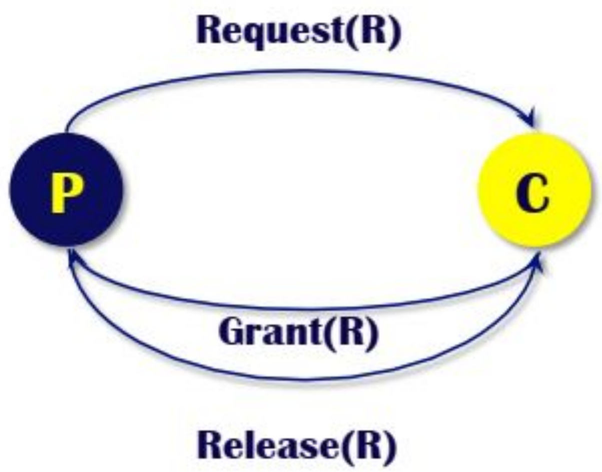
Mutual Exclusion in Distributed Systems

Centralized Algorithm

Centralized Algorithm – Free Resource

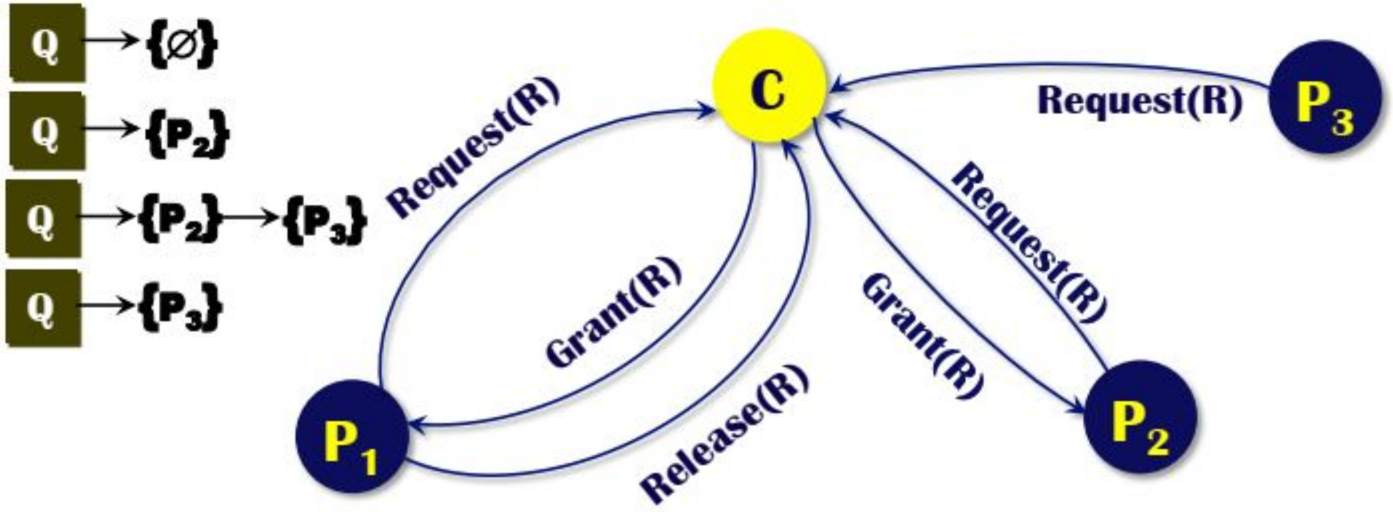
- Centralized Algorithm achieves DME by closely "**mimicking**" ME in single processor systems
  - One process, **C**, is the **Coordinator** – Coordinates access to resources
  - Other processes issue requests to access resource

1. **Request Resource**
2. ***Wait for Response***
3. **Receive Grant**
4. ***Access Resource***
5. **Release Resource**



Centralized Algorithm – Allocated Resource

- If resource is currently by another process, **C** does not reply until resource is released
  - Maintain queue of pending requests, serviced in FIFO order



Centralized Algorithm

- Advantages
  - Easy to understand, verify and implement
  - Access Fairness – FIFO order of service is simple to implement
    - Note FIFO policy does not guarantee "Fair Share" access
- Limitations
  - Not scalable to large scale distributed systems – Coordinator is likely to become a **bottleneck**
  - A process requesting a resource **cannot distinguish** between being in a "**blocked state**", waiting for the resource to be released, from **waiting** for a **failed** coordinator
    - Timeout mechanisms or alive messages may be required – to be designed carefully to avoid early and late timeouts or unnecessary alive messages

PERMISSION BASED DME

DISTRIBUTED ALGORITHM

Distributed Algorithm for ME

Ricart and Aggarwal Algorithm

A Distributed ME Algorithm

- ***Ricart and Agrawala Algorithm*** assumes there is a mechanism for "**totally ordering of all events**" in the system and a **reliable** message system
  - ***Lamport's algorithm*** can be used for total ordering
- A process wanting to enter it CS sends a message with (CS name, process id, current time) to all processes, including itself
- When a process receives a CS request from another process, it reacts based on its current state with respect to the CS requested.
  - **Three** possible cases must be considered

