

3140702
Operating System

Unit – 5

Deadlock



Subject Faculty: Prof. Suhag Baldaniya

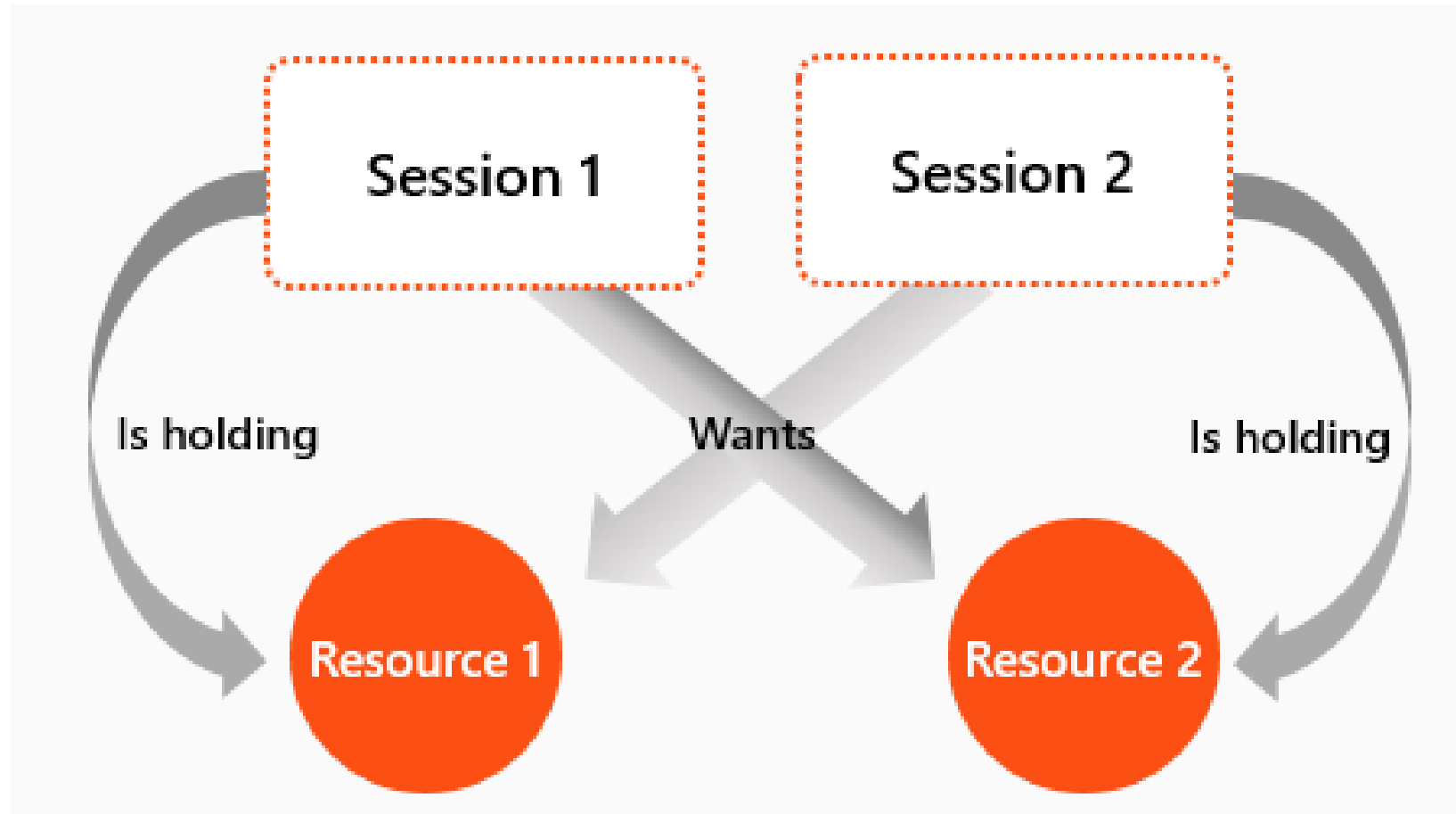
Disclaimer

- *It is hereby declared that the production of the said content is meant for non-commercial, scholastic and research purposes only.*
- *We admit that some of the content or the images provided in this channel's videos may be obtained through the routine Google image searches and few of them may be under copyright protection. Such usage is completely inadvertent.*
- *It is quite possible that we overlooked to give full scholarly credit to the Copyright Owners. We believe that the non-commercial, only-for-educational use of the material may allow the video in question fall under fair use of such content. However we honor the copyright holder's rights and the video shall be deleted from our channel in case of any such claim received by us or reported to us.*

Topics to be covered

- Deadlock and its Principles
- Four Necessary Conditions of Deadlock
- Deadlock Prevention
- Resource Allocation Graph (RAG)
- Deadlock Avoidance - Banker's algorithm
- Deadlock Detection & Recovery.

What is Deadlock?



Definition of Deadlock

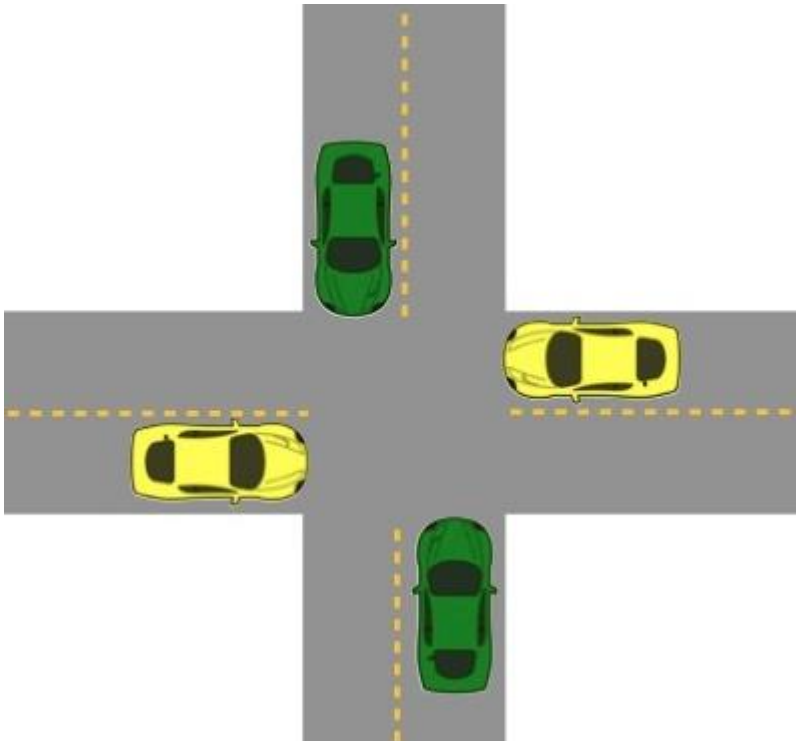
- A deadlock consists of a **set** of blocked processes, each **holding** a resource and **waiting** to acquire a resource held by another process in the set.

