SSN COLLEGE OF ENGINEERING, KALAVAKKAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS1711 - MOBILE APPLICATION DEVELOPMENT LAB

Assignment 3

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

ANALYSIS OF STARVATION FOR THE GIVEN RESOURCE REQUEST ORDER

Consider 4 processes P1, P2, P3 and P4 in a distributed system. The Resource request model is
expressed as

P1→ P2 || P3 → P4 || P1.

(Note: → indicates sequential and || indicate concurrent executions)

a. Apply Lamport's D-Mutex algorithm for the given resource request model. (10 Marks)
b. Inspect the steps for the occurrence of starvation (10 Marks)
c. Conclude whether the system suffers due to starvation or not for the given scenario. (10 Marks)
d. Examine the importance of reliability of the processes involved in the system. (10 Marks)

UCS701 - Distributed System
Assignment - 2

Name: P.T. Jayanthan
Roll: 205001049

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

a) Lamport D-Mutex Algorithm

	P_1	P_2	P_3	P_4
queue	(1,1) (2,2) (2,3) (2,1) (3,4)	(1,1) (2,2) (2,3) (3,[illegible]) (3,4)	(1,1) (2,3) (2,2) (3,1) (3,4)	(1,1) (2,2) (2,3) (3,4) (3,[illegible])
	req(1,1)			
		rec req(1,1) rep req(1,1)	rec req(1,1) rep rep(1,1)	rec req(1,1) rep req(1,1)
	rec rep(1,1) P_2 rec rep(1,1) : P_3 rec rep (1,1) P_4 exec CS rel req(1,1)			
		rec rel (1,1) del req(1,1) req(2,2)	rec rel (1,1) del req(1,1) req(2,3)	rec rel(1,1) del req(1,1)
	rec req (2,2) rec req (2,3) rep (2,2) rep (2,3)	rec req (2,3)	rec req (2,2)	rec req (2,2) rec req (2,3) rep req (2,3) rep req (2,3)
		rec rep (2,2) : P_1 rec rep (2,2) : P_4	rec rep (2,3) : P_1 rec rep (2,3) : P_4 rep (2,2)	
		rec rep (2,2) : P_3 exec CS rel req (2,3)		
	rec rel (2,2) del req (2,2)	~~rec rel~~ rep req (2,3)	rec rel (2,2) del req (2,2)	rec rel (2,2) del req (2,2)

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

b. Inspect the steps for the occurrence of starvation

Starvation is a concern when a process is repeatedly denied access to a resource, even though it is capable of progressing. It is important to note that Lamport's D-Mutex algorithm is inherently fair and designed to uphold progress conditions. Fairness in this context implies that the process that requests first should be given the opportunity to execute first in a fair manner. However, even with this fairness, there is still a possibility of starvation if one process persistently requests access before others.

In the given resource request model, the sequence begins with P1 requesting access, and other processes respond appropriately. Subsequently, P2 and P3 concurrently place requests. To ensure progress, P3's request is promptly replied to, and after P2 completes its critical section, it sends a reply for P3. This allows P3 to execute its critical section. The scenario continues with concurrent requests from P4 and P1. If P1's request is granted first, P4 may experience a delay. It's crucial to recognize that P4 executes its critical section before P1, ensuring that all processes make progress and complete their executions properly.

c. Conclude whether the system suffers due to starvation or not for the given scenario.

In the given scenario and based on the sequence of progress, there is no evidence of starvation. Each process eventually enters the critical section, and no process is consistently denied access to the resource. The algorithm guarantees progress and ensures fairness in accessing the critical section. Although there is a potential delay in the P3 → P4 progression, as P1 is given a chance before P4, this delay does not result in a complete lack of progress or consistent denial of access to the critical section. Therefore, after inspecting the steps in part b, we can confidently conclude that the system does not suffer from starvation.

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

Reliability is a fundamental factor in the context of distributed systems, especially when multiple processes and systems interact and share resources. In the presence of unreliable processes, various issues related to mutual exclusion and critical section execution may arise. An unreliable process may fail to follow the specified protocol, leading to confusion and unnecessary communication messages. Furthermore, it may not respond, causing potential starvation or even deadlocks. An unreliable process could also misuse resources and fail to release them properly according to the protocol.

The Lamport's D-Mutex algorithm relies on reliable communication and the trustworthiness of the processes. Ensuring that processes adhere to the protocol is essential for the algorithm to function correctly, maintaining fairness and preventing issues such as starvation or deadlock. Therefore, the reliability of the processes involved in the system is a critical aspect that directly impacts the effectiveness of the algorithm.