CS & IT ENGINEERING

Operating System

Deadlock

Lecture - 04

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Deadlock Avoidance

Topic Banker's Safety Algorithm

Topic Banker's Resource Request Algorithm

Topics to be Covered

Topic Banker's Resource Request Algorithm

Topic Deadlock Detection

Topic Recovery from Deadlock

Topic : Banker's Algorithm

Process	Allocation	Max	Available	Need
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	~~3 3 2~~ 3 2 2	7 4 3
P_1	2 0 0	3 2 2	P1⇒ 5 2 2 P3⇒ 7 4 3	1 2 2
P_2	3 0 2	9 0 2	⋮	6 0 0
P_3	~~2 1 1~~ 2 2 1	2 2 2	safe	~~0 1 1~~ 0 0 1
P_4	0 0 2	4 3 3		4 3 1

[NAT]

#Q. What will happen if process P3 requests one additional instance of resource type B ?

$Req_3 \langle 0,1,0 \rangle$

$\Downarrow$

granted

Before request system was safe and safe sequence was starting with P3 hence all valid requests of P3 will be always granted.

Topic : Resource Request Algorithm

1. If $Request_i$ <= $Need_i$
   Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2. If $Request_i$ <= Available
   Goto step (3); otherwise, P_i must wait, since the resources are not available.
3. Have the system pretend to have allocated the requested resources to process P_i by modifying the state as follows:
   Available = Available – Requesti
   $Allocation_i$ = $Allocation_i$ + $Request_i$
   $Need_i$ = $Need_i$– $Request_i$

Topic : Banker's Algorithm

Process	Allocation	Max	Available	Need
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	~~3 3 2~~ ~~2 3 0~~	7 4 3
P_1	~~2 0 0~~ 3 0 2	3 2 2	2 2 0	~~1 2 2~~ 0 2 0
P_2	3 0 2	9 0 2		6 0 0
P_3	~~2 1 1~~ 2 2 1	2 2 2		~~0 1 1~~ 0 0 1
P_4	0 0 2	4 3 3		4 3 1

System safe

[NAT]

#Q. What will happen if process P1 requests one additional instance of resource type A and two instances of resource type C ? $Req_1 \langle 1, 0, 2 \rangle$
What will happen if process P3 requests one additional instance of resource type B? $Req_3 \langle 0, 1, 0 \rangle$

Req of P1 and P3 both granted.

options for 2 Requests Req1 and Req2 from 2 different processes.

1. only Req1 granted (Req2 rejected)
2. only Req2 granted (Req1 rejected)
3. Both requests req1, req2 granted
4. only one of Req1, Req2 granted but not both.
5. Neither of req1, req2 granted.

Topic : Deadlock Detection

1. When all resources have single instance $\Rightarrow$ using wait-for-graph
2. When resources have multiple instances $\Rightarrow$ Banker's algo

