SSN COLLEGE OF ENGINEERING, KALAVAKKAM – 603 110
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
B.E. Computer Science and Engineering
CS6601 DISTRIBUTED SYSTEMS

Date: 05.02.2018, 8.00-9.30 AM **UNIT TEST – 2** **Max. Marks: 50**
Academic Year: 2017-2018 EVEN **Batch: 2015-2019**
Semester: 6. **Faculty: Mr. H.Shahul Hamead & Ms. Y.V.Lokeswari**

PART -A (5X2 =10 Marks)

1. For a total of 10 processors in the system, how many number of faulty processors are allowed to arrive at a consensus. **(K3, CO4)**
   Ans: 3 faulty processors N> 3*f. f=3 for N=10
2. Differentiate between objects and components. **(K2, CO2)**
   Ans: Objects (Accessed through Interfaces)
   Components (Accessed through Interfaces and assumptions made on other components to fulfill its function).
3. What is Safety, Liveness and Fairness property in Mutual Exclusion **(K2, CO4)**
   Ans: Safety Property:
   At any instant, only one process can execute the critical section.
   Liveness Property:
   This property states the absence of deadlock and starvation. Two or more sites should not endlessly wait for messages which will never arrive.
   Fairness:
   Each process gets a fair chance to execute the CS. Fairness property generally means the CS execution requests are executed in the order of their arrival (time is determined by a logical clock) in the system
4. What is a P2P system and how it is different from Client-Server system. **(K2, CO2)**
   Ans: P2P system, any system is free to join or leave the network at any time. In Client - Server system, communication is designated to one to one communication. Communication channel is restricted only for agreed client and sever systems.
5. What is a mobile code and give one example. **(K2, CO2)**
   Ans: a running program (including both code and data) that travels from one computer to another in a network carrying out a task on someone's behalf. Eg: Java Applet.

PART – B (24 Marks)

6. Elaborate about the architectural elements in distributed systems. (16) **(K2, CO2)**
   Ans:
   1. Communicating Entities
   1.Processes
   2.Nodes
   3.Threads
   4.Objects (Accessed through Interfaces)
   5.Components (Accessed through Interfaces and assumptions made on other components to fulfill its function).
   6.Web services (S/W application that interacts with other software agents using XML messages and Internet protocols)
   2. Communicating Paradigm
   1.Interprocess communication

2. Remote Invocation

1. Request-Reply Protocols
2. Remote Procedure calls (RPC)
3. Remote Method Invocation (RMI)

3. Indirect Communication
(Space and Time Uncoupling)

1. Group Communication (One – to –many, Each group is addressed by a Group Identifier)

2. Publish-Subscribe Systems (Producers distribute information to large number of similar consumers)

3. Message Queues (Producer process send messages to Queue and consumer process access messages from queues)

4. Tuple Spaces (Processes place arbitrary items (tuples) in tuple space and other processes can read or remove tuples)

5. Distributed Shared Memory (Abstraction for sharing data between processes)

3.Roles and Responsibilities

1. Process takes roles as either client or server
2. Servers may in turn be clients of other servers.
3. Roles and Responsibilities

1. Peer – to –peers : All processes involved in task , interact cooperatively as peers.
2. Service provided by multiple servers

4. Placement

How entities (objects, services) map on the underlying physical distributed infrastructure

Placement needs to take into account the following

- The patterns of communication between entities,
- The reliability of given machines and their current loading,
- The quality of communication between different machines

OR

7. Discuss about the failure model and security model in distributed system. (8+8)
(K2, CO2)

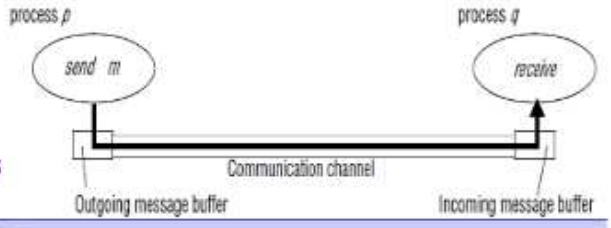
Ans: Failure Model

3. Fundamental Model

2. Failure Model

This defines the ways in which failure may occur.

