Unit 2

Distributed Mutual Exclusion

Mutual Exclusion in single Computer System Vs distributed System

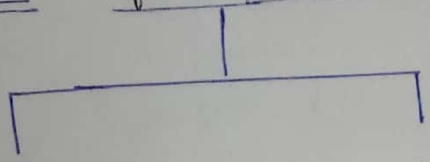
= In single computer systems, the status of a shared resource and the status of user is available in the shared Memory. and solutions to the Mutual Exclusion problem can be easily implemented using Shared Variables (eg. Semaphores)

⇒ In D.S. both the shared resources and the users may be distributed and shared memory does not exist. Approaches based on shared variables are not applicable to D.S. and approaches based on message passing must be used

⇒ The problem of M.E. become much more complex in D.S because of the lack of both shared memory and a common physical clock and because of unpredictable message delay

Scanned by CamScanner

Classification of Mutual Exclusion Algo.



Non Token Based

⇒ These algo. requires 2 or more successive round of Msg exchanges among the site

Token Based Algorithm.

= In these algorithm, a unique token is shared among the site
A site allowed to enter the cs if it possesses the token. It continues to hold the token until the execution of cs is over.

Requirement of Mutual Exclusion

The primary objective is to guarantee that only one request accesses the cs at a time.
The following characteristics are considered important in a Mutual exclusion Algo.

(1) Freedom from Deadlocks 2 or more sites should not endlessly wait for msg. that will never arrive

(2) Freedom from Starvation. A site should not be forced to wait indefinitely to execute CS. while other sites are repeatedly executing CS.

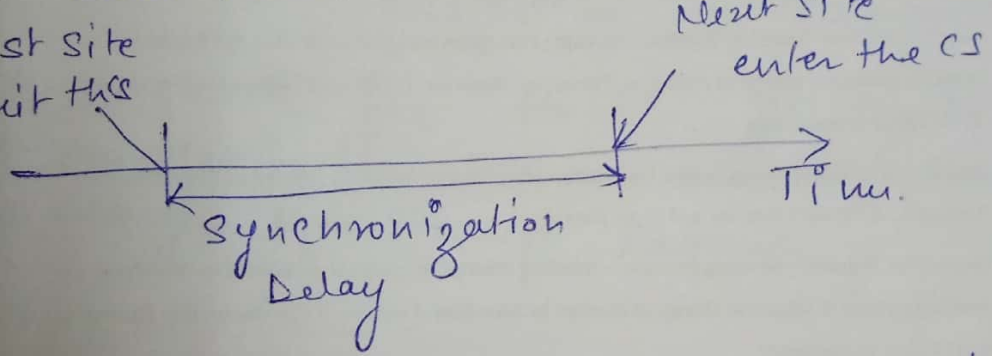
Scanned by CamScanner

3) Fairness: Requests must be executed in the order they are made.

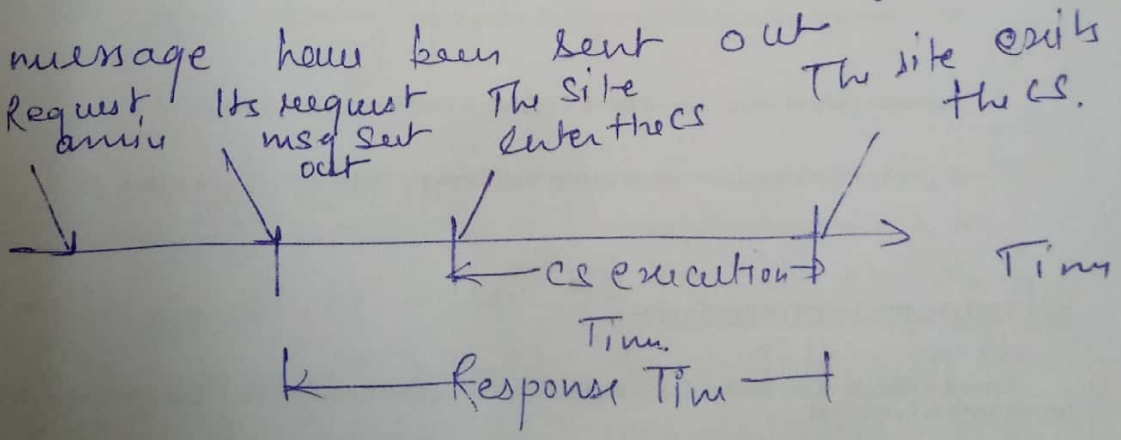
4) Fault Tolerance: A mutual exclusion algo. is fault tolerant if in the wake of failure it can reorganize itself so that it continues to function without any disruption

Performance Measure:

(1) Synchronization Delay.: Time required after a site leave the CS and before the next site enter the CS.



(2) Response Time: Time interval a request wait for its CS execution to over after its request message have been sent out



3) Throughput = the rate at which the system executes requests for the CS.

if Sd = synchronization Delay
E = avg CS execution time

Throughput = 1/(Sd + E).

17

Scanned by CamScanner

Non Token Based Algorithm

In non token Based Algorithm, a site communicate with a set of other sites to arbitrate who should execute the CS next.

⇒ for a site Si, request set Ri contains ids of all those sites from which site Si must acquire permission before entering the CS.

⇒ Non Token Based mutual exclusion algo use timestamps to order requests for the CS.
In all these algo. logical clocks are maintained and updated according to Lamport's scheme

Each request for the CS gets a timestamp and smaller timestamp requests have priority over larger timestamp requests

Lamport's Algorithm

This algorithm requires messages to be delivered in the FIFO order b/w every pair of sites.