Topic : Deadlock Detection

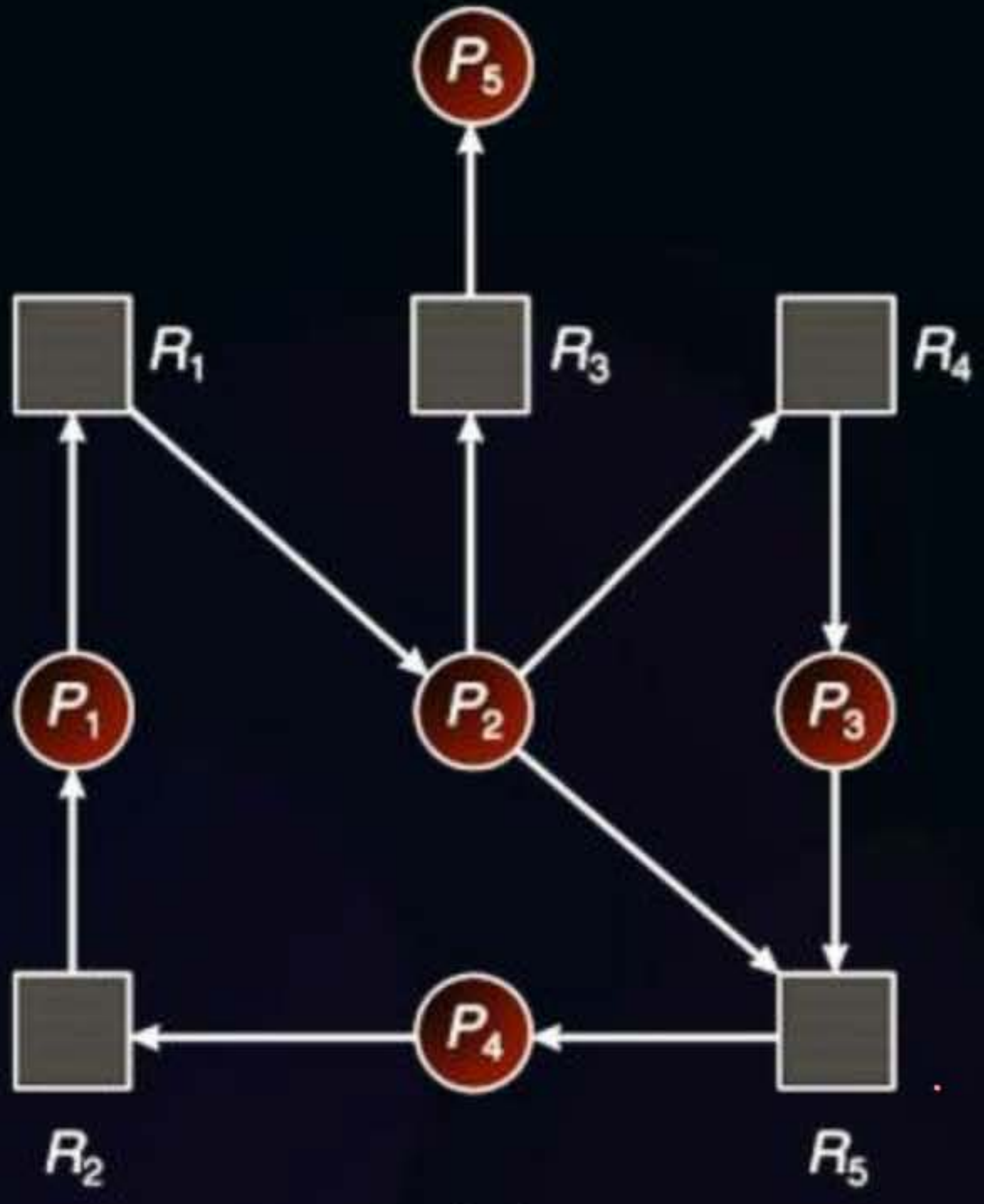
When all resources have single instance:

