# ASSIGNMENT-2

Hoorichoran CSE-A
205001043

Given Request model

$P_1 \rightarrow P_2 \parallel P_3 \rightarrow P_4 \parallel P_1$

a) Apply Lamport's D-Mutex Algo

| $P_1$ | $P_2$ | $P_3$ | $P_4$ |
|---|---|---|---|
| (1,1) (2,2) (2,3) (3,1) (3,4) | (1,1) (2,2) (2,3) (3,1) (3,4) | (1,1) (2,3) (2,2) (3,1) (3,4) | (1,1) (2,2) (2,3) (3,4) (3,1) |
| req(1,1) | | | |
| | rec req(1,1) | rec req(1,1) | rec req(1,1) |
| | rep req(1,1) | rep req(1,1) | rep req(1,1) |
| rec rep(1,1):P2 | | | |
| rec rep(1,1):P3 | | | |
| rec rep(1,1):P4 | | | |
| exec CS | | | |
| rel req(1,1) | | | |
| | rec rel(1,1) | rec rel(1,1) | rec rel(1,1) |
| | del req(1,1) | del req(1,1) | del req(1,1) |
| | req (2,2) | req(2,3) | |
| rec req(2,2) | rec req (2,3) | rec req (2,2) | rec req(2,2) |
| rep req(2,2) | | | rep req(2,2) |
| rec req(2,3) | rec rep(2,2):P1 | | rec req(2,3) |
| rep req(2,3) | rec rep(2,2):P4 | | rep req (2,3) |
| | | rec rep(2,3):P1 | |
| | | rec rep (2,3): P4 | |
| | | rep req (2,2) | |
| | rec rep(2,2):P3 | | |
| | exec CS | | |
| | rel req (2,2) | | |
| | | rec rel (2,2) | rec rel(2,2) |
| rec rel(2,2) | | | |
| del req(2,2) | rep req (2,3) | del req (2,2) | del req(2,2) |

| P1 | P2 | P3 | P4 |
|---|---|---|---|
| | | rec rep(2,3): P2 | |
| | | exec CS | |
| | | rel req(2,3) | |
| rec rel(2,3) | rec rel(2,3) | | rec rel(2,3) |
| del req(2,3) | del req(2,3) | | del req(2,3) |
| req(3,1) | | | req(3,4) |
| rec req(3,4) | rec req(3,4) | rec req(3,1) | rec req(3,1) |
| | rep req(3,1) | rep req(3,1) | |
| | rec req(3,4) | rec req(3,4) | |
| rec rep(3,1):P2 | rep req(3,4) | rep req(3,4) | |
| rec rep(3,1):P3 | | | |
| | | | rec rep(3,4):P |
| | | | rec rep(3,4):P |
| | | | rep req(3,1) |
| rec rep(3,1):P4 | | | |
| exec CS | | | |
| rel req(3,1) | | | |
| rep req(3,4) | rec rel(3,1) | rec rel(3,1) | rec rel(3,1) |
| | del req(3,1) | del req(3,1) | del req(3,1) |
| | | | rec rep(3,4):P |
| | | | exec CS |
| | | | rel req(3,4) |
| rec rel(3,4) | rec rel(3,4) | rec rel(3,4) | |
| del req(3,4) | del req(3,4) | del req(3,4) | |

**b) Inspect the steps for starvation.**
Starvation can occur when a process or system is repeatedly denied access to a resume, even though it is capable of progressing Lamport's D-Mutex algorithm is fair and
upholds progress conditions.
Fairness means whichever process requests first should get a chance to execute first in a fair manner. But, in maintaining fairness, starvation could still occur if one process keeps requesting first before other processes or systems.
For the given request model, at first PI requests for resolve. This has no problem and the reply is given by other processes. After this, P2 and P3 concurrently place requests. To avoid no progress P3 reply is sent and after P2 finishes executing Critical Section, it sends reply
for P3. Thus P3 also executes the critical section After this P4 and PI places concurrent requests. If P1 request is done first, then P4 loses the chance to do first causing delay.
Here P4 executes the Critical section first and then Ph executes. Thus all process progress and finish execution properly.

**c) Conclude whether the system suffers due to starvation or not.**
In the given scenario and from the steps of progress, there is no evidence of starvation.
Each process eventually enters the critical section and no process is indefinitely denied access to the resume. The algorithm guarantees progres and ensure fairness in accessing the critical section. There is no indication of a process being consistently denied access to the resource.
The progress from P3 → P4  alone can cause a delay. This is because P1 is given chance
before P4 which means P4 is delayed.
This does not result in no progress or consistant denial for accessing the critical section.
From part b inspection we can conclude that the system does net suffer due to starvation.

**d) Examine the importance of reliability of processes involved in the system.**

Reliability is a crucial factor in the context of distributed systems. When multiple processes and systems are interacting and sharing resources, reliability plays a significant role.
Assuming an unreliable process is present there could be multiple problems related to mutual exclusion and executing critical sections. An unreliable process may not follow the protocol specified leading to confusion and an extensive amount of unnecessary communication messages. The unreliable process may not reply causing starvation or even deadlocks. It can also utilize the resources and may not release it properly according to the protocol. The Lamport's D- Mutex algorithm is based on reliable communication and reliability of the processes. Hence reliability of the processes involved in the system is an important key aspect in the algorithm