Tim278

Mvf282

**EE 360P HW3**

1.(10 pts) Some applications require two types of accesses to the critical section –read access and write access. For these applications, it is reasonable for multiple read accesses to happen concurrently. However, a write access cannot happen concurrently with either a read access or a write access. Modify Lamport’s mutex algorithm for such applications.

We only have to modify the read request because a write request must wait for all threads, while a read is only limited by write requests

A write request can enter CS as stated by Lamport’s algorithm:

A read request can enter CS iff:

* It has a request in the queue with a timestamp t
* t < all other **write** requests in the queue
* It has received an ack for its request from every other process or any message with timestamp > t

2.(10 pts) (a) Extend Lamport’s mutex algorithm to solve k-mutual exclusion problem which allows at most k processes to be in the critical section concurrently.

For a process to enter CS, the number of processes with smaller request timestamps than than the current or whose acks are still pending is less than k, since at most k processes can be in the CS concurrently. If there are more than (k – 1) processes with smaller requests or pending acks, then the process cannot not enter CS.

(b) Extend Ricart and Agrawala’s mutex algorithm to solve the k-mutual exclusion problem.

For any process that requests access to the CS, it is now forced to wait for at least N-k “okay” messages vs N meessages where N is the total number of processes for the system. With this change we can ensure that less than k processes are in the critical section currently.