For Example:

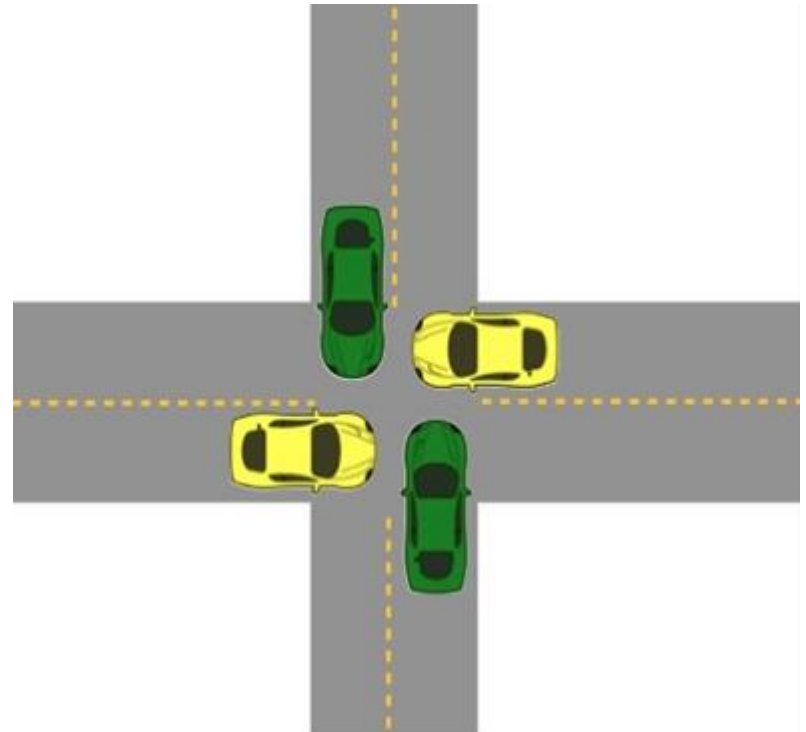
- two processes each want to record a scanned document on a CD.
- process 1 requests for scanner & gets it
- process 2 requests for CD writer & gets it
- process 1 requests CD writer but is blocked
- process 2 requests scanner but is blocked.
- At this point both processes are blocked and will remain so forever,
- **This situation is called a deadlock**

Deadlock

- The processes are the cars.
- The resources are the spaces occupied by the cars



Deadlock possible



Deadlock occurred

Sequence of events to use a Resource

```
graph TD; A[Request the resource] --> B[Allocate the resource]; B --> C[Release the resource];
```

Request the resource

Allocate the resource

Release the resource

Necessary Conditions for Deadlock

1. Mutual exclusion:

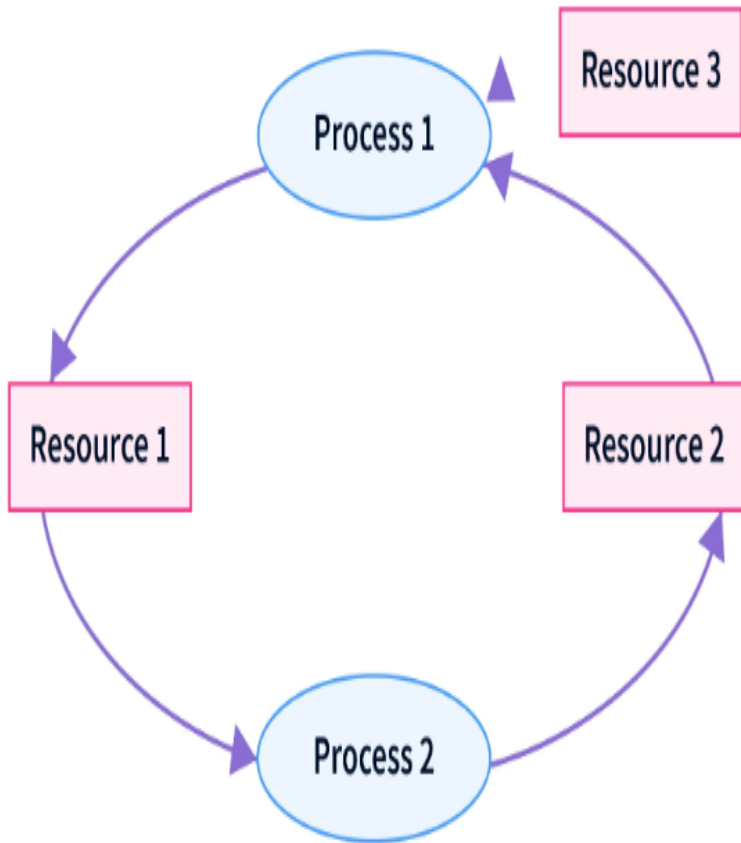
- **At least one resource type** in the system which can be used in **non-shareable mode** (i.e. one at a time). for example printer.
- Because many resource can be shared by more than one process at a time .(e.g., Memory location).

2. Hold and Wait:

- Processes are allowed to request for new resources without releasing the resources they are currently holding.
- Hold and wait is a condition in which a process is holding one resource while simultaneously waiting for another resource that is being held by another process.

Necessary Conditions for Deadlock

- The process cannot continue till it gets all the required resources.



- Resource 1 is allocated to Process 2
- Resource 2 is allocated to Process 1
- Resource 3 is allocated to Process 1
- Process 1 is waiting for Resource 1 and holding Resource 2 and Resource 3
- Process 2 is waiting for Resource 2 and holding Resource 1

Necessary Conditions for Deadlock

3. No pre-emption:

- A resource can be released only voluntarily by the process holding it after that process has completed its task.
- Preemption is temporarily interrupting an executing task and later resuming it.
- **For example,** if a process P1 is using a resource and a high priority process P2 requests for the resource, process P1 is stopped and the resources are allocated to P2.