Algo:

Request the Critical Section:

1) When a site Si wants to enter the CS, it sends a REQUEST(tsi, i) message to all the site in its request set Ri, and places the request on request_queue((tsi, i) is the timestamp of the request)

(2) When a site S_j receives the REQUEST (ts_i, i) message from site S_i, it returns a timestamped REPLY message to S_i and places site S_i's request on request_queue.

Executing the C.S.

Site S_i enters the CS when the 2 following conditions hold.

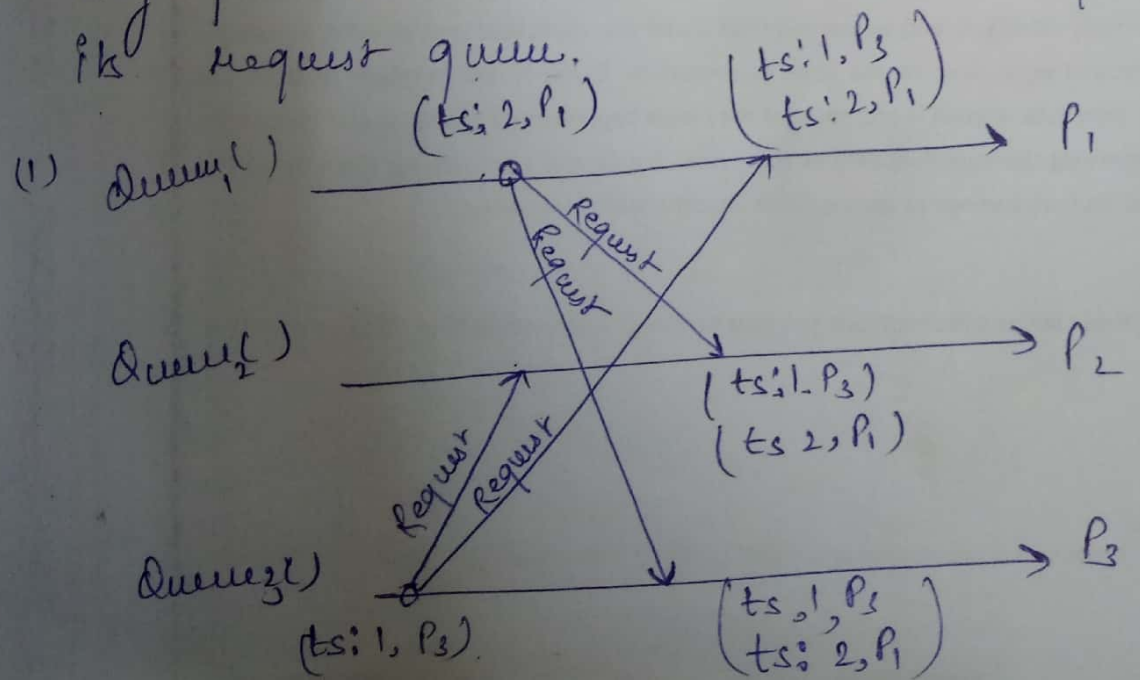
$[L_1]$: S_i has received a message with timestamp larger than (ts_i, i) from all other sites.

$[L_2]$: S_i's request is at the top of request_queue.