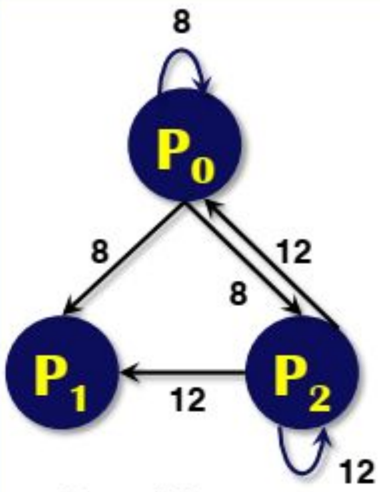
Ricart AND Agrawala Algorithm – Basic Cases

- RA Algorithm distinguishes between 3 cases
  a. If the receiver is not in the CS **and** it does not want to enter the CS, it sends an **OK** message to the sender.
  b. If the receiver is in the CS, it does not reply and queues the request.
  c. If the receiver wants to enter the CS but has not yet, it compares the ***timestamp of the incoming message*** with the ***timestamp of its message*** sent to everyone – ***The lowest timestamp wins.***
     - If the incoming timestamp is **lower**, the receiver sends an OK message to the sender.
     - If its own timestamp is **lower**, the receiver queues the request and sends nothing.

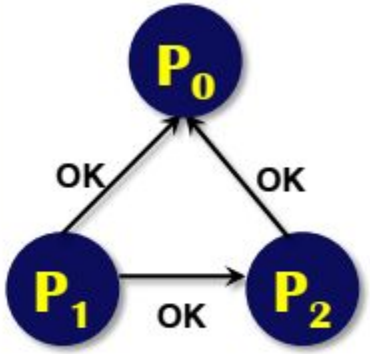
RA DME Algorithm – CS Entry and Exit

- After a process sends out a request to enter a CS, it waits for an OK from all the other processes.
  - Upon receiving all OK message, process enters the CS
- Upon exiting CS, it sends OK messages to all processes on its queue, which are waiting for that CS
  - Upon sending OK messages, processes are deleted from the queue.

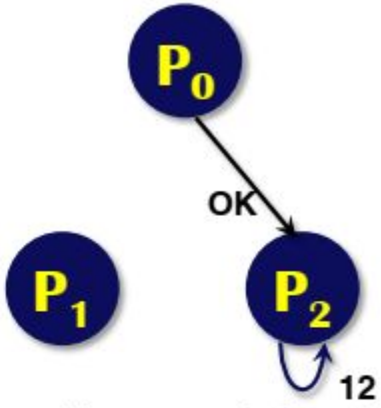
RA Algorithm – Example



- P_0 and P_2 request access to the same resource, nearly at the same time
  - Both send requests with timestamps, 8 and 12, respectively



- P_0 has the lowest timestamp, 8
  - P_0 wins access to the resource



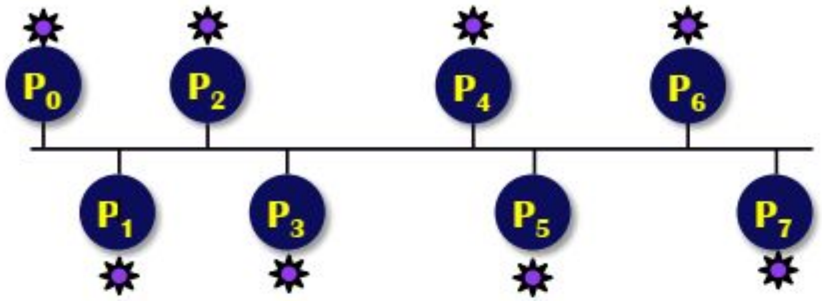
- Upon completion of its CS, P_0 sends OK message to P_2
  - P_2 can enter its own CS

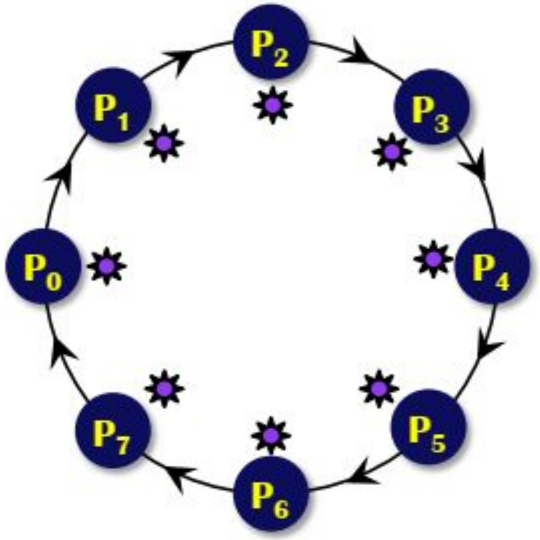
Distributed Permissions Analysis

- Benefits
  - No central bottleneck
    - This results in improved performance
  - Fewer messages than the decentralized algorithm
- Limitations
  - The system is exposed to n points of failure
    - If a node fails to respond, the entire system locks up

Token Ring Algorithm for ME

A Token Ring Algorithm





- An unordered group of processes on a network.
- A logical ring constructed in software
- Token circulates at high speed on the network
- A process must have a token to enter its CS