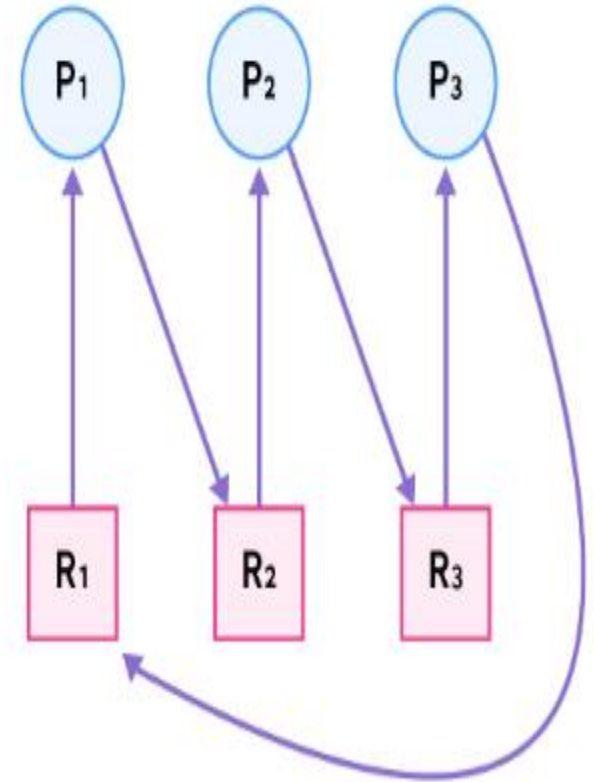
Necessary Conditions for Deadlock

4. Circular wait:

- Two or more processes must form a circular chain in which each process is waiting for a resource that is held by the next member of the chain.
- In circular wait, two or more processes wait for resources in a circular order.

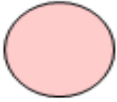
Necessary Conditions for Deadlock


- To eliminate circular wait, we assign a priority to each resource.
- A process can only request resources in increasing order of priority.
- **In the example above**, process **P3** is requesting resource **R1**, which has a number lower than resource **R3** which is already allocated to process **P3**.
- So this request is invalid and cannot be made, as **R1** is already allocated to process **P1**.

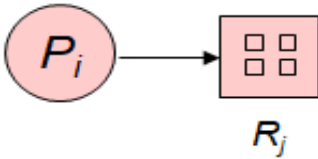


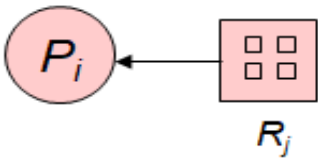
Deadlock Modeling

- For Deadlock modeling a directed graph, called **Resource Allocation Graph (RAG)** is used.
- Notation used for representation of RAG -

- Process 

- Resource Type with 4 instances 

- P_i requests instance of R_j 

- P_i is holding an instance of R_j 

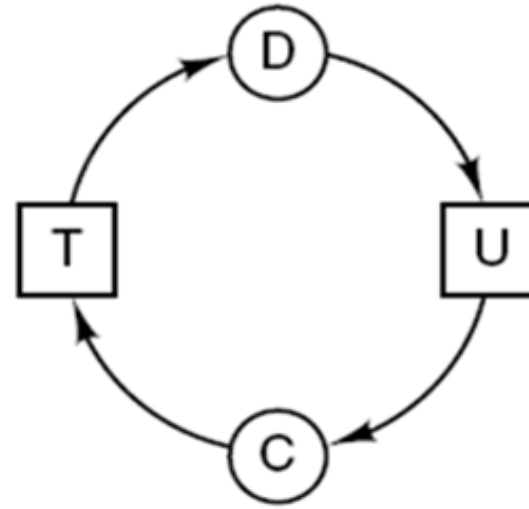
Construction of Resource Allocation Graph



(a)



(b)



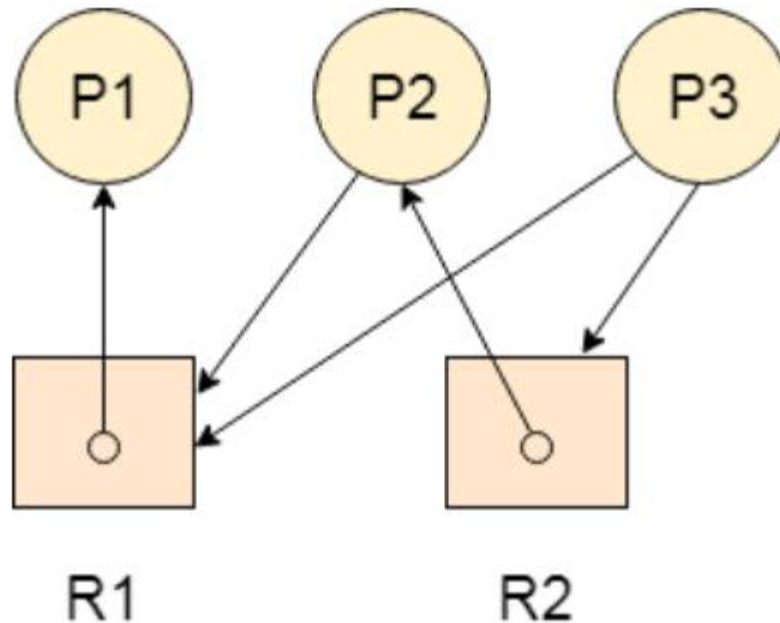
(c)

Figure : Resource allocation graphs

(a) Holding a resource. (b) Requesting a resource. (c) Deadlock.