Deadlock detection is done using wait-for-graph

Topic : Wait For Graph

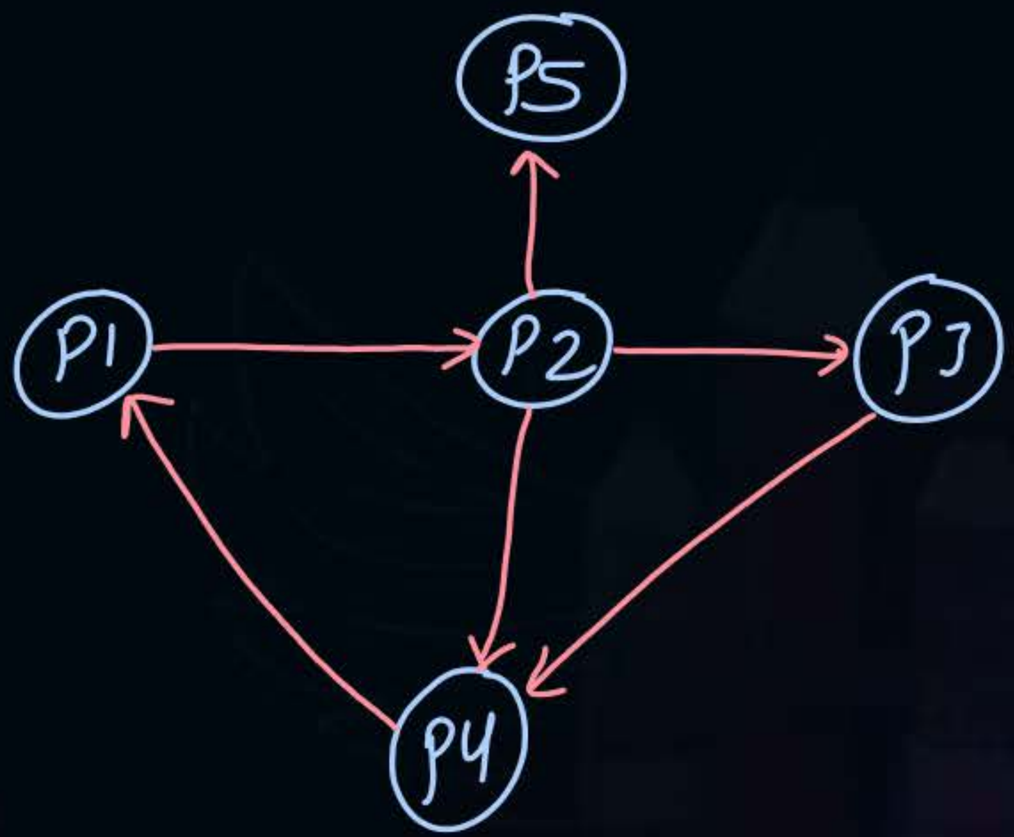
