# TFTP Concurrent Server Design Document:

## Part a)

- We are using UDP sockets for implementing a TFTP server. Only RRQ messages are being processed by the server.
- Following types of messages are being used : DATA, ACK, ERROR, RRQ.
- DATA packets are being represented by a struct called DATA. ACK and ERROR packets are being represented by a hybrid struct called ACKORERROR owing to the similarity in their packet content as defined in the RFC. Similarly, RRQ packets are being handled using a struct called RRQ.
- Opcode and block number are being represented by unsigned short int variables, and data is being sent in the form of a character buffer of maximum 512 bytes.
- It is a stop and wait protocol ie, a new packet is not sent until the acknowledgment for previous packet arrives.
- Timeout is implemented using select() call. If there is a timeout, the packet is retransmitted. Maximum three retransmissions are allowed.
- At server side, we maintain a struct mySocket for each client which stores information like socket file descriptor corresponding to the client, time at which previous DATA segment was sent, file pointer for reading from the file, block number of sent DATA segment, client sockaddr_in struct, and number of retransmits for this sent packet.
- These structs are stored in a queue, such that the one with least sentAt value is first in the queue, so that we can conveniently update the timeout value in select() call using the head of the queue.
- Updated value of timeout is (5 - (current time - sentAt of head)) where 5 is time to wait for retransmission.
- If select returns 0, packet corresponding to client which is at head of queue is retransmitted and the client is pushed back at the end of the queue after updating sentAt and number of retransmission values.
- Whenever an ACK is received, the mySocket struct corresponding to the client is removed from the queue. The next DATA segment is read and sent to the client, and sentAt value, DATA segment, block number in mySocket struct are updated and the struct is pushed at the end of the queue.
- After transmitting a packet of size less than 512 bytes, socket corresponding to client at server side is closed.
- If a file that is not present in the server is requested by the client, the server replies with an ERROR message as specified in the RFC along with error code 1 and message as File not found.

## Part b)

- For implementing Jacobsen’s algorithm, we have added three double variables, estimatedRTT, estimatedDeviation and estimatedTimeout.

- Initial values of estimatedRTT is 1, and that of estimatedDeviation is 0.1.
- If an acknowledgment for a non retransmitted packet is received, we update these variables as follows :
    - estimatedRTT = (1-ALPHA) * estimatedRTT + ALPHA * actualRTT, where we use ALPHA=0.125, and calculate actualRTT as (current_time()-sentAt)/CLOCKS_PER_SEC.
    - estimatedDeviation = (1-BETA) * estimatedDeviation + BETA * |actualRTT - estimatedRTT|, where BETA = 0.25.
    - Values of ALPHA and BETA are taken from the textbook: Computer Networking by Kurose and Ross.
    - Note that both the variables above are updated simultaneously.
    - Now, we calculate estimatedTimeout = estimatedRTT + 4 * estimatedDeviation.
- In case of timeout, we retransmit and just double the RTO value.