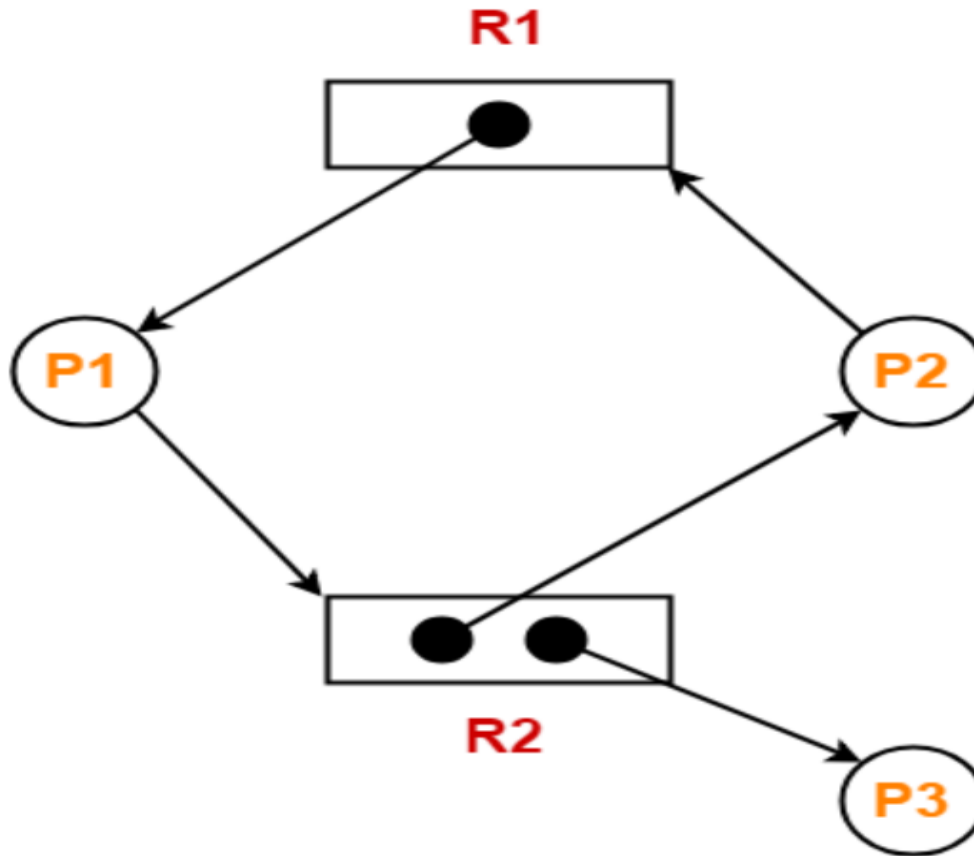
Resource Allocation Graph

Problem-1 : Find the system is in a deadlock state or not?



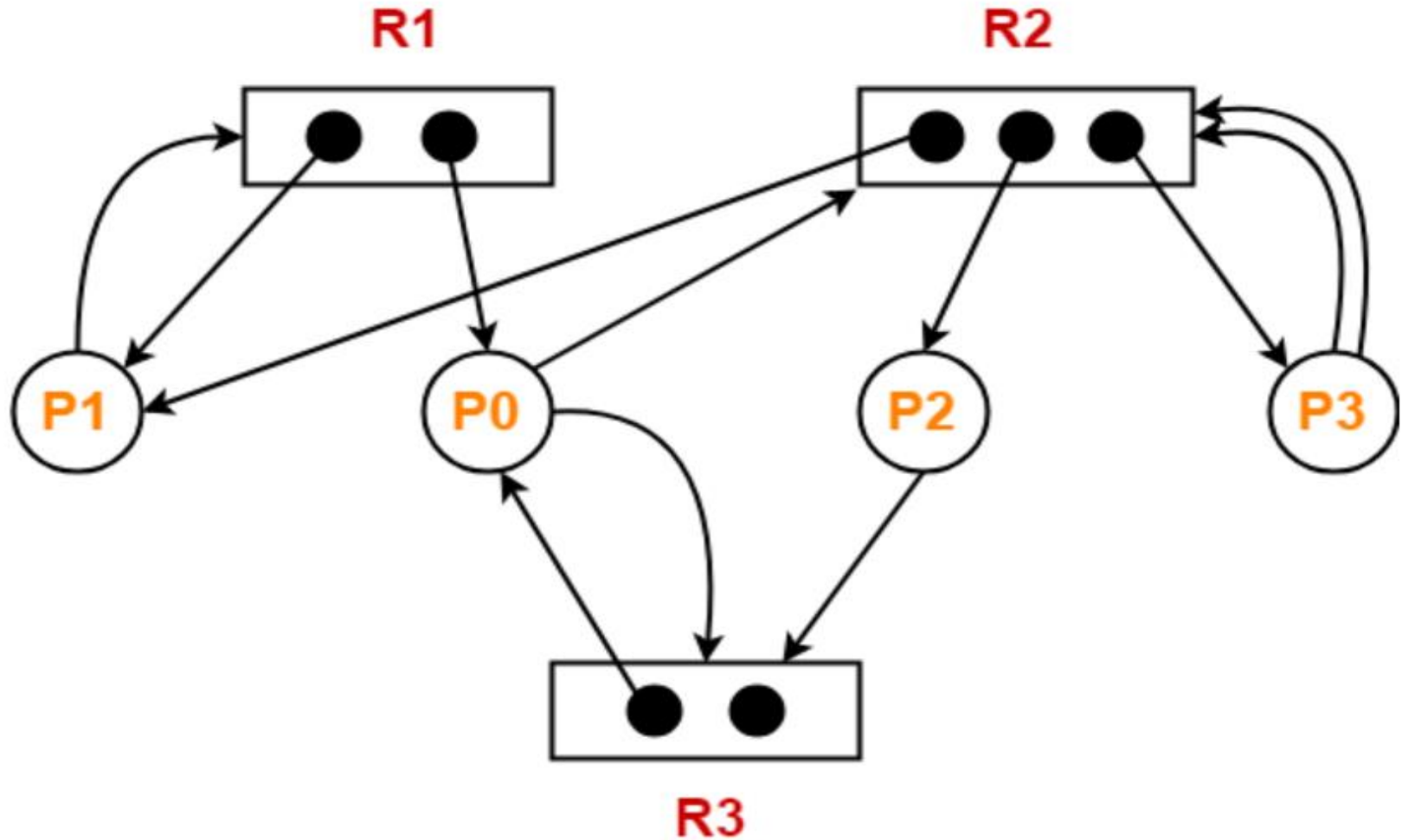
Resource Allocation Graph

Problem-2 : Find the system is in a deadlock state or not?



Resource Allocation Graph

Problem-3 : Find the system is in a deadlock state or not?



Handling Deadlocks In DOS

- Handling of deadlocks in distributed systems is more complex than in centralized systems.
- Because the resources, the processes, and other relevant information are scattered on different nodes of the system.
- **Strategies to handle deadlocks:**
 1. **Prevention** – constraints are imposed on the way in which processes request resources in order to prevent deadlocks.
 2. **Avoidance** – resources are carefully allocated to avoid deadlocks.
 3. **Detection and recovery** – deadlocks are allowed to occur and a detection algorithm is used to detect them. After deadlock is detected it is resolved by certain means.
 4. **Ignore** – do nothing, just ignore the problem.