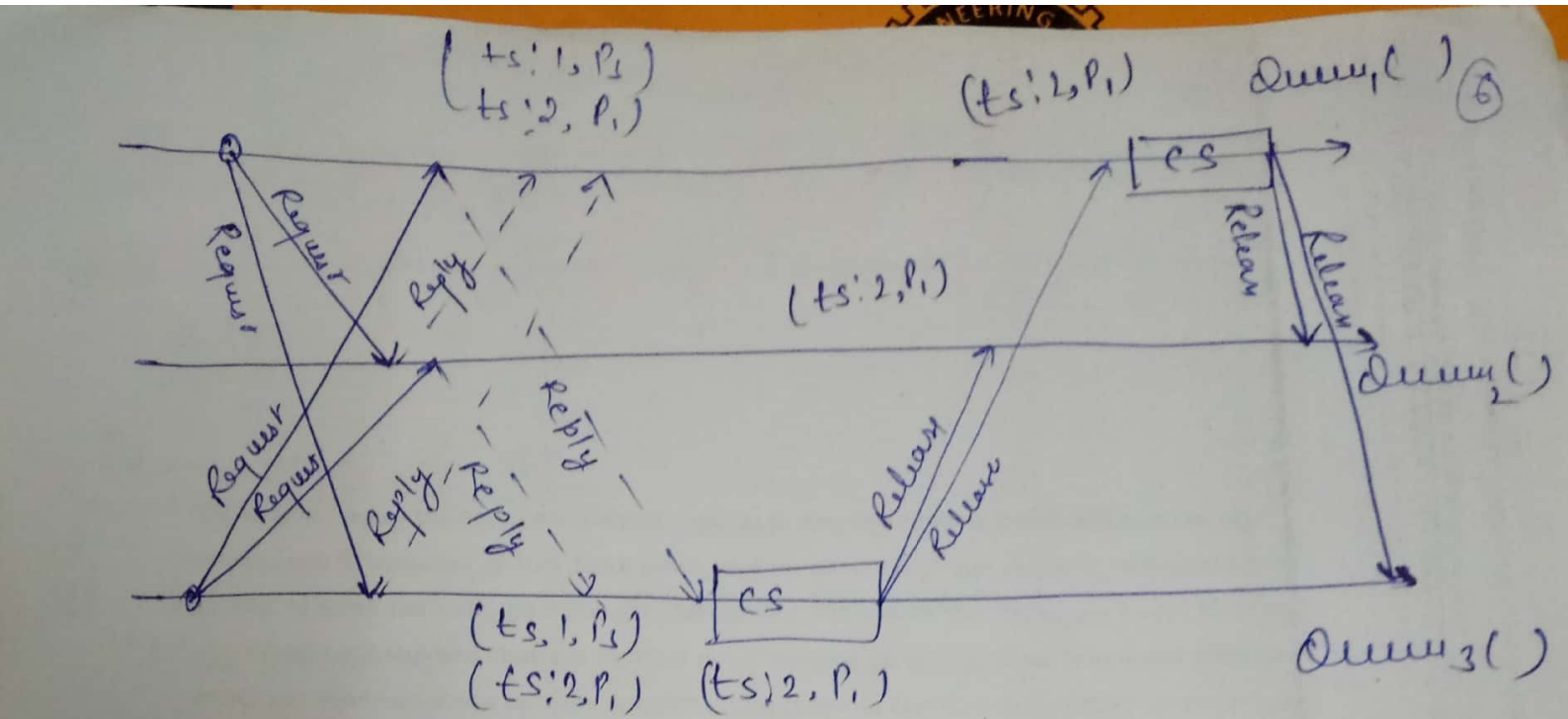
Releasing the Critical Section

(1) Site S_i upon exiting the CS, removes its request from top of its request queue and sends a timestamped RELEASE message to all the sites in its request set.

(2) When a site S_j ~~release~~ receives a RELEASE msg from site S_i, it removes S_i's request from its request queue.



Scanned by CamScanner



Lamport Algorithm requires $3(N-1)$ message per CS invocation.

$(N-1)$ Request

$(N-1)$ Reply

$(N-1)$ Release message

Synchronization delay in the algo. is T.

Scanned by CamScanner

The RICART AGRAWALA Algorithm

⇒ It is an optimigation of Lamport's algorithm

Algo

Requesting the C.S.

1) When a site S_i wants to enter the CS, it sends a timestamped REQUEST message to all the sites in its request set.

2) When a site S_j receives a REQUEST message from site S_i, it sends a REPLY message to site S_i. if site S_j is neither requesting nor executing the CS or if the site S_j is requesting and S_i's request's timestamp is smaller than site S_j's own request's timestamp. The request is deferred otherwise

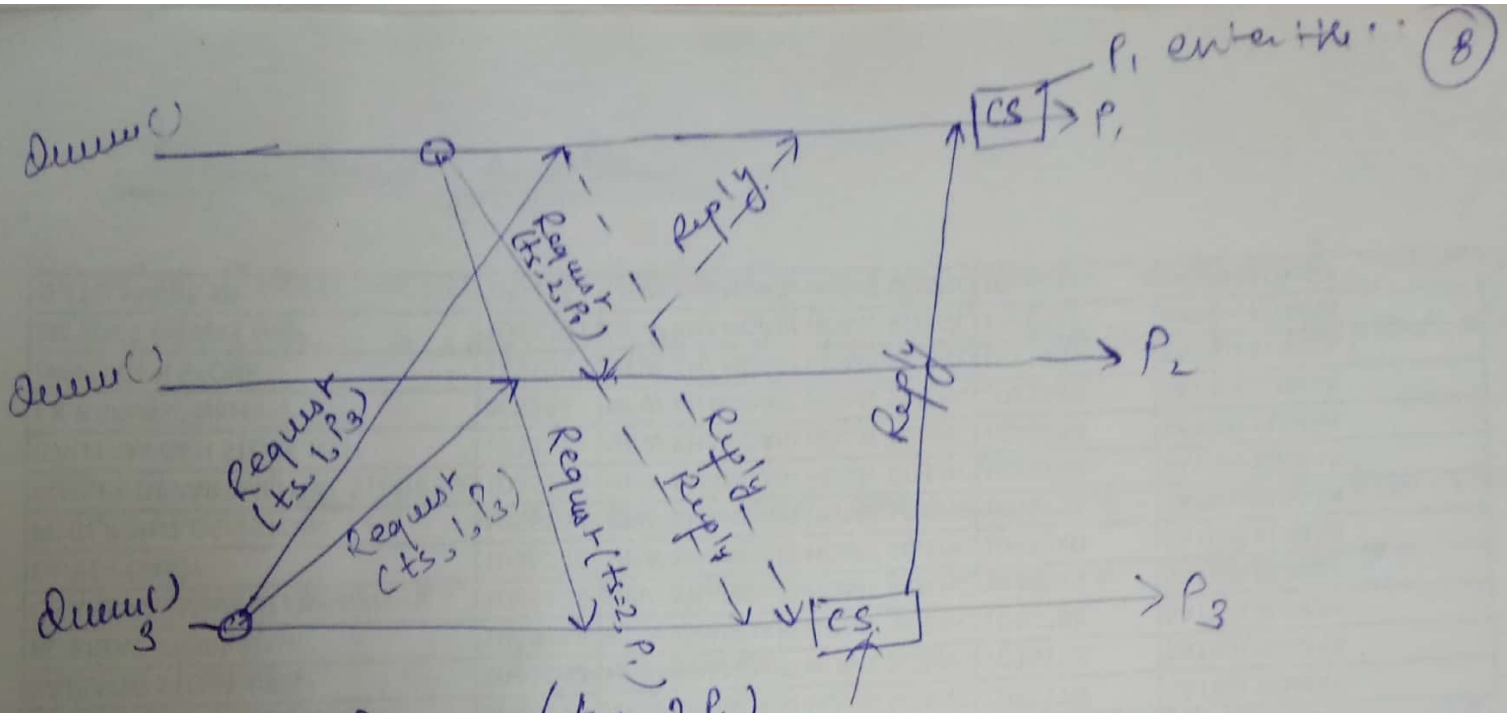
Executing the CS

3) site S_i enter the CS. after it has received REQUEST message from all the sites in its request set.