Omission Failures and Arbitrary Failures



Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send* operation but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times or commit omissions; a process may stop or take an incorrect step.

Timing Failure

Masking Failure

•Multiple servers that hold replicas of data

•Can enable a new service to be designed to mask the failure of the components.

•A service masks a failure either by hiding it altogether or by converting it into a more acceptable type of failure.

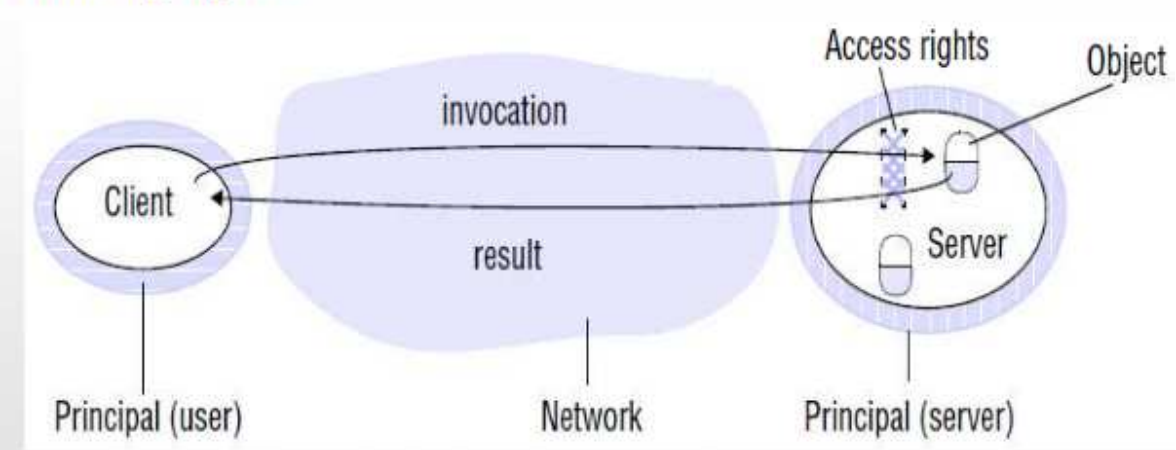
Reliability: Validity and Integrity.

Security Model

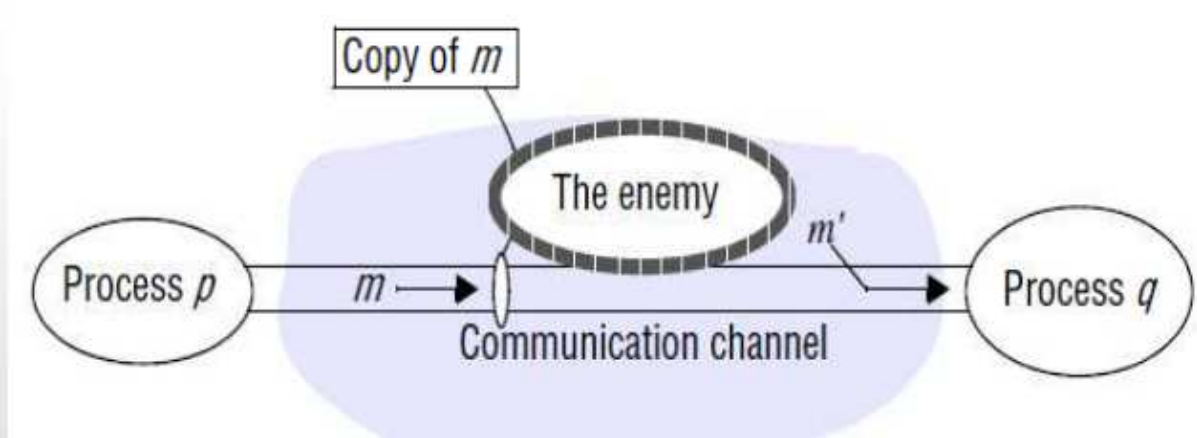
3. Security Model

- securing the processes and the channels
- Protecting the objects that they encapsulate against unauthorized access