Deadlock Prevention

- Deadlock can be prevented by violating the one of the four conditions that leads to deadlock.
 - Attacking the Mutual Exclusion Condition
 - Attacking the Hold and Wait Condition
 - Attacking the No Preemption Condition
 - Attacking the Circular Wait Condition

Violation of Mutual Exclusion Condition

- No deadlock if each resource can be assigned to more than one process.
- This can violate the hardware properties of a resource.
- We can not assign some resources to more than one process at a time **such as printer**.
- Not feasible

Violation of Hold and Wait Condition

1. **Conservative Approach** : Process is allowed to start execution if and only if it has acquired all the resources (less efficient, not implementable, easy, deadlock independence)
 - A process is **allowed to run if all resources it needed is available**. Otherwise nothing will be allocated and it will just wait.
 - Problem with this strategy is that a process may not know required resources at start of run.
 - Resource will not be used optimally.
 - It may cause starvation of a process that needs many resources.

Violation of Hold and Wait Condition

2. **Do not Hold:** A process will acquire all desired resources but before making any fresh request it must release all the resources that it currently hold (efficient, implementable)
3. **Wait Timeout:** We place a maximum time up to which a process can wait after which process and release all the holding resources.

Violation of No Preemption Condition

Forcefully Pre-emption: We allow a process to forcefully pre-empt the resource by other processes

- For example - When a process request for a resource that is currently not available, all the resources held by the process are taken away from it and process is blocked.
- This method may be used by **high priority process or system process**.
- The process which are in **waiting state** must be **selected as a victim** instead of process in the **running state**.

Violation of Circular Wait Condition

- To provide a **global numbering of all the resources**.
- Processes can request resources whenever they want to, but all requests must be made in increasing or decreasing order.
- A process need not acquire them all at once.
- Circular wait is prevented if a process holding resource n cannot wait for resource m , if $m > n$.

1.Printer, 2.Scanner, 3.Plotter, 4.Tape drive, 5.CD ROM

- A process may request 1st a CD ROM drive, then tape drive. But it may not request 1st a plotter, then a Tape drive.
 - Resource graph can never have cycle.
-
- P1 – R1, R2, R4, R43, R56
 - P1 – R1, R4, R2 NOT POSSIBLE

Deadlock Avoidance

- In most systems, however, resources are requested one at a time. The system must be able to decide whether granting a resource is **safe or not and only make the allocation when it is safe.**

Safe and Unsafe state

- If the system can allocate resources to the process in such a way that it can avoid deadlock. Then the system is in a **safe state**.
- If the system can't allocate resources to the process safely, then the system is in an **unsafe state**.

Safe State

- A total of 10 instance of the resource exist, so with 7 resources already allocated, there are 3 still free.
- *Figure : Demonstration that the state in (a) is safe.*

Has Max		
A	3	9
B	2	4
C	2	7

Free: 3
(a)

Has Max		
A	3	9
B	4	4
C	2	7

Free: 1
(b)

Has Max		
A	3	9
B	0	—
C	2	7

Free: 5
(c)

Has Max		
A	3	9
B	0	—
C	7	7

Free: 0
(d)

Has Max		
A	3	9
B	0	—
C	0	—

Free: 7
(e)

Unsafe State

- Figure : Demonstration that the state in (b) is not safe.

Has Max		
A	3	9
B	2	4
C	2	7

Free: 3
(a)

Has Max		
A	4	9
B	2	4
C	2	7

Free: 2
(b)

Has Max		
A	4	9
B	4	4
C	2	7

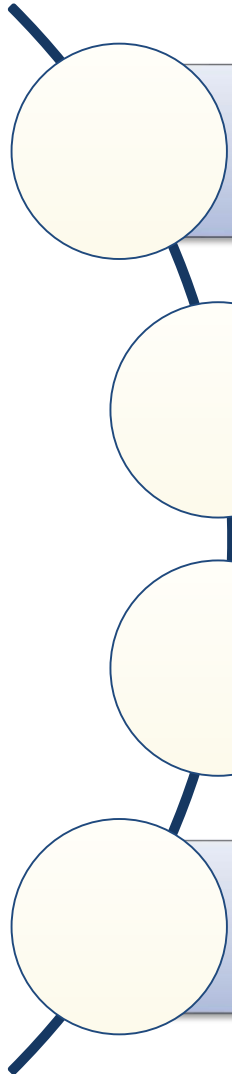
Free: 0
(c)

Has Max		
A	4	9
B	—	—
C	2	7

Free: 4
(d)

- Thus from the safe state system can guarantee that all process will finish and from unsafe state no such guarantee can be given.

The Banker's Algorithm for a Single Resource



It is modeled on the way a small-town banker might deal with a group of customers to whom he has granted lines of credit.

What the algorithm does is check to see if granting the request leads to an unsafe state.

If it does, the request is denied.

If granting the request leads to a safe state, it is carried out.

Numerical - The Banker's Algorithm

- Considering a system with five processes P₀ through P₄ and three resources of type A, B, C. Resource **type A has 10 instances, B has 5 instances and type C has 7 instances**. Find the safe sequence of the system with current allocation.

Process	Allocation	Max	Available
	A B C	A B C	A B C
P ₀	0 1 0	7 5 3	3 3 2
P ₁	2 0 0	3 2 2	
P ₂	3 0 2	9 0 2	
P ₃	2 1 1	2 2 2	
P ₄	0 0 2	4 3 3	

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Numerical - The Banker's Algorithm

- A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST? **(GATE-2007 | 2-MARKS)**

	alloc				request		
	X	Y	Z		X	Y	Z
P0	1	2	1		1	0	3
P1	2	0	1		0	1	2
P2	2	2	1		1	2	0

(A) P0

(B) P1

(C) P2

(D) None of the above, since the system is in a deadlock

Numerical - The Banker's Algorithm

Suppose there are four processes in execution with 12 instances of a Resource R in a system. The maximum need of each process and current allocation are given below:

Process	Max. need	Current Allocation
P1	8	3
P2	9	4
P3	5	2
P4	3	1

With reference to current allocation, is system safe ? If so, what is the safe sequence ? (UGC-NET 2016 AUG)