It is created from resource allocation graph

Topic : Wait For Graph



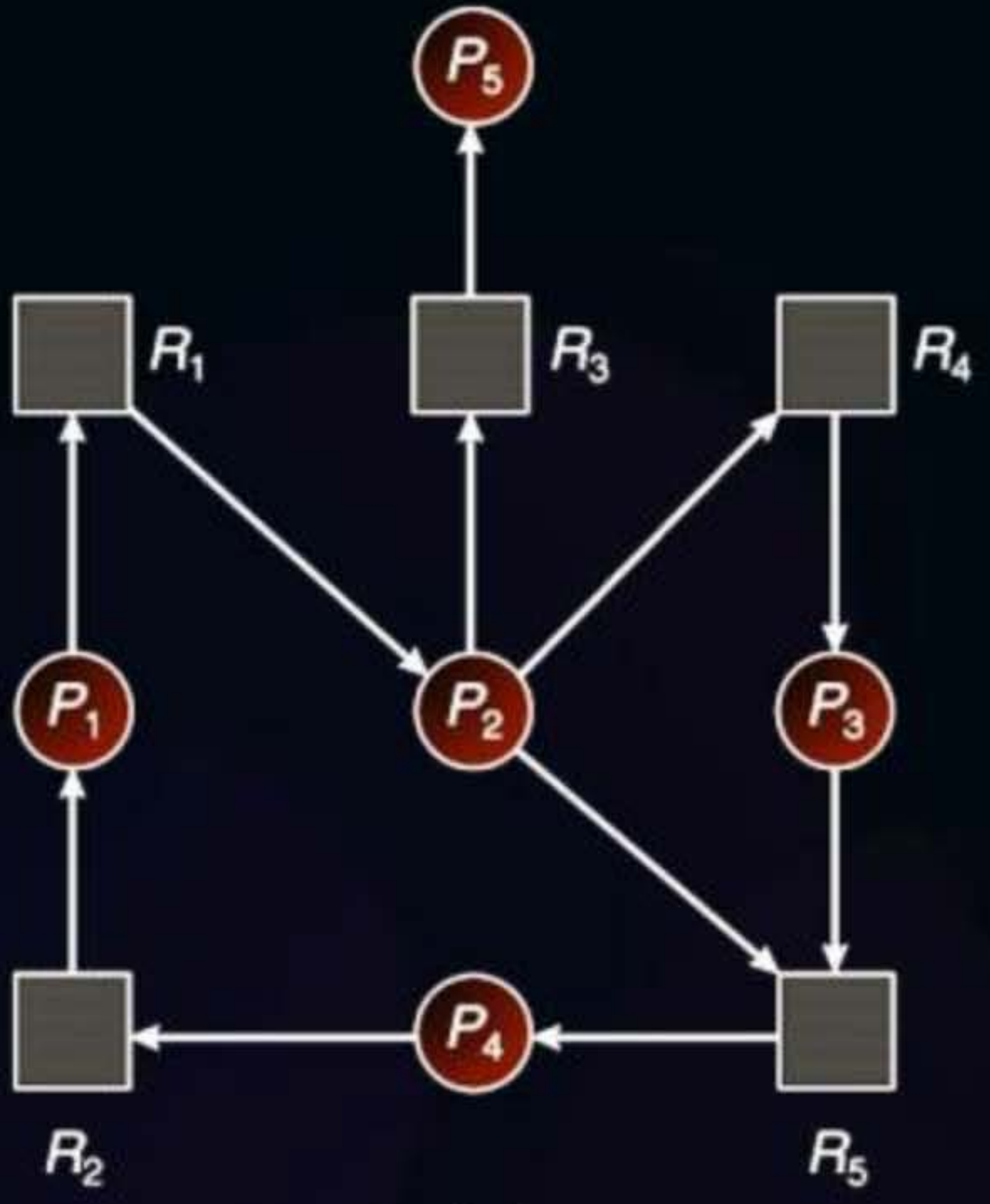
(a)