Releasing the C.S.

4) when a site S_i exits the CS. It sends REPLY message to all the ~~deff~~ deferred request

Scanned by CamScanner



The Ricart Agrawala algo requires $2(N-1)$ messages per CS. $(N-1)$ Request, $(N-1)$ Reply messag.

Synchronization delay = T.

Scanned by CamScanner

Token Based Algorithm

* In Token based Algorithm, a unique token is shared among all sites. A site is allowed to enter its CS if it possesses the token.
* Token based algorithm use sequence no. instead timestamps. Every request for the token contains a sequence no. instead of timestamps.
* Every request for the token contain a sequence no.
* A site increments its sequence no. counter every time it makes a request. for the token
  A primary function of the sequence no is to distinguish b/w old and current requests.

Suzuki Kasami Broad Cast Algorithm

* If a site attempting to enter the CS. does not have the token. It broad cast a REQUEST Message. for the token to all the other sites.
* A site that possesses the token send it to the requesting site upon reciving its REQUEST. message.
* If a site recives a REQUEST. message when it is executing the CS. It sends the token only after it has exited the CS.
* A site holding the token can enter its CS. repeatedly until it sends the token to some other site.

Scanned by CamScanner

Algorithm

The token: → Queue (FIFO) Q of requesting processes

LN[1..n]: Sequence no of request that j executed most recently.

Requesting the C.S

(1) If the requesting site Si does not have the token then it Increment its sequence no. RN[i], and send a ~~Send~~ REQUEST(i, sn) message to all other sites.

(2) When a site Sj receives this message it sets RNj[i] to max(RNj[i], sn). If Sj has the idle token, then it send the token to Si. if RNj[i] = LN[i]+1

Executing the CS :

3) Site Si executes the CS when it has received the token

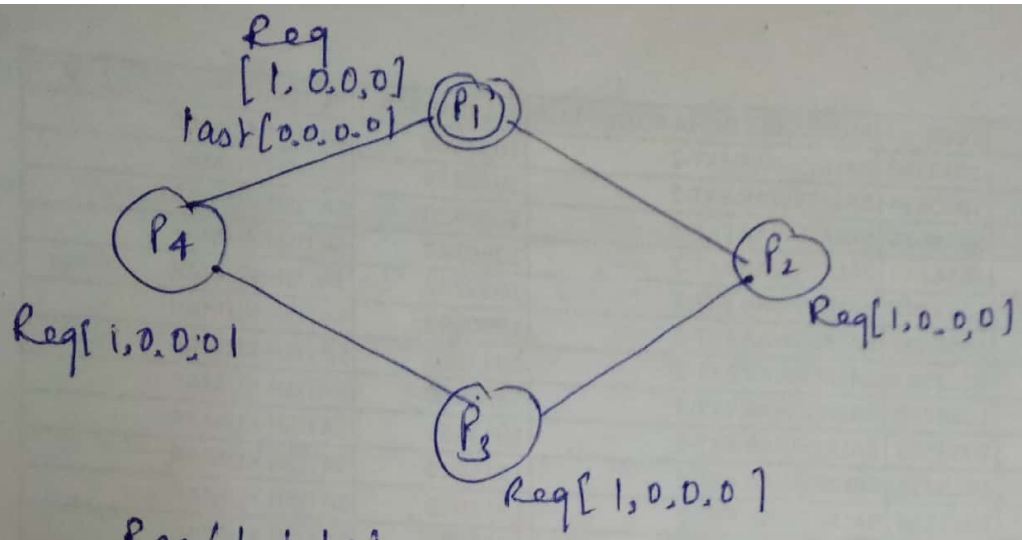
Releasing the critical Section:

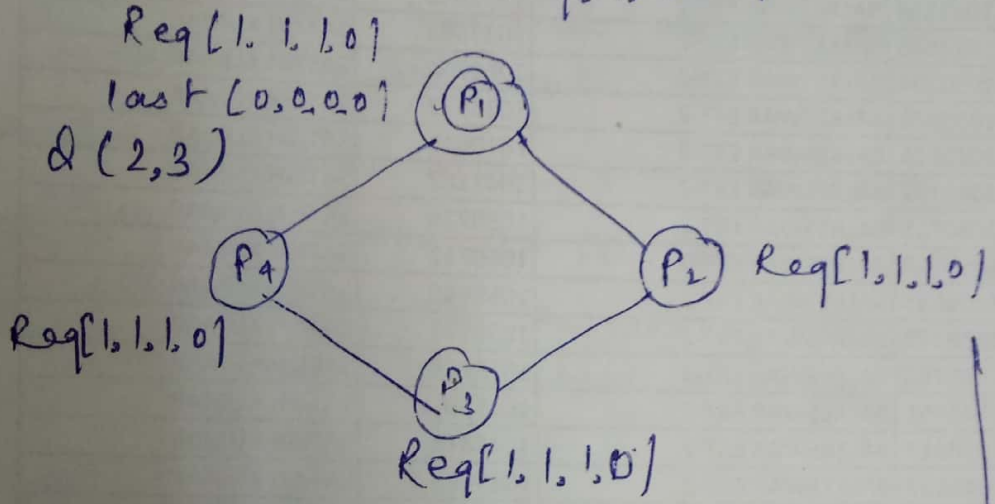
4) It sets LN[i] element of the token array equal to RN i[i]

5) for every site Sj whose ID is not in the token queue it appends its ID to the token queue if RNi[j] = LN[j]+1

6) If token queue is non empty after the above update then it delete the top site ID from the queue and send the token to the site indicated by the ID