A

No

B

Yes, $P_1 P_2 P_3 P_4$

C

Yes, $P_4 P_3 P_1 P_2$

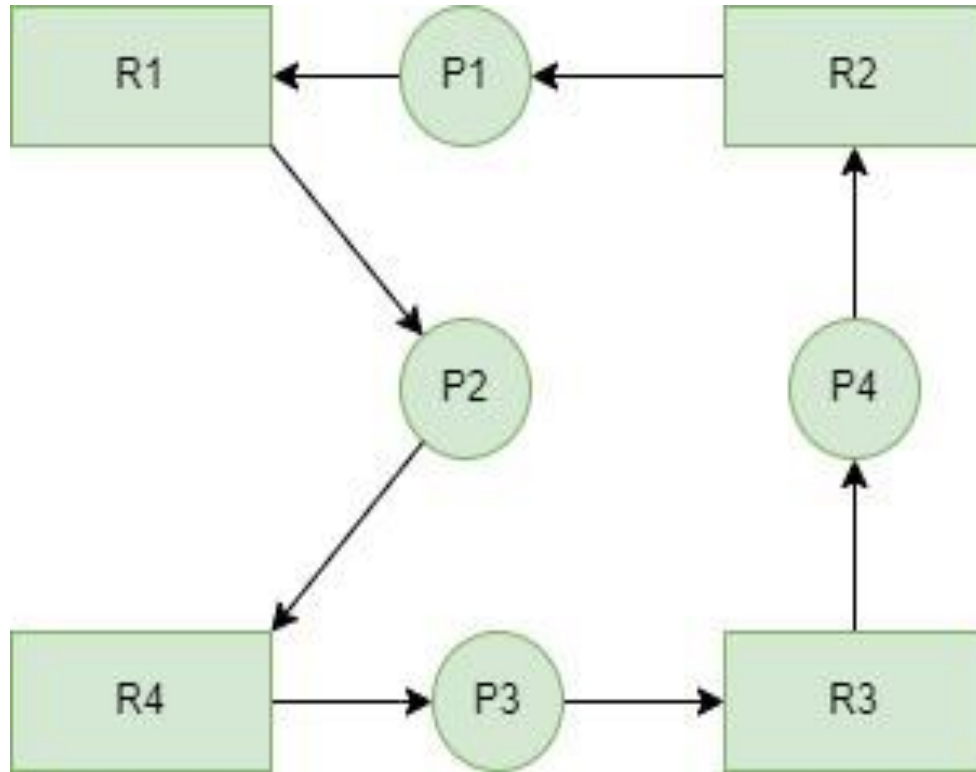
D

Yes, $P_2 P_1 P_3 P_4$

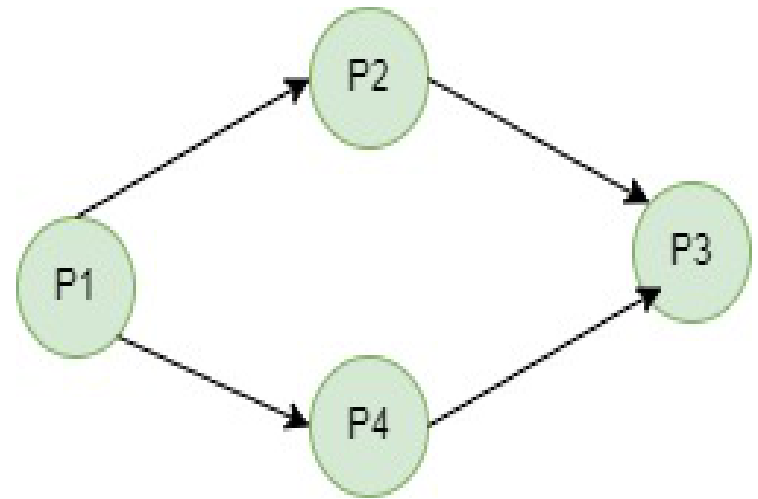
Deadlock Detection

- Deadlock detection and recovery is the process of detecting and resolving deadlocks in an operating system.
- When deadlock occurs ?
- Difference between prevention and detection ?
- Instead, it lets them occur, tries to detect when this happens, and then takes some action to recover after the fact.
- For deadlock detection, the system must provide
 - An algorithm that examines the state of the system to **detect** whether a deadlock has occurred
 - And an algorithm to **recover** from the deadlock

Deadlock Detection-Single Instance



Resource Allocation Graph



Wait for Graph

Deadlock Detection: Multiple Resources of Each Type

- Three process and four classes for device are there

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix}$$

Tape drives
Plotters
Scanners
CD Roms

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

Tape drives
Plotters
Scanners
CD Roms

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Deadlock Detection: Multiple Resources of Each Type

- We will check the request matrix, process 0 request can not satisfied.
- The process 1 request can not be satisfied.
- Process 2 request can be satisfied so resources are granted.
- Now $A=(0\ 0\ 0\ 0)$
- After completion of process 2
- $A=(2\ 2\ 2\ 0)$
- Now we can assign resources to process 1.
- After process 1 $A=(4\ 2\ 2\ 1)$
- So there no dead lock.

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix}$$

Tape drives
Plotters
Scanners
CD Roms

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

Tape drives
Plotters
Scanners
CD Roms

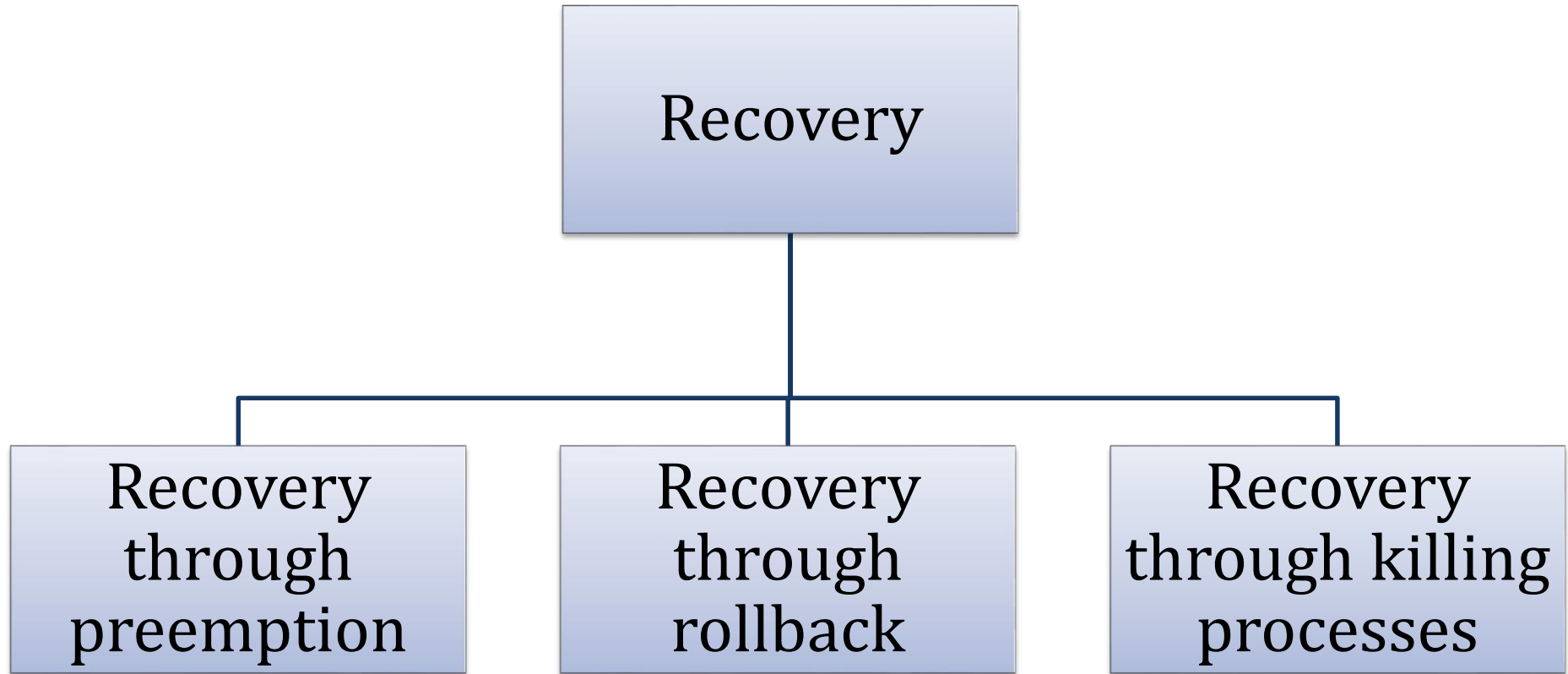
Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