Protecting objects



Enemy : is capable of sending any message to any process and reading or copying any message sent between a pair of processes



Enemy: Causes threats to Processes and Communication Channels

•Threats to Processes

–Servers

–Clients

•Threats to Communication Channels

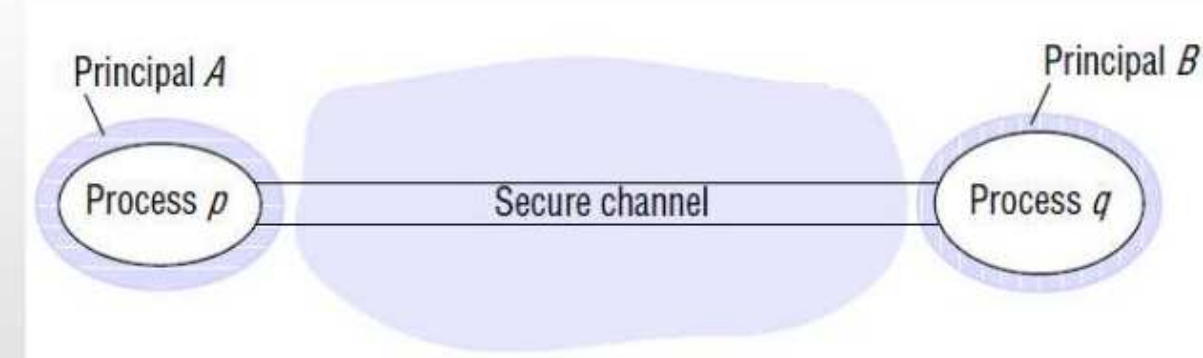
–Copy alter or inject messages that travel across network

–Save copies of message and replay them at later time.

•Denial of Service.

Mobile Code.

- **Defeating Security Threats**
  - **Cryptography**
  - **Authentication**
  - **Secure Channels (Protocols like VPN and SSL)**



8. (i) Discuss in brief about generations of distributed systems in Physical model. (4)

(K2, CO2)

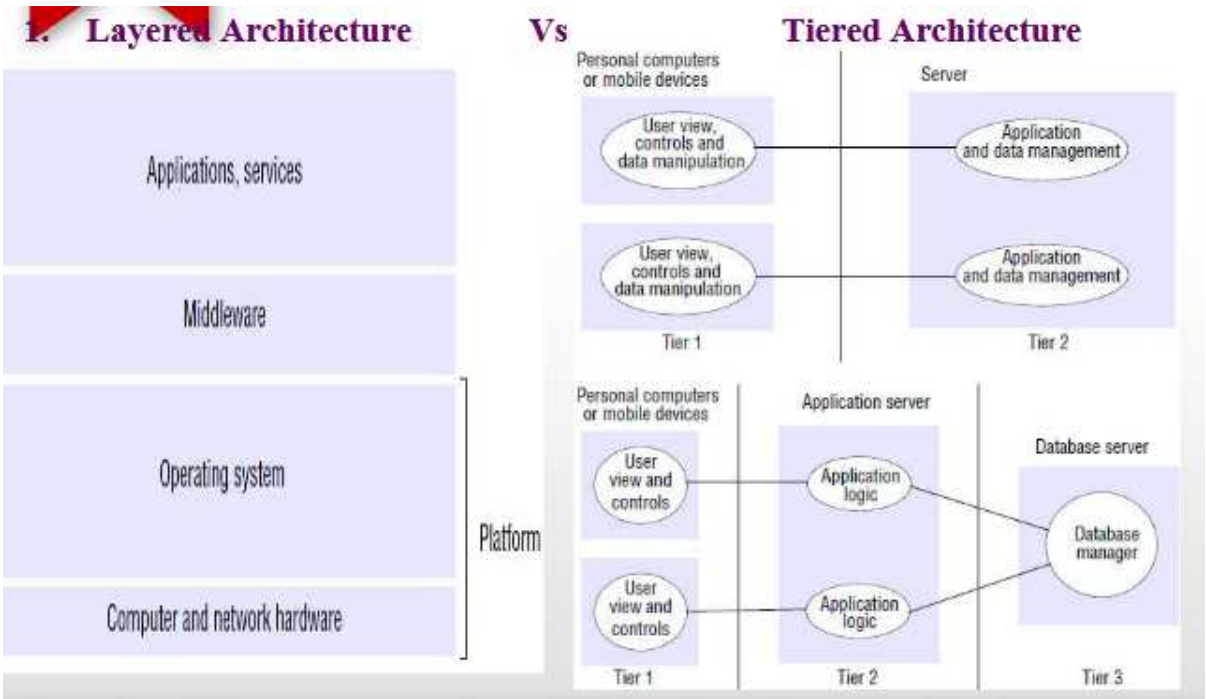
Distributed systems:	*Early*	*Internet-scale*	*Contemporary*
Scale	Small	Large	Ultra-large
Heterogeneity	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
Openness	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
Quality of service	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

