CSC 8980 Distributed Systems **Name: Ishu Goyal**

Assignment – 3

Question 1:

Algorithm:

1. P_0 initializes the logical clock to 0
2. Each Process P_i requests access to the resource using Request(Tp_i, i) by sending a request message with their respective logical clock timestamp to P_0.
3. P_0 orders and places the Request(Tp_i, i) received in accending order of logical clock timestamps on request_queue.
4. P_0 then sends an Inform(Tp_i, i) message to P_i which is at the top of request_queue.
5. P_i acknowledges receipt of the inform message inform(Tp_i, i) by sending an ack message using ack (Tpi, i) P_0.
6. P_0 grants access to the resource by sending a grant message i.e. grant(Tp_i, i) to the P_0
7. P_i releases the resource using release(Tp_i, i)
8. P_0 removes the P_i from the request_queue since it sent the release message and P_0 updates its logical clock to the maximum timestamp of all received messages.

- This algorithm is designed to ensure that processes requesting access to a centralized resource are granted access in the order in which they request it, based on their logical clock timestamp.
- The algorithm is also designed to be deadlock starvation free, meaning that each process requesting access to the resource will eventually be granted access because it ensures that the processes requesting access to the resource are granted access in the order in which they request it, based on their logical clock timestamp in FIFO.

Question 2:

To Prove: $\varepsilon / (1 - \kappa) \leq \mu$ together with PC1 And PC2 make anomalous behavior impossible.

Proof:

We have that:

$\varepsilon / (1 - \kappa) \leq \mu$

⇨ $\varepsilon \leq \mu(1 - \kappa)$

Also,

PC1: $\varepsilon \leq \mu$

PC2: $\varepsilon \leq \kappa\varepsilon$

On, combining these two inequalities, we get:

⇨ $\varepsilon \leq \mu(1 - \kappa) \leq \mu$

⇨ $\varepsilon / (1 - \kappa) \leq \mu$

Hence proved

Question 3:

Writer Priority i.e. When writer wants to access Critical Section -

- When a writer wants to access the Critical Section it sends a REQUEST to everyone.
- If any reader/writer does not want to access the Critical Section at the same time, they immediately REPLY to the request.
- If any other writer wants to access the Critical Section at the same time, then based on the sequence number, one with lower sequence number will receive REPLY and will defer the REPLY for another writer till it finishes Critical Section execution.
- If the sequence number is the same, then the writer number will decide the priority.
- If any reader wants to access CS when the writer wants to access it, the reader will REPLY to the writer's request immediately and the writer will defer the reader's request.

Reader Priority i.e. When reader wants to access Critical Section -

- When a reader wants to access the Critical Section it sends a REQUEST to everyone.
- If any writer does not want to access the Critical Section at the same time, they immediately REPLY to the request.
- If any other reader wants to access the Critical Section at the same time, then based on the sequence number, the reader with lowest sequence over others reader and writer(last priority once readers are done), will receive REPLY and will defer the REPLY for writers till it finishes Critical Section execution.
- If any writer wants to access CS when the reader wants to access it, the writer will REPLY to the reader’s request immediately and the reader will defer the writers request.