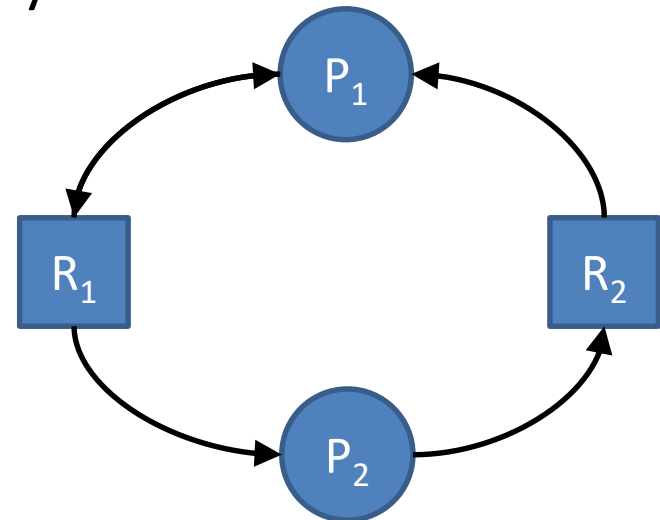
Deadlock Recovery



Deadlock Recovery

■ Recovery through preemption

- In some cases, it may be possible to temporarily take a resource away from its current owner and give it to another process.
- Recovering this way is frequently difficult or impossible.
- Choosing the process to suspend depends largely on which ones have resources that can easily be taken back.



Deadlock Recovery

■ Recovery through killing processes

- The simplest way to break a deadlock is to kill one or more processes.
- Kill all the process involved in deadlock.
- Kill process one by one.
- After killing each process check for deadlock.
 - If deadlock recovered then stop killing more process.
 - Otherwise kill another process.

Deadlock Recovery

- **Recovery through rollback**

- Checkpoint a process periodically.
- Check pointing a process means that its state is written to a file so that it can be restarted later.
- When deadlock is detected, rollback the preempted process up to the previous safe state before it acquired that resource.