(ii) Explain about the performance evaluation parameters for distributed mutual exclusion. (4) **(K2, CO4)**

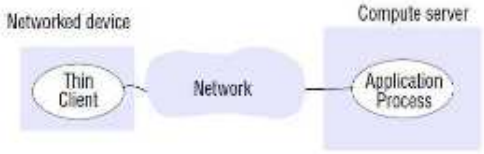
Ans: Message Complexity, Response Time, Synchronization delay, System Throughput, Low and high load performance.

OR

9. Explain about Architectural patterns in distributed system**.** (8) **(K2, CO2)**



2. Architectural Patterns



Java Script: It allows UI and application logic to be programmed and executed in a browser window

AJAX

Enable major interactive web applications to be developed and deployed. It allows current page to be displayed with new values of application.

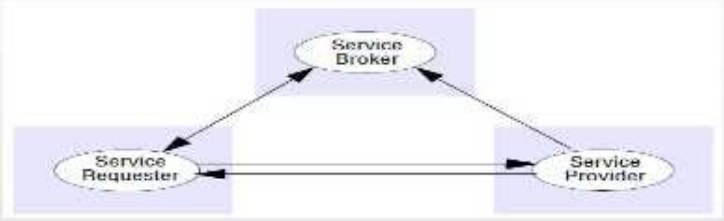
Thin Clients

It depends heavily on another computer (server) to fulfill its computational roles

Proxy

Designed to support location transparency in Remote procedure calls.

Web Services



Reflections:

Introspection – Dynamic discovery of properties of system

Intercession - Ability to dynamically modify structure or behavior.

PART – C **(16 Marks)**

10. Explain Lamport's Algorithm for Non-Token based Mutual Exclusion and discuss about optimization method for Lamport's Distributed Mutex Algorithm. (16) **(K2, CO4)**

Lamport's DMX Algorithm

Requesting the critical section:

- When a site S_i wants to enter the CS, it broadcasts a REQUEST(ts_i, i) message to all other sites and places the request on $request\_queue_i$. ((ts_i, i) denotes the timestamp of the request.)
- When a site S_j receives the REQUEST(ts_i, i) message from site S_i,places site S_i's request on $request\_queue_j$ and it returns a timestamped REPLY message to S_i.

Executing the critical section: Site S_i enters the CS when the following two conditions hold:

L1: S_i has received a message with timestamp larger than (ts_i, i) from all other sites.

