Tuna Dalbeler	21802539	CS442
Çağrı Eren	21801831	Project 3 Report
Kaan Ateşel	21703694	

The entry point of our application is *tcp.py* as mentioned in the project specification pdf.

How you implemented your program?

We implemented our application by adding broadcast methods that we removed in project 2. We added the *sendAll()* method etc. Then we go over the book and the slides to understand how to algorithm works.

After the entry program (tbc.py) opens NP child processes, the child process opens 4 new threads for different jobs.

The first thread, the broadcast_thread, waits the time according to specifications (MINT, MAXT and AVGT) and sends a request message as broadcast. Continues until NR requests are sent.

The second thread, the listenRequests thread, listens to incoming messages from other processes. These messages are REQ (for requests) and ACK (for acknowledge) messages. For REQ messages, takes them and place them in the inboundREQs list. And for ACK messages, places them into the inboundACKs list.

The third thread is the order_manager_thread. Takes incoming requests and places them in a sortedQueue. Then sends broadcasts an ACK message including the Lamport Clock timestamp. “sortedQueue” is sorted according to a Lamport Clock. Then matches incoming ACKs with the corresponding REQs. If a Request takes ACK from all the Nodes(Processes), placed in the deliveredMSGs queue.

The fourth thread, writer_thread, watches the deliveredMSGs queue and if there is a message in there, writes to the file and removes it from the queue.

We also have two extra classes. LogialClock.py and Req.py. LogicalClock class provides basic Lamport Clock functions. Req class holds information for requests and which ACKs came corresponding to the request.

Which algorithm did you use in the program?

The main idea of the program is to have a consistent update sequence. Therefore any update operation or write operation should be done in the same order which means we have to reach totally-ordered multicast. In a scenario where no message is lost each sender will send a broadcast message which has been marked with its timestamp. This broadcast message is also sent to the sender itself. After the messages are sent, the nodes are receiving the messages then they sort the received messages according to their timestamp value and put them into their local queue. Finally, all nodes will have a local queue in the same order. After each process has its queue in the same order they send an acknowledgement to the other process. After each process received the whole acknowledgement from all other processes the message can be delivered. Then all messages are delivered in the same order.

We used the algorithm in the course book.[1]

[1] Distributed Systems, Maarten Van Steen and Andrew Tanenbaum, 3rd edition, Version 2020.