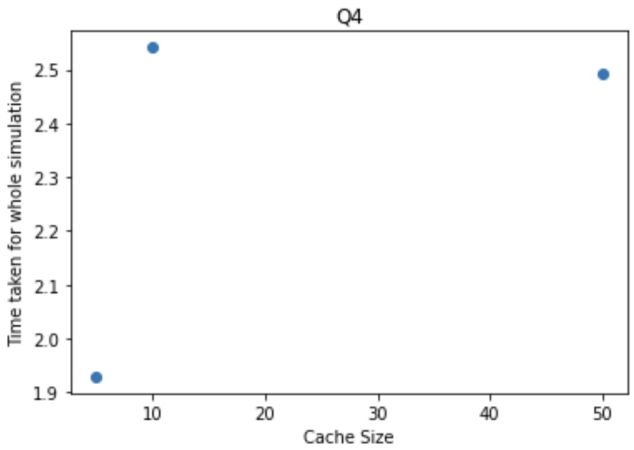
   No, there are no chunks that differ significantly in time taken, mostly all the chunks have the same RTT. Some peaks are visible in the graph, but if we look at the magnitude on the y-axis, the difference isn't quite significant. This is because the chunks are divided equally among the clients, and the sizes of the chunks are the same as well:- 1 KB.



3. The time increases linearly with the number of clients involved. This is because, as the number of clients increases, the cache size also increases. When we put the 2 effects together, the time taken nearly forms a linear relation with the number of clients.



4. I have tested my program on A2_large_file.txt, and it works but, it takes too too long for high values of n. For just n=8 it took 30 minutes to run. Therefore, for this part of the question, I shall perform the same simulation with A2_small_file.txt. The shape of the graph will be the same in any case.



5. The sequential method takes more time because of the number of times we encounter a cache hit. If we randomize the process the chances of it passing through the cache is much higher. For example, if on the one hand, all clients want chunk number 1, and on the other, one client wants chunk number 4, one wants 8, one wants 87, one wants 103, the chances of anyone being in the cache is much greater in the second case than is there in the first case. The explanation is basic probability theory.

Food For Thought

1. a) In a PSP network, every client contains its own data, and the process is decentralized.
   b) Because the service-providing nodes in a PSP network are dispersed, the service-requesting nodes do not have to wait very long.
   c) We save a lot of energy that could have gone waste in data transfer
   d) It is much faster as well, especially due to the cache.

2. There is added security. This is explained by the fact that a server manages the bits and pieces of information the clients hold. Security measures can be imposed on the reliable server. There is more accountability as well, due to the fact that there may be an authority monitoring the server requests and responses.

3. I would assign some Ids to each file as a header and then append it to the data chunk. (a 4-byte id, for example). Whenever some client/server receives the chunk, they separate the last 4 bytes to get the id and the rest of the chunk.