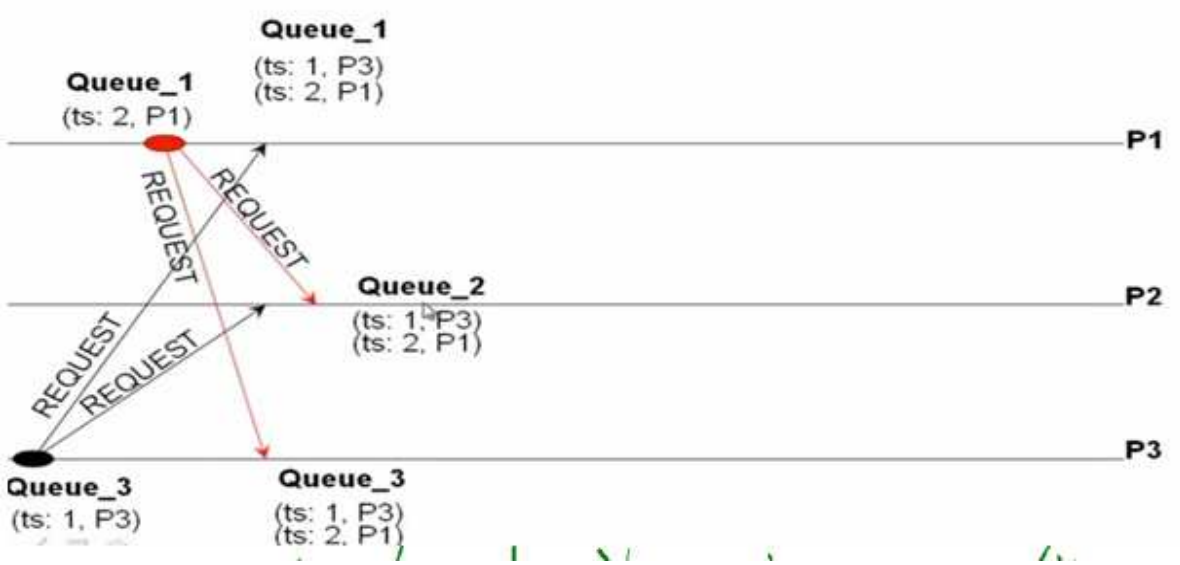
L2: S_i's request is at the top of $request\_queue_i$.

Releasing the critical section:

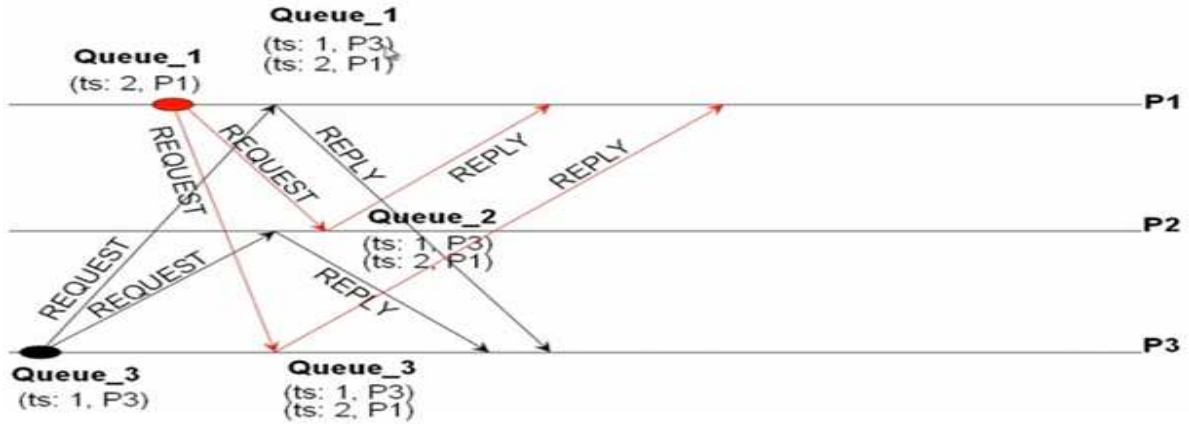
- Site S_i, upon exiting the CS, removes its request from the top of its request queue and broadcasts a timestamped RELEASE message to all other sites.
- When a site S_j receives a RELEASE message from site S_i, it removes S_i's request from its request queue.