wait-for-graph has only processes as vertices



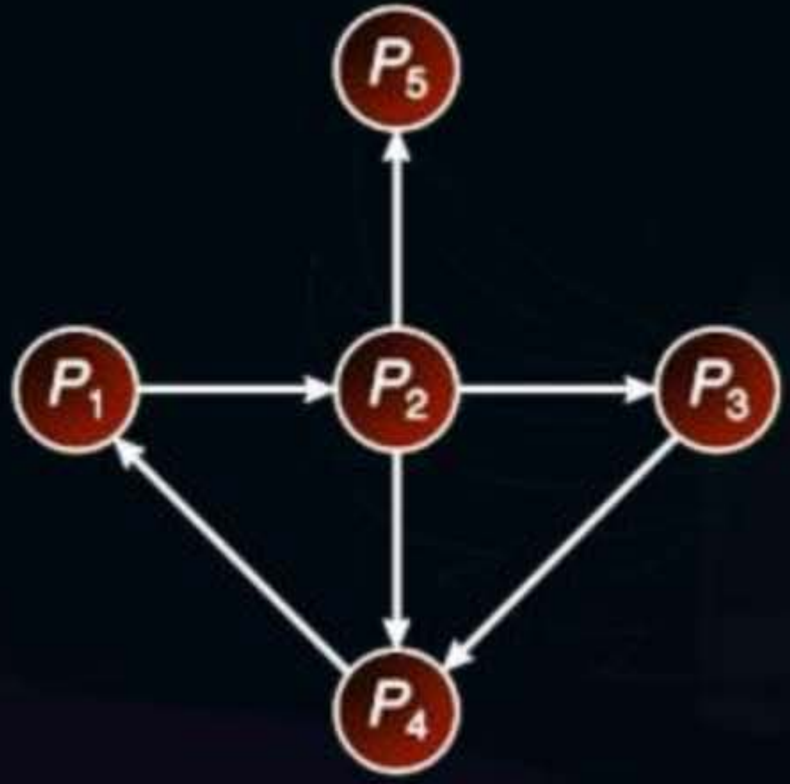
	Allocation					Request				
	R1	R2	R3	R4	R5	R1	R2	R3	R4	R5
P1	0	1	0	0	0	1	0	0	0	0
P2	1	0	0	0	0	0	0	1	1	1
P3	0	0	0	1	0	0	0	0	0	1
P4	0	0	0	0	1	0	1	0	0	0
P5	0	0	1	0	0	0	0	0	0	0

