DS

Unit: 2 (Introduction)

Mutual Enclusion: Mutual Enclusion:

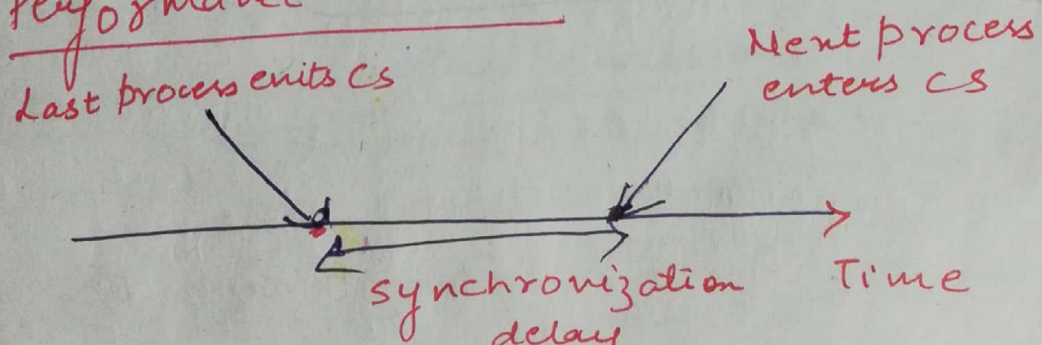
→ we have a critical section (CS) that is a shared resource among a fixed no. of processes. we want to an algo. for the processes to coordinate such that the following requirement are ~~stasi~~ satisfied

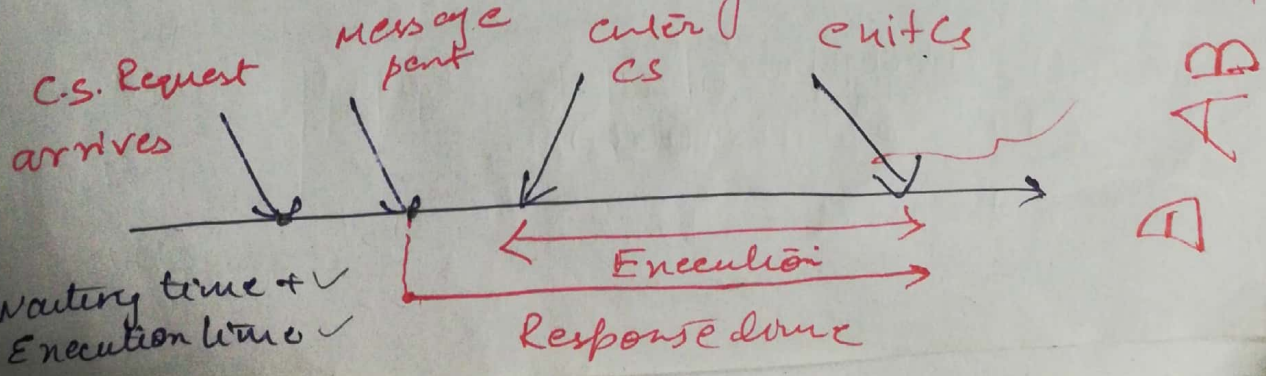
① Safety:- two processes should not get access simultaneously access to the CS.

② Liveness:- Every request for the CS is eventually granted. (no starvation)

③ Fairness:- Requests must be granted in the order they are made.

Performance Metrics





R.T = Waiting time + Execution time

(2)

Objectives:

→ minimize the no. of msg per CS invocation

→ " " synchronization delay (time b/w leaving of the CS by one process & the entering the CS by the next one.

→ minimize the Response time (the time taken b/w the req. msg transmission & exit of CS, includes the CS execution time, E)

→ Maximize the System throughput (rate at which system executes requests for CS)

→ if sd is the synchronization delay
E avg. CS execution time

$$\text{Throughput} = 1/(sd+E)$$

→ no. of msg per CS (minimization)
→ minimise the response time
→ Maximize the System throughput

$$\text{Throughput} = \frac{1}{(sd+E)}$$

sd → Synchronization delay
E → execution time

Low and High Load Performance

Loading conditions

Low Load

↳ there is seldom more than one request for M.E simultaneously.

High Load.

↳ there is always a pending request for mutual exclusion at a site.

↳ Site is seldom in idle state under high load.

Best & Worst Case Performace

↳ High Load.