When a site removes a request from its request queue, its own request may come at the top of the queue, enabling it to enter the CS.

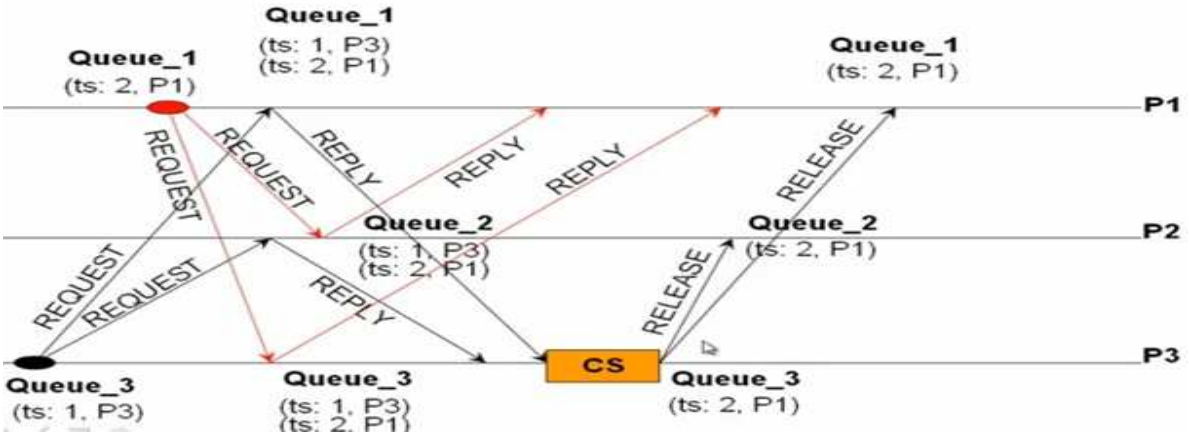
Example: Step 1



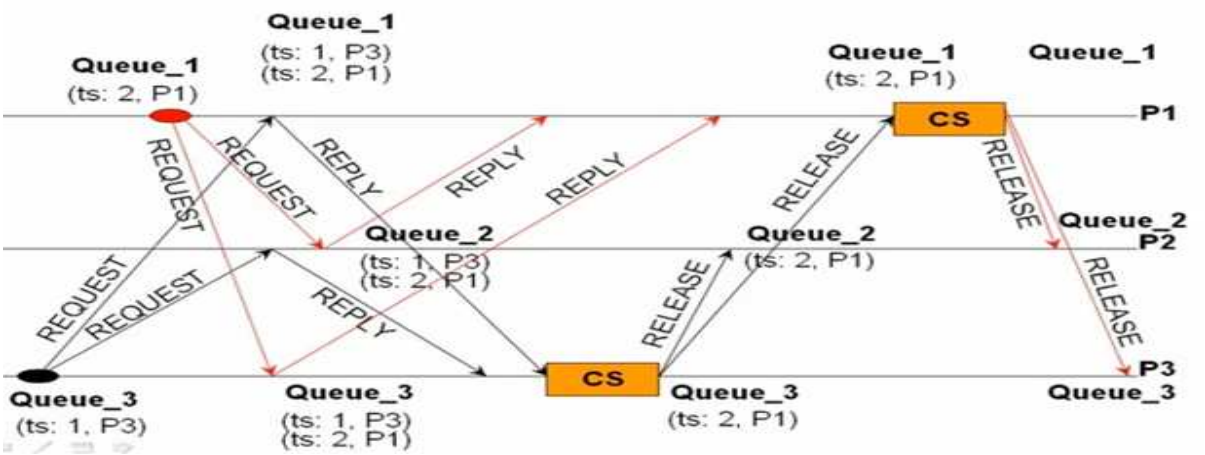
Step 2



Step 3



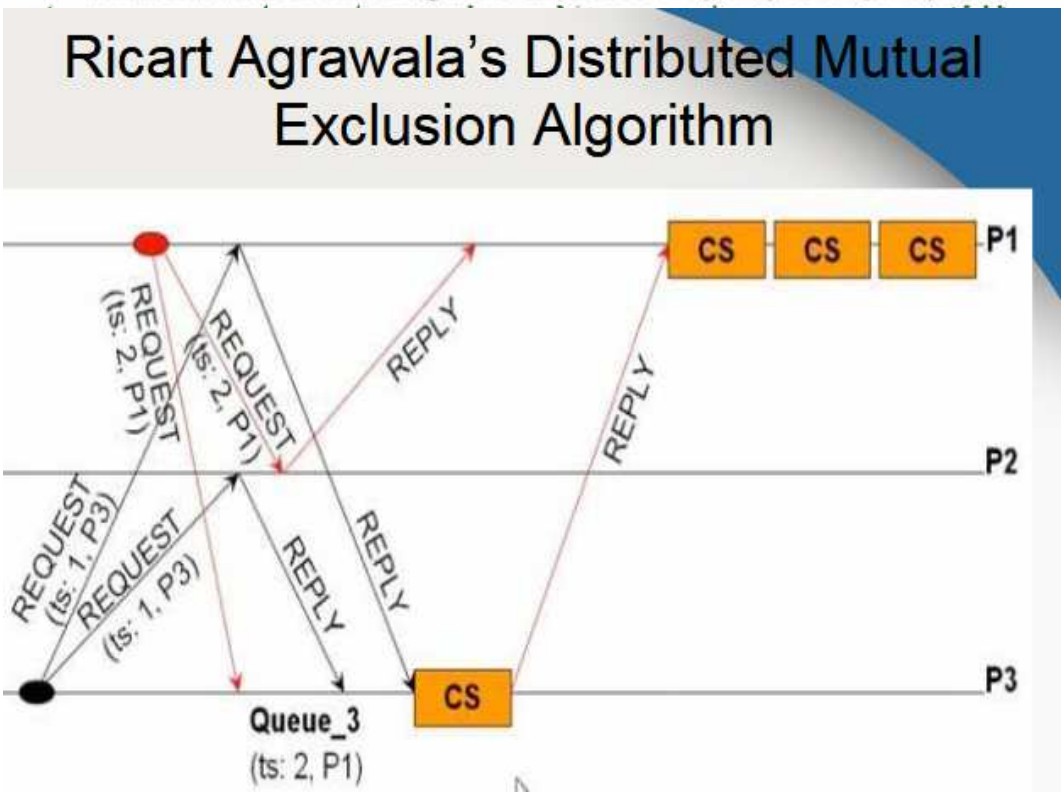
Step 4



Optimization of Lamport's DMutex Algorithm.

- **# Messages:** 3 (N-1)
  - N-1 REQUESTS; N-1 REPLY; N-1 RELEASE
- **Synchronization Delay:** T
- **System Throughput:** 1/(T+E)

- **Optimization:** Process Pi need not send a REPLY message to process Pj if the timestamp of the REQUEST message of Pj is higher than that of Pi. Process Pi can simply send a RELEASE message when done with its CS execution.
  - So, a process has to basically wait for a REPLY or RELEASE message from every other process and its REQUEST should be in the front of the queue to enter the CS.
  - With this, the # messages is between 2*(N-1) to 3*(N-1).