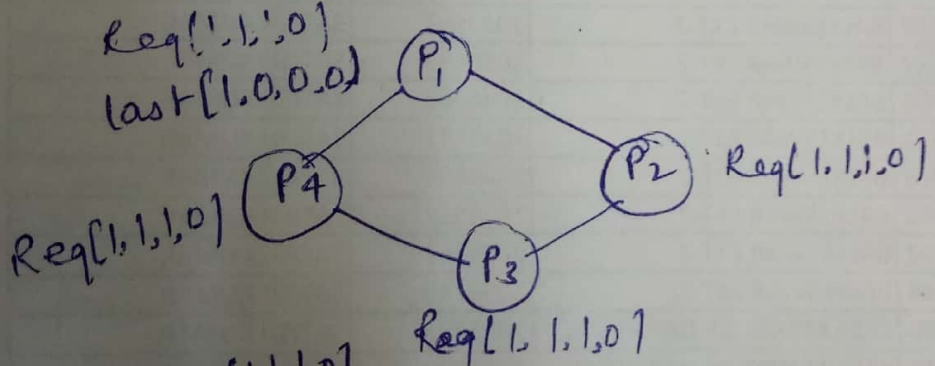
Scanned by CamScanner

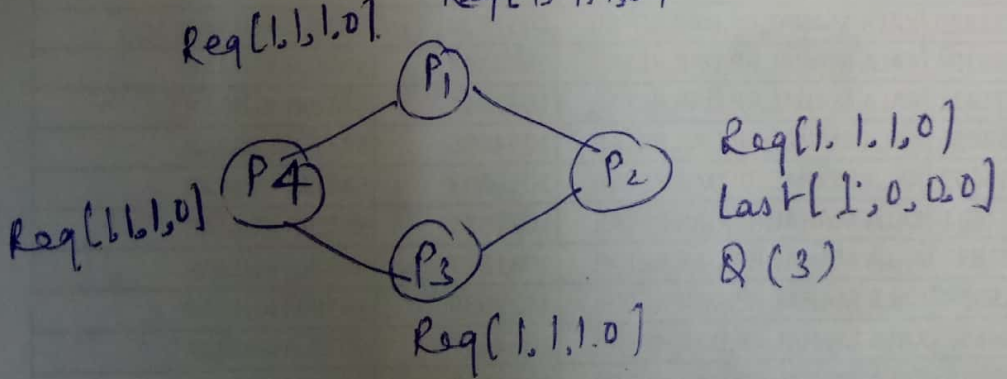


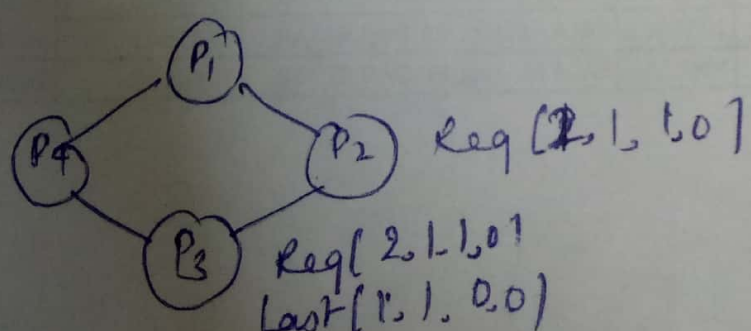


Synchronization Delay = 0

The algo requires 0 or N message per CS invocation







Scanned by CamScanner

Raymond's Tree Based Algorithm:

⇒ In this algo. sites are logically arranged as a directed tree. such that the edge of the tree are assigned directions toward the site that has token.

⇒ Every site has a local variable holder that points to an immediate neighbor node on a directed path to the root node

⇒ At root site, holder points to itself.

⇒ Every site keeps a FIFO Queue, called request-q.

⇒ which stores the requests of those neighboring sites that have sent a request to this site

Algo:

Requesting the C.S.

(1) When a site wants to enter the CS. It send a Request Message to the node along the directed path to the root, provided it does not hold the token and its request-q is empty. It then add its request to its Request-q.

(2) When a site on the path receives this message it places the REQUEST in request-q and send a REQUEST msg along the directed path to the root provided it has not sent out a REQUEST msg on its outgoing edge

Scanned by CamScanner

3) When a root site receives a REQUEST message, it sends the token to the site from which it received the request message ~~on its outgoing edge~~ and sets its holder variable to point at that site

4) When a site receives the token it deletes the top entry from its request_q, sends the token to the site indicated in this entry, and sets its holder variable to point at that self

Executing the CS

5) A site enters the CS when it receives the token and its own entry is at the top of its queue. Then, the site deletes the top entry from its request-q and enters the CS.

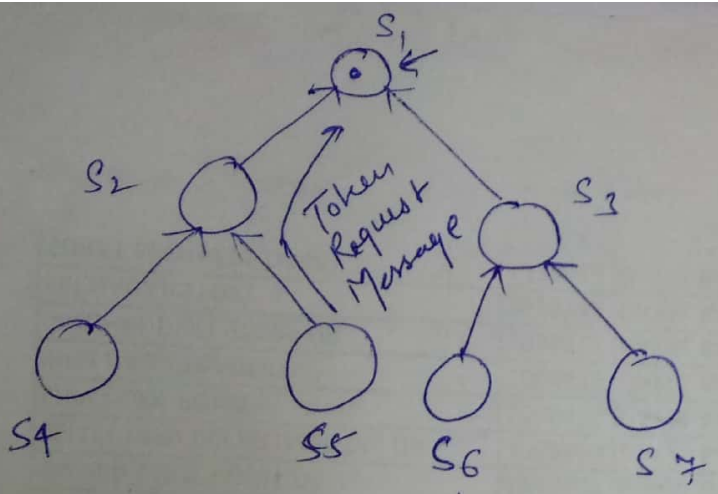
6) Releasing the CS After a site has finished execution of the CS it takes the following actions:

(a) If its request-q is non empty then it deletes the top entry from its request-q, sends the token to that site and set its holder variable to point at that site

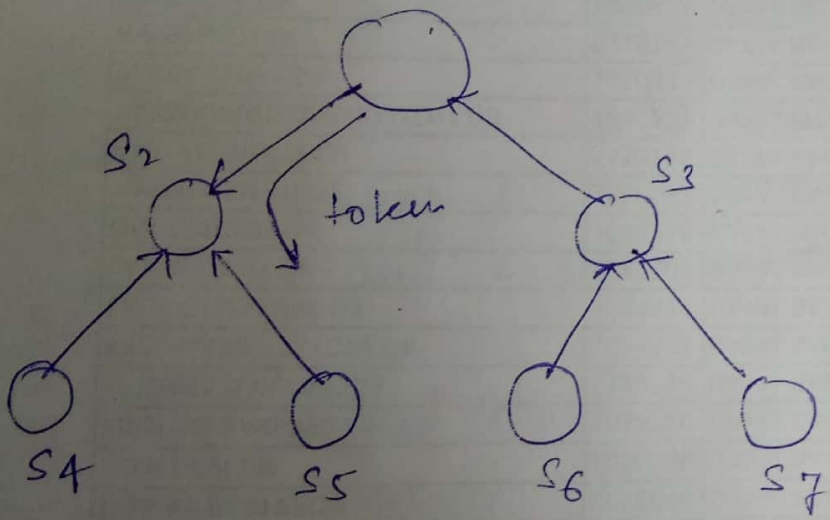
(8) ~~If the~~

⇒ Algo is free from deadlocks

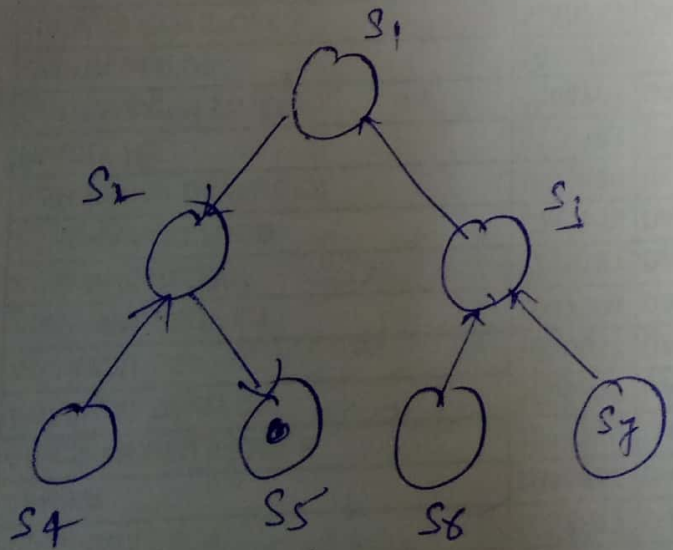
Scanned by CamScanner



Site S5 is requesting the token



The token is in transit to S5



S5 received the token

Scanned by CamScanner

Distributed Deadlock Detection

System Model: The problem of deadlocks has been generally studied in D.S. under the following Model

- The system have only reusable resources.
- Processes are allowed only exclusive access to resources
- There is only one copy of each resource

A process can be in 2 states – Running or Blocked In the running state, a process has all the needed resources and is either executing or is ready for execution.
In the blocked state, a process is waiting to acquire some resources.

Resource Vs. Communication Deadlock:

Two Types of deadlock ⟶ Resource Deadlock
⟶ Communication Deadlock

In Resource Deadlock, processes can simultaneously wait for several resources and can not proceed until they have acquired all those resources.

In Communication Deadlock; processes wait to communicate with other processes among a set of processes. A waiting process can be unblocked on receiving a communication from any ~~other~~ one of these processes.

Scanned by CamScanner

Deadlock Handling Strategies in distributed System

Deadlock Prevention: Deadlock prevention is commonly achieved by either having a process acquire all the needed resource simultaneously before it begins execution. or by preempting a process that hold the needed resource

Deadlock Avoidance In the deadlock avoidance approach to D.S., a resource is granted to a process if the resulting global ~~state~~ System state is safe.

Beog of the following problems, deadlock avoidance can be seen as impractical in DS.

(1) Every site has to maintain information on the global state of System. which translate into huge storage ~~information. Can the global Data of the system~~ requirements and extensive communication cost.

(2) The process of checking for a safe global state must be mutually exclusive.

(3) Due to the large no of process and resources it will Computationally expensive to check for a safe state.

Deadlock Detection: Deadlock Detection in DS has 2 favorable Cond^n.

(1) Once a cycle is formed in the WFG. it persist until it is detected and broken

(2) Cycle detection can proceed. concurrently with the normal activity of a System.

Scanned by CamScanner

Resolution: Deadlock resolution involves breaking existing wait for dependencies in the system WFG to solve the deadlock.
It involves Rolling back one or more processes that are deadlocked and assign their resources to blocked processes in the deadlock so that they can resume execution

Distributed Deadlock Detection:

⇒ occurance of a cycle b/w Resource and process in WFG is known as deadlock Detection
⇒ Deadlock Detection is the most popular strategy for handling deadlocks in D.S.

We can Control deadlock detection in 3 types

(1) Centralized
(2) Distributed
(3) Hierarchical

(1) Centralized Deadlock Detection Algo.