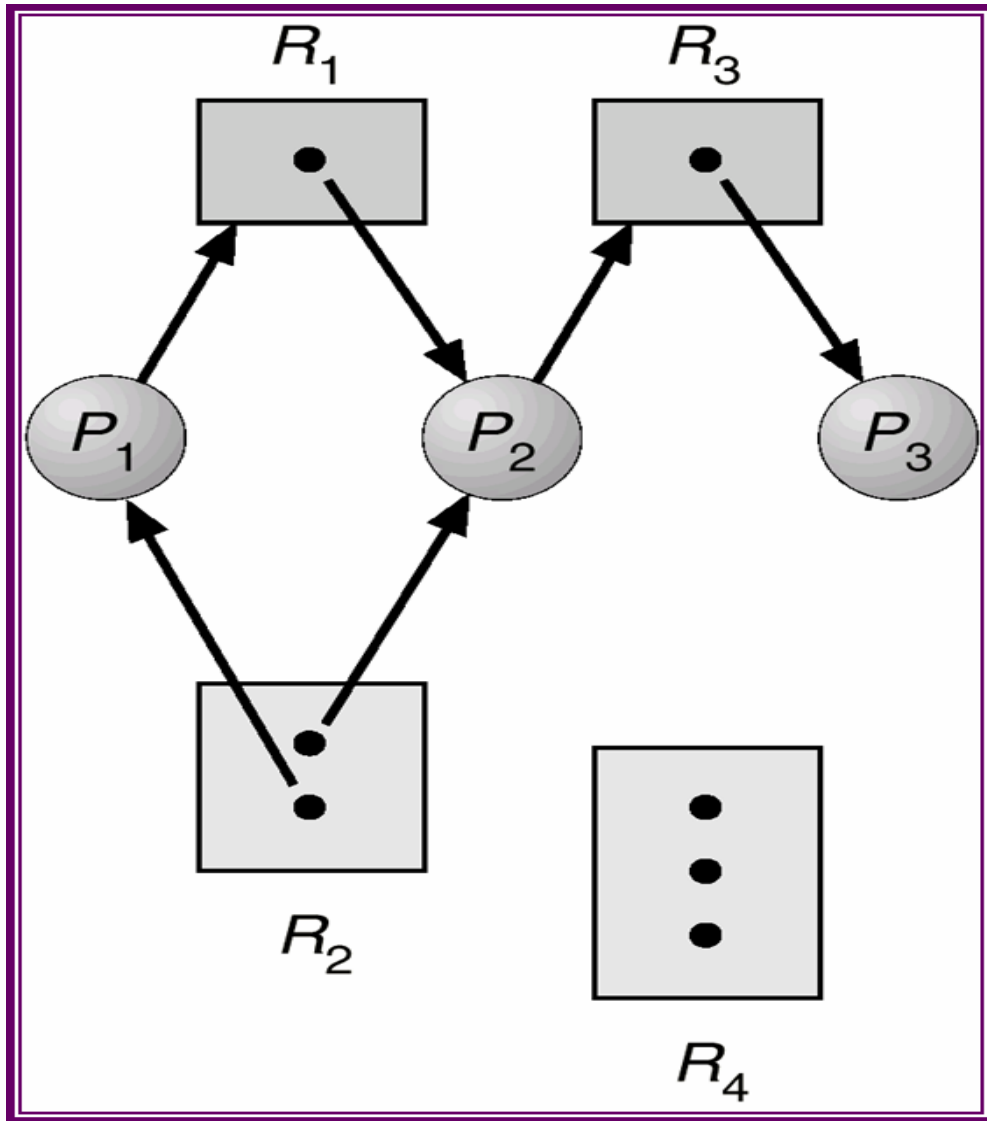
Advantages- Deadlock Detection

- **Improved System Stability:** Detecting and resolving deadlocks can help to improve the stability of the system.
- **Better Resource Utilization:** By detecting and resolving deadlocks, the operating system can ensure that resources are efficiently utilized and that the system remains responsive to user requests.
- **Better System Design:** Deadlock detection and recovery algorithms can provide insight into the behavior of the system and the relationships between processes and resources, helping to inform and improve the design of the system.

Disadvantages- Deadlock Detection

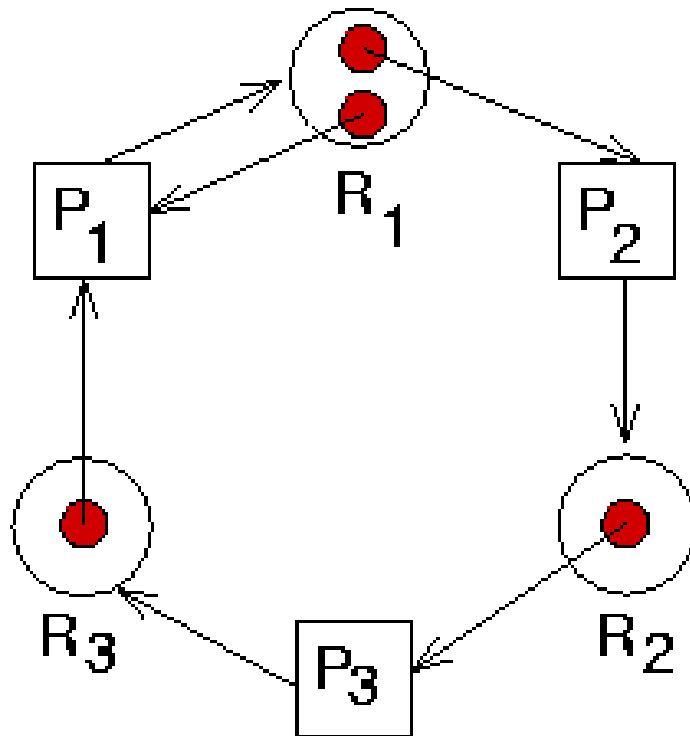
- **Performance Overhead:** The system must regularly check for deadlocks and take appropriate action to resolve them.
- **Complexity:** Deadlock detection and recovery algorithms can be complex to implement, especially if they use advanced techniques such as the Resource Allocation Graph.
- **Risk of Data Loss:** In some cases, recovery algorithms may require rolling back the state of one or more processes, leading to data loss or corruption.

Practice Problem



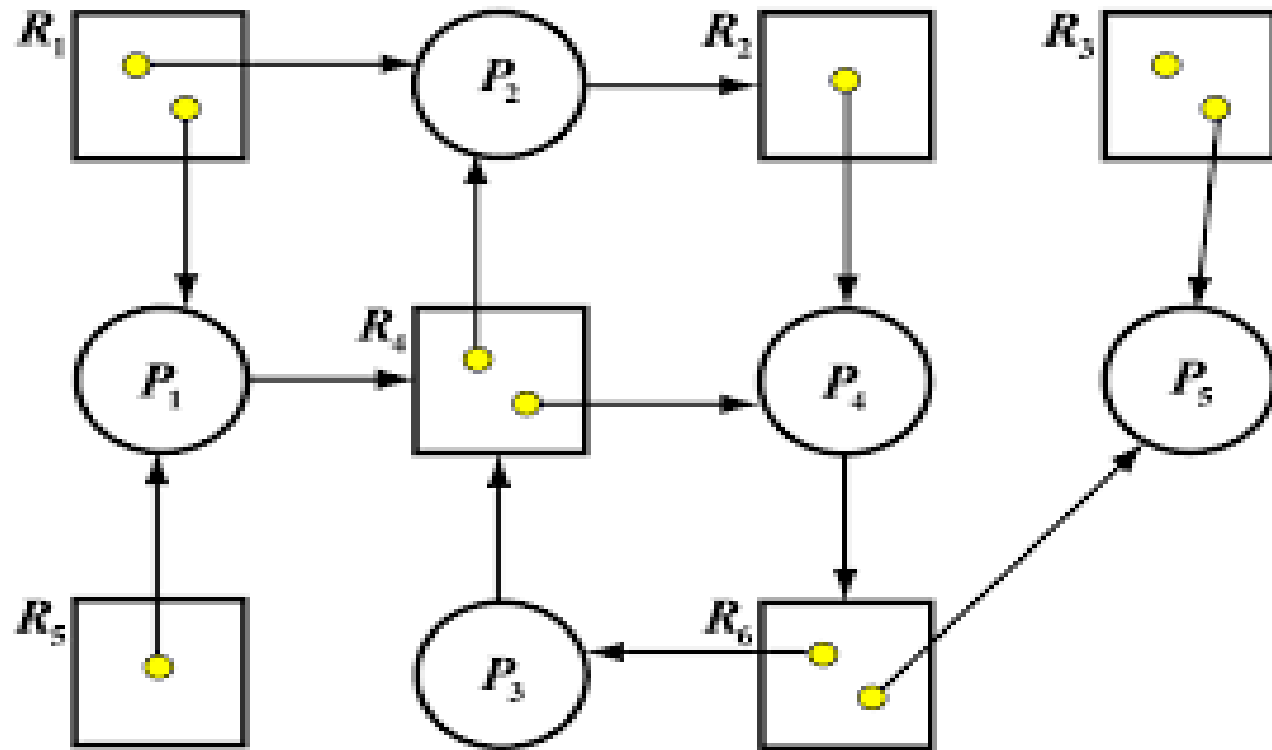
Detect Deadlock using RAG
and draw wait for graph.

Practice Problem



Detect Deadlock using RAG
and draw wait for graph.

Practice Problem



Detect Deadlock using RAG
and draw wait for graph.