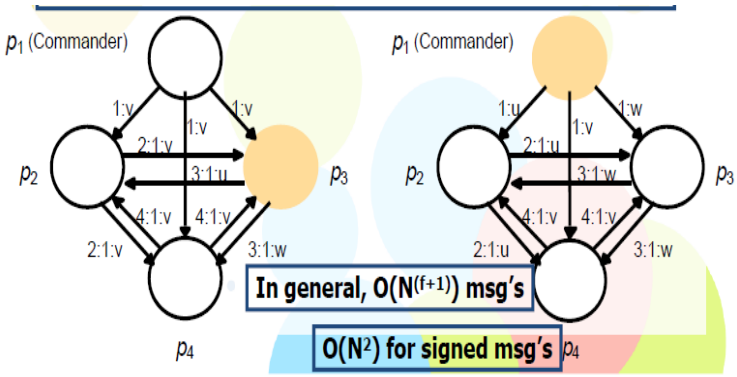


OR

11. Consider Byzantine Consensus for the following scenario (4+4+4+4)
(K3, CO4)

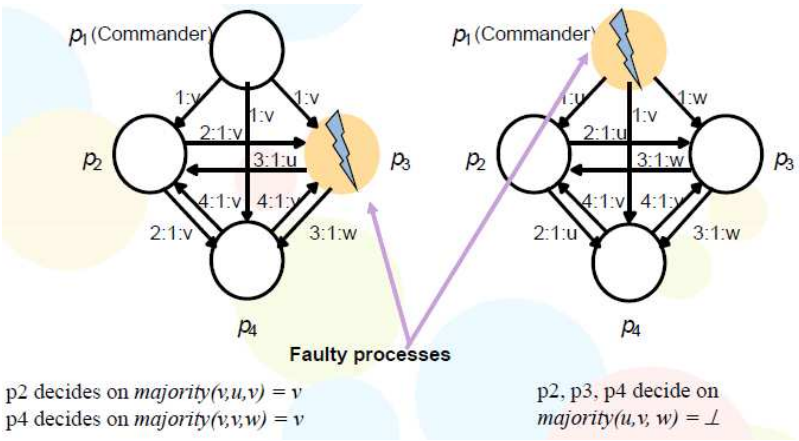
a. How many no of faulty processors can be present for a distributed system with 'N' number of nodes to achieve agreement? Simulate the case.

Ans: Condition N > 3 * f, where N is number of processes, f is number of faulty process. Always the number of faulty processes should be 1/3 of total number of processes.

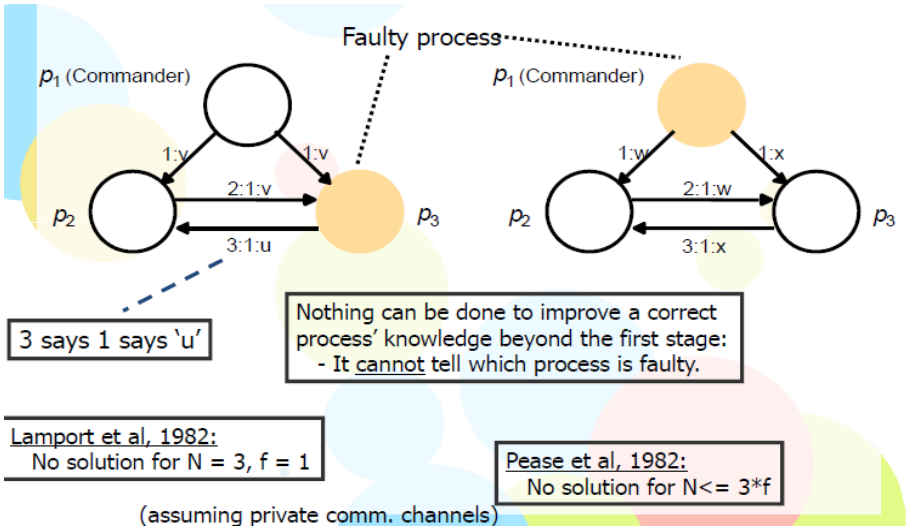


b. Simulate the case where source is a faulty processor.

Ans: When the source is faulty, it is difficult to achieve agreement in Byzantine consensus.



c. Simulate the case where byzantine consensus cannot be arrived for 3 processors where one being fault.



d. What would happen if it is applied on asynchronous systems.

Ans: In Asynchronous Distributed systems, it is difficult to reach consensus due to absence of failure detection mechanisms and there is no time limit on any message sending or receiving.

But, we can reach to consensus only if we wait for unbounded time period may in years or more.