Topic : Wait For Graph



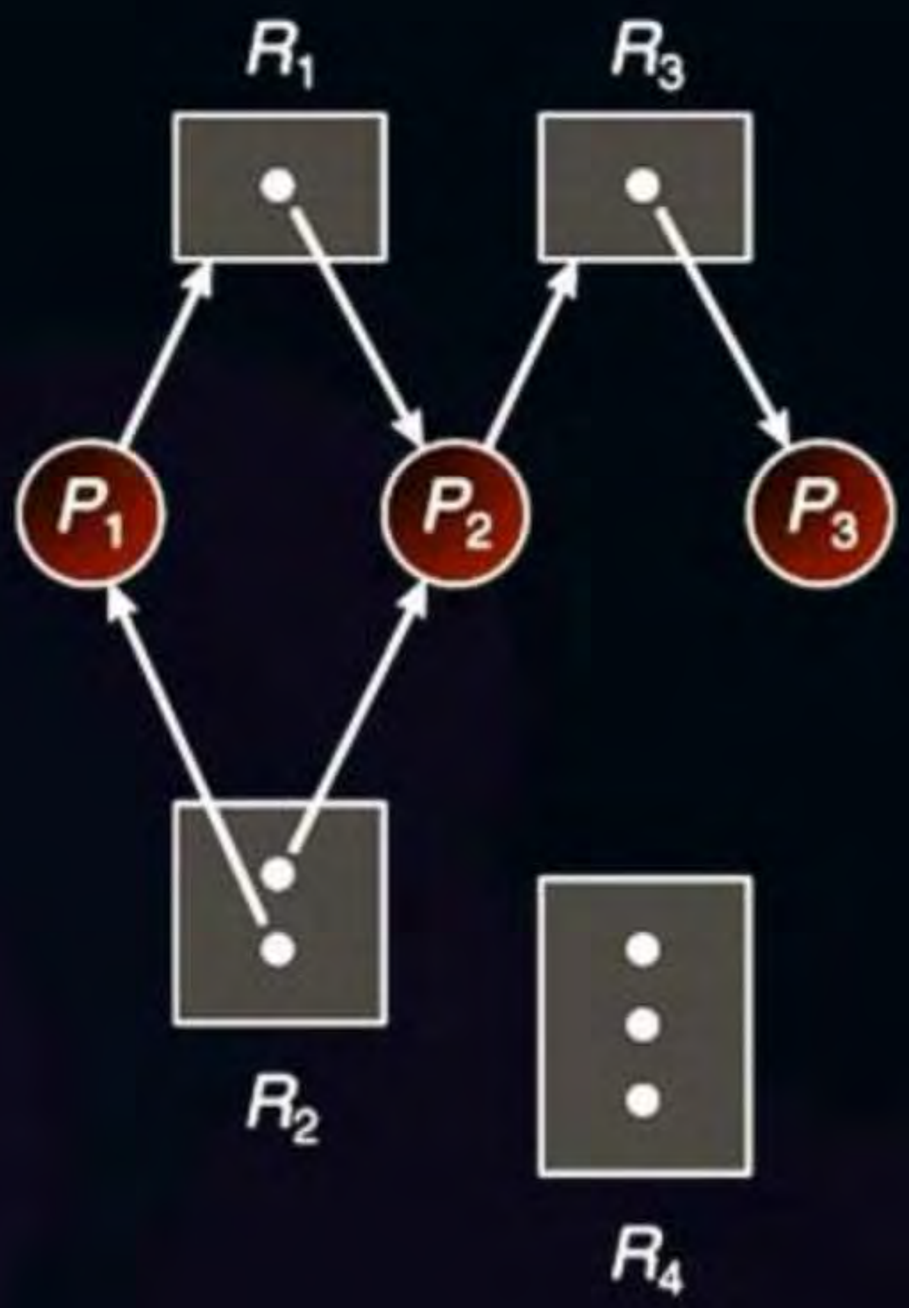
(a)

if cycle in wait-for-graph then deadlock



(b)