- In this only one node is responsible for building and analyzing a real WFG for a cycle
- Each site maintain its WFG and update Control site periodically or on request
- A central control site constructs the global WFG and searches for cycle.
- Control site maintain WFG continuously
- All site request resources and release resources by sending corresponding message to control site

Scanned by CamScanner

*) Control site update WFG for each request and release of resources

*) for every request edge added to WFG and control site checks WFG for deadlock

Disadvantages: single point of failure and congestion

Distributed Deadlock Detection Algorithm:

In this each node participates equally in detecting the deadlock --- WFG

Principle • All sites are responsible for detecting a global deadlock

• Global state graph distributed over many sites. Several of them participate in detection

• Detection initiated when a process suspected to be deadlocked

Advantage; No single point of failure and no congestion

Disadvantage: Difficult to implement

Hierarchial Deadlock Detection Algorithm:

In this, Nodes are organized or arranged in a tree pattern

Principle: In this the sites are organized in a tree structure with one site at root of the tree.

Each node has information about the dependent Node.

Deadlock is detected by the node that is the common ancestor of all sites which have resource allocations

Scanned by CamScanner

in conflict

- Deadlock is detected at the lowest level

In this sets of nodes are required to report periodically to a control site node but control sites are organized in a tree structure

- The Master control site forms the root of the tree with leaf nodes having no control responsibility and interior nodes serving as controllers for their branch

Centralized Deadlock Detection Algorithm

The Completely Centralized Algorithm:

⇒ The completely centralized algo. is the simplest centralized deadlock detection algorithm. wherein a designed site called the control site. maintains the WFG of the entire system. and checks it for the existence of deadlock cycles.

⇒ All sites request and release resources by sending request resource and release resource message to control site.

⇒ When a control site receive a request resource or a release resource message it updates its WFG The control site . Check the WFG for deadlocks whenever a request edge is added to the WFG.

Scanned by CamScanner

⇒ This algo is simple and easy to implement.
⇒ but it is also highly inefficient bcoz all resource acquisition and release request must go through the control site. even when the resource is local
⇒ large delays.
⇒ large Communication overhead
⇒ Reliability is book because of control site

The Ho Ramamoorthy Algorithms

Two phase Model (Can be for AND or OR Model)

⇒ In this each site has a status Table of locked and waited resource
⇒ The control site will periodically ask for this table from each node
⇒ The control node will search for cycles and if found will request, the table again from each node
⇒ Only common info. in both reports will be analyzed for confirmation of a cycle

One Phase (Can be for AND or OR Model)

- In this each site keeps 2 tables for all local process; Process Status
  ; Resource Status Table
- The control site will periodically ask for these table via msg from each node
- The control site will build and analyze the WFG. looking for cycles and resolving them when found

Scanned by CamScanner

In One phase algo. does not detect the ~~dead~~ false deadlock because it eliminates the inconsistency in state info.

⇒ The 1 phase algo is faster and requires fewer msg as compared to 2 phase algo

Distributed Deadlock Detection Algo.

In Distributed deadlock detection algo. all sites collectively cooperate to detect a cycle in the state graph

A Path Pushing Algorithms:

⇒ In this Path information sent from waiting node to blocking node

⇒ This approach deal with Transactions Each transaction may have sub transactions but they executes in sequentially manner

⇒ In this all the transactions are totally ordered or sequentially.

⇒ Path info sent from waiting node to blocking node

⇒ In this WFG. Constructed by disseminating dependency Sequences.

- Each node builds a WFG based on local info and info. from other site.
- Detect and resolves local deadlocks
- Transmit to other sites deadlock info in form of waiting path

Scanned by CamScanner

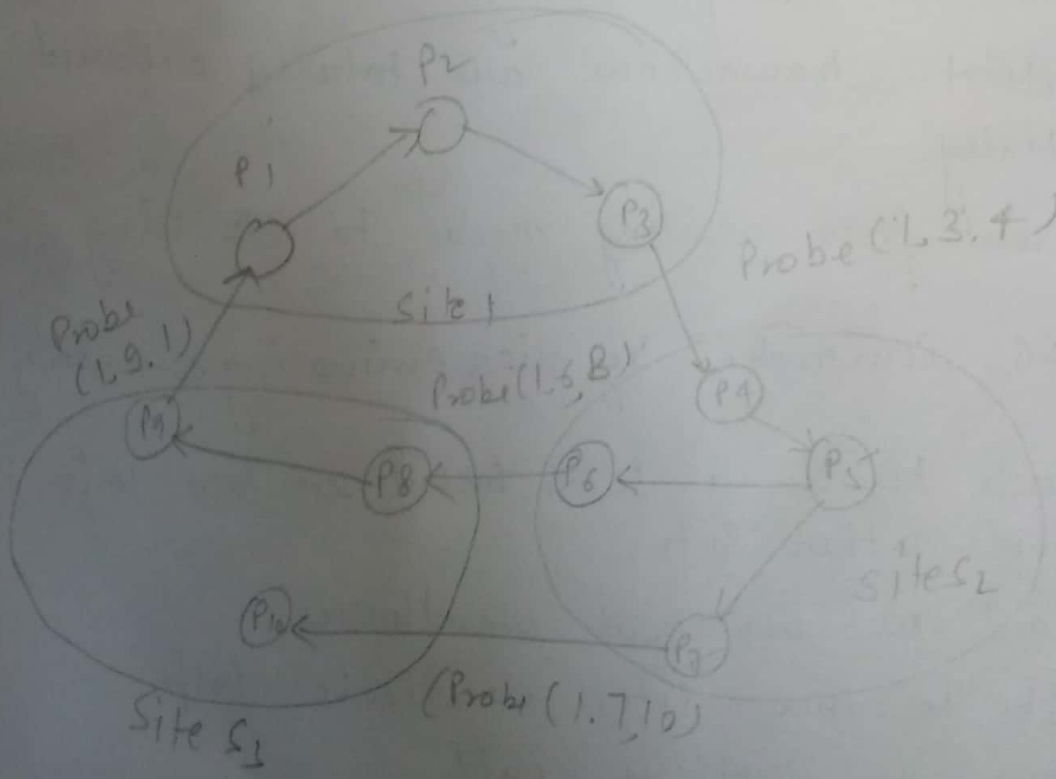
Edge chasing Algorithm:

⇒ In this message are sent along graph edges.