Response time

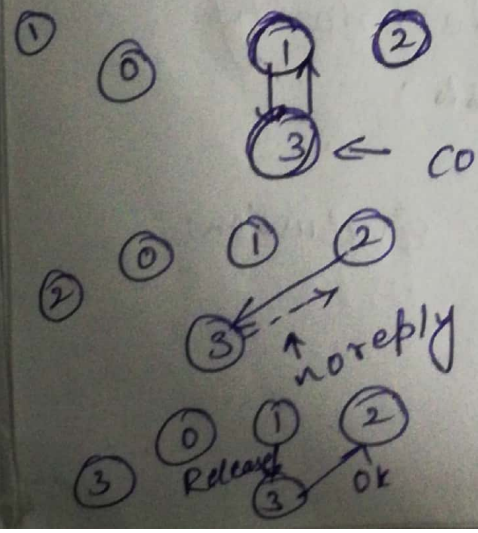
= 2T + E → execution time

↓ avg msg delay

Simple soln to mutual Exclusion

A site called a control site

↓ assigned the task of granting permission for the CS execution.



centralized soln (Drawback)

1. Single point of failure
2. Comm^n link near the control site are likely to be congested & became a bottleneck.

(4)

~~Third~~

(3) Synchronization delay of this algorithm is 2T

→ first release permission to the control site

→ then control site should grant permission to the next site to execute the CS.

$$\text{Throughput} = \frac{1}{(2T+E)}$$

(Non token Based Algorithm)

⇒ a site communicates with a set of other sites to arbitrate who should execute the CS next

For a site S_i.

request set R_i contains ids of all those sites from which site S_i must acquire permission before entering the CS.

⇒ use timestamps to order requests for the CS

⇒ logical clocks are maintained and updated according to Lamport's scheme.

⇒ Each request for the CS get a timestamp, & smaller timestamp requests have priority over lager timestamp requests.

eg: ① Lamport Algo for Mutual Exclusion.

② Ricart Agrawala Algo

③ Maekawa's Algorithm

Token Based Algorithms

→ a unique token is shared among all sites.
→ a site is allowed to enter its CS if it possesses the token.
→ depending upon the way a site carries out its search for the token.
→ various token based algo. are there.
→ token based algorithm use sequence nos instead of timestamps.
→ Every request for the token contain a sequence no. & the sequence numbers of sites advance independently.
→ A site increments its sequence no. counter every time it makes a request for the token.
→ seq. no. is use to distinguish b/w old & current request.

Algo

1. Suzuki-Kasami's Broadcast algo.
2. Singhal's Heuristic algo.
3. Raymond's Tree Based.

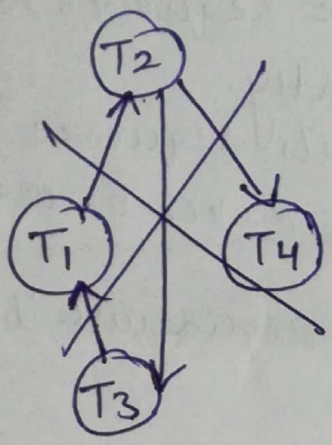
(6) Distributed Deadlock Detection :-

P_1
P_2
P_3

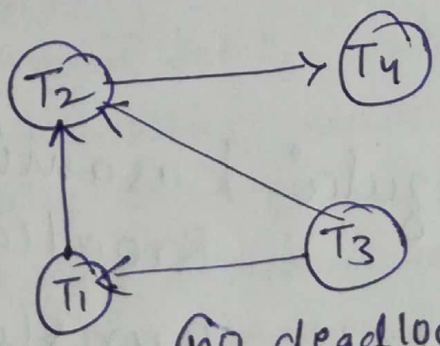
⇒ In D.S. even when there is no deadlock within a single site, there may be inter site deadlock.

Local wait for Graph
Global wait for Graph

at site S_1



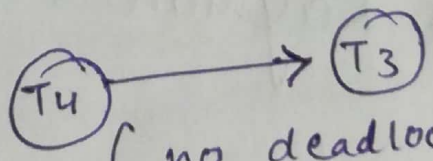
Local wait for graph



at site 1

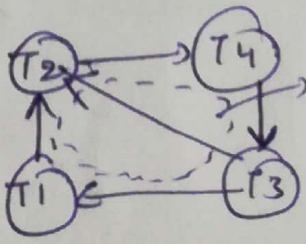
(no deadlock) at S_1

at site 2



(no deadlock) at S_2

Local Graph



Cycle deadlock is there.

(Global wait for graph)

Deadlock Handling ~~Strate~~ Strategies in D.S (7)

1. Deadlock Prevention ✓
2. ,, Avoidance ✓
3. Deadlock Detection. ✓

ss | Resource Deadlocks