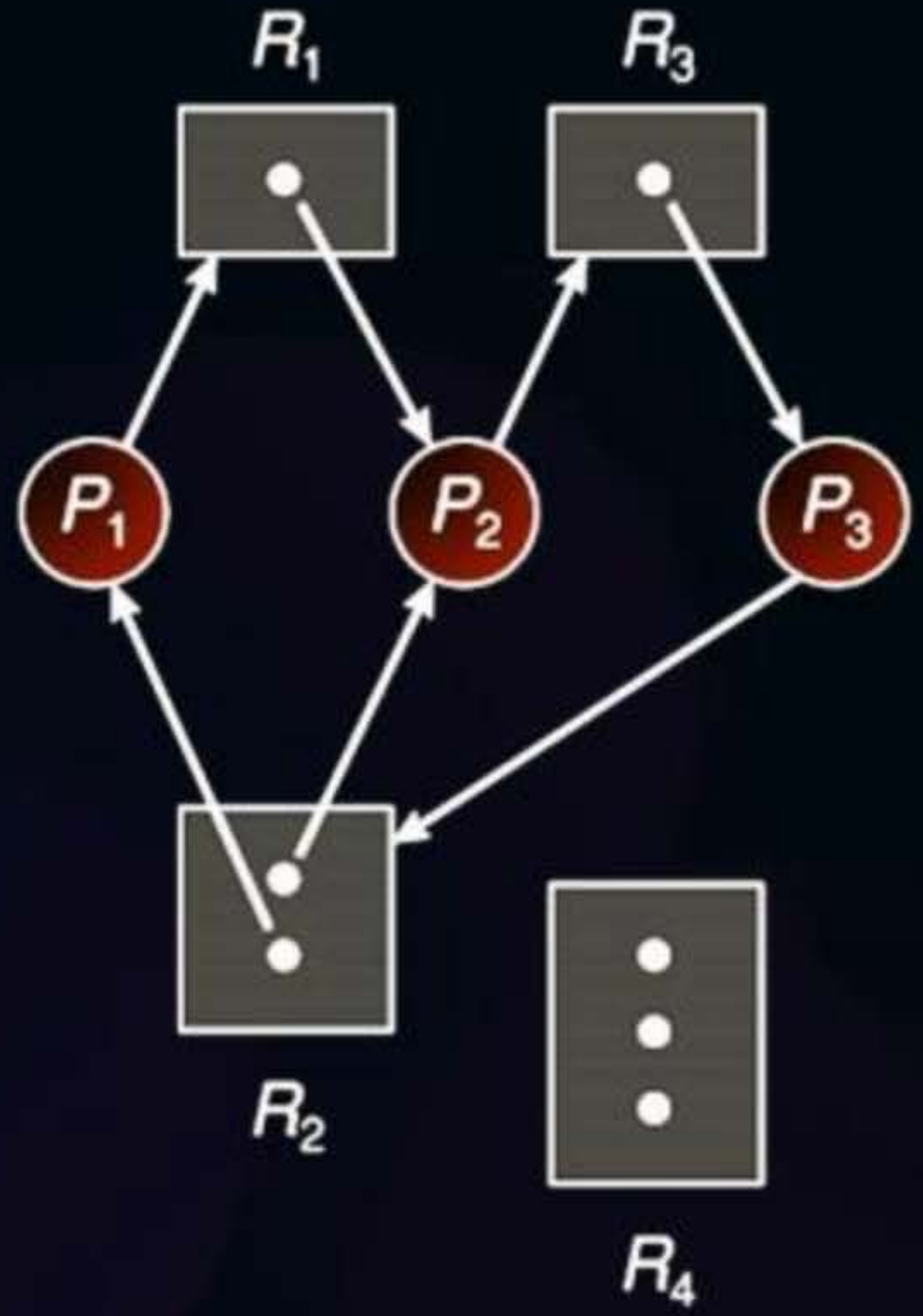
Topic : Wait For Graph

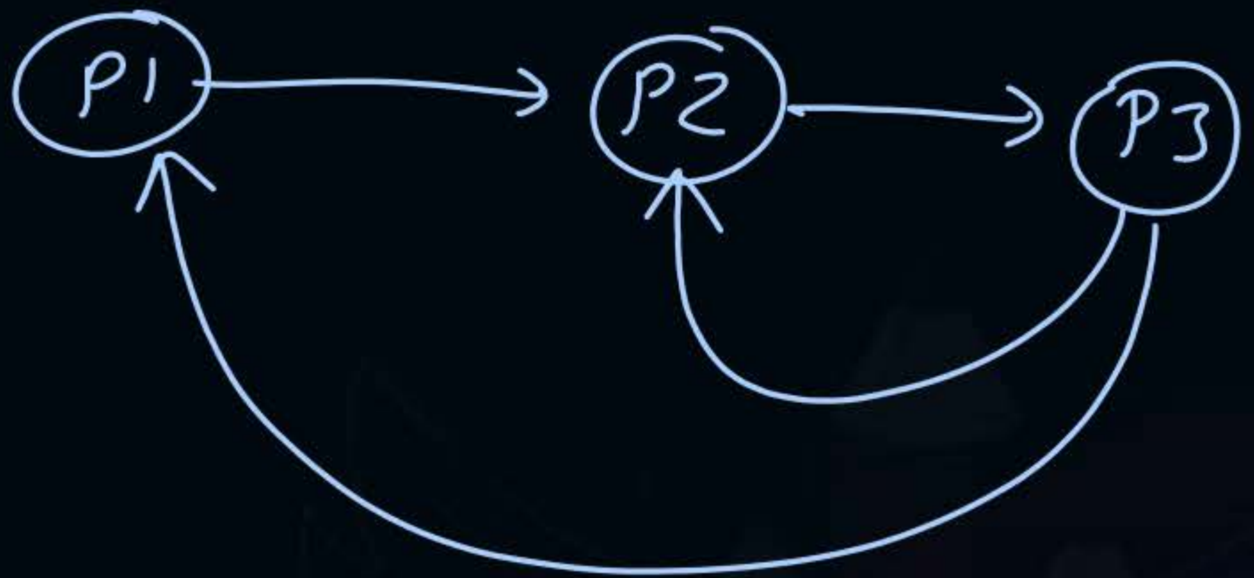


Topic : Wait For Graph

If a resource category contains more than one instance, then the presence of a cycle in the resource-allocation graph indicates the possibility of a deadlock, but does not guarantee one.