- Special message (probes) sent along edges of WFG to dect a cycle
- When Blocked procss recive msg. resends it on its outgoing edge of WFG
- When a procss recive a msg it initiated declare deadlock

⇒ Process send special msg

⇒ A blocked procss recive msg and circulates it along its outgoing dependency edge



Scanned by CamScanner

Hierarchical Deadlock Detection Algo.

⇒ Sites are arranged in hierarchical fashion and a site is responsible for detecting deadlock involving only its Childreen sites.

The Menasce Muntz Algorithm

⇒ In the hierarchical deadlock detection algorithm of Menasce Muntz, all the controller are arranged in tree fashion. (Controller manages a resource or it responsible for deadlock detection)

⇒ The Controller at the bottom most Level (Leaf Controller Manages resource and others. are responsible for deadlock detection.

=A Leaf Controller maintain a part of the global TWF Graph. Concerned with the allocation of resources at that Leaf Controller

⇒ whenever a changes occurs in a controller TWF Graph due to a resource allocation. wait or release, it is propagated to its parent Controller

The parent Controller makes changes in its TFG Graph searches for cycle and propagate the changes upward if necessary.

A non Leaf Controller can recive up to date info. Concerning the TWF Graph of its Childreen continuously.

Scanned by CamScanner

Ho Ramamoorthy Algo.

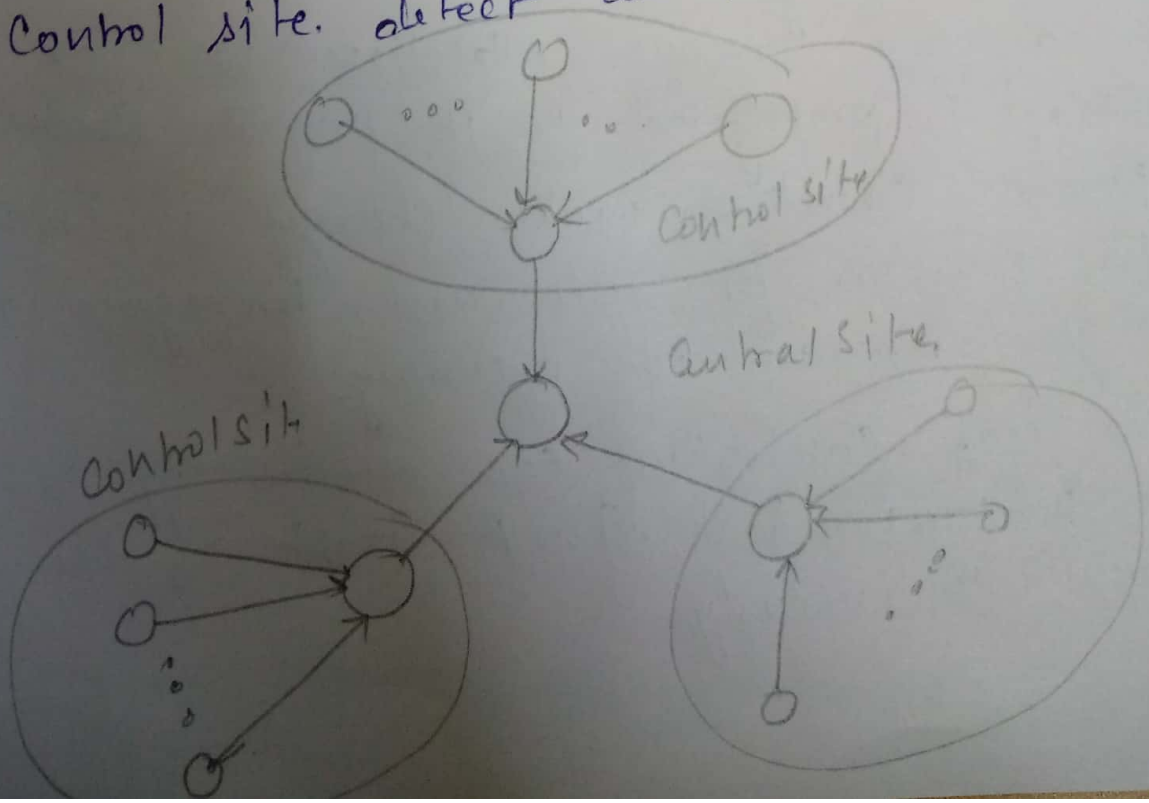
⇒ In the hierarchical algo. of Ho and Ramamoorthy sites are grouped into several disjoint clusters. Periodically a site is chosen as a central control site. which dynamically chooses a control site for each cluster.

⇒ The Central Control site requests from every control site their intercluster transaction status Info. and wait for relations.

⇒ Control site collects status table from all the sites in its cluster and applies the one phase deadlock detection algo. to detect all deadlock involving only intercluster transactions

⇒ It sends intercluster transaction status info. and wait for relations. to the central . Control sites.

⇒ ~~If~~ Control site ~~deposits~~ detects all dead locks located in its cluster and the Central Control site. detect all intercluster deadlocks



Scanned by CamScanner