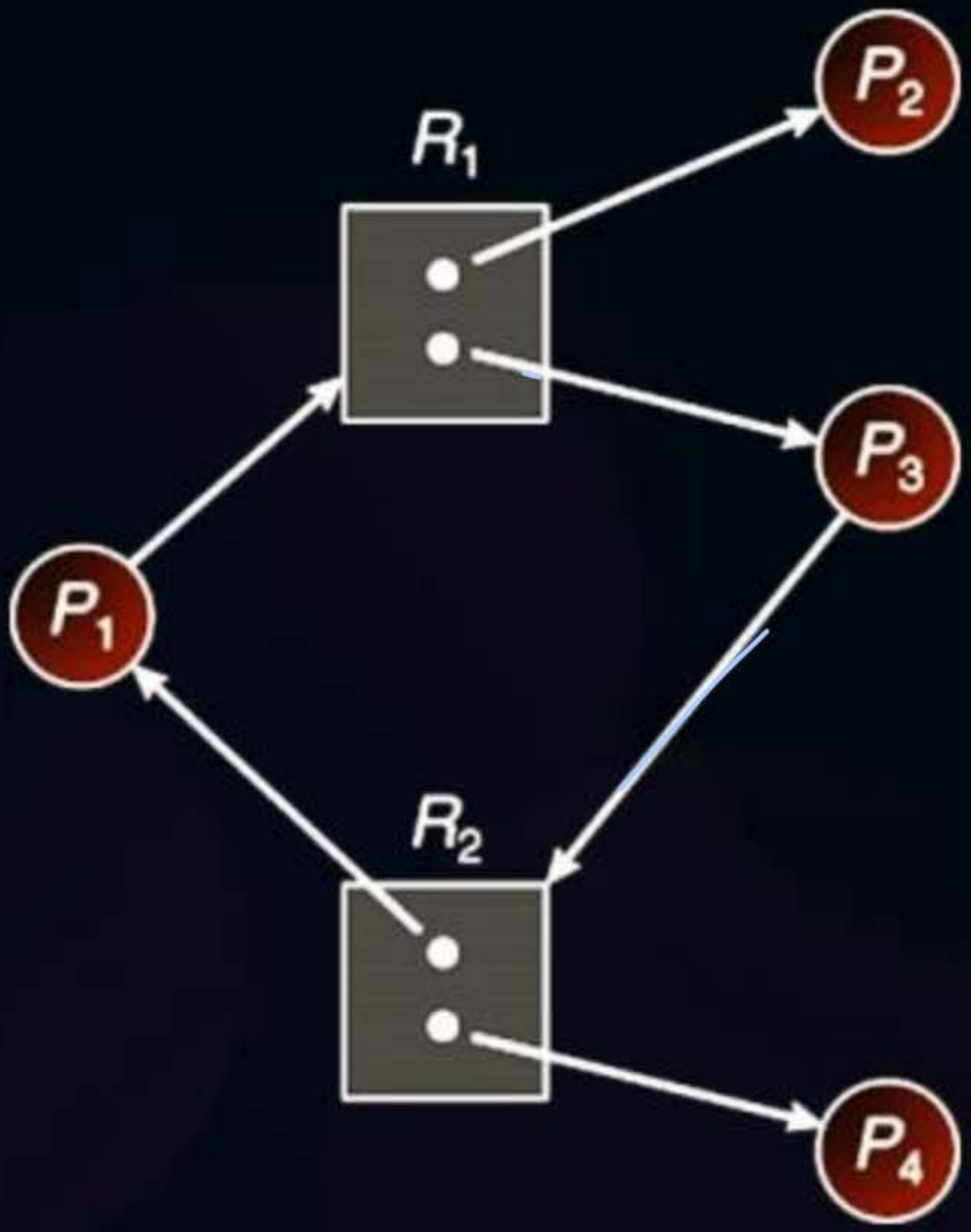
Topic : Wait For Graph: Example



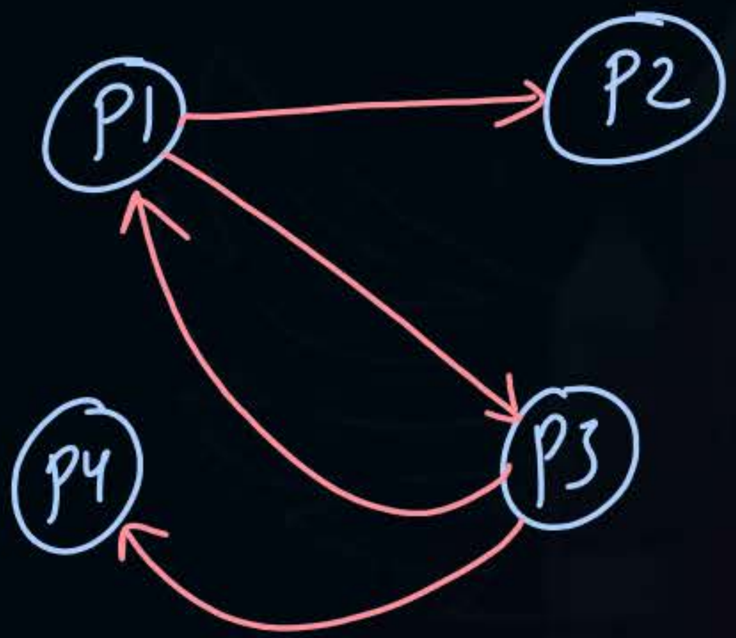


deadlock

Topic : Wait For Graph: Example



wait-for-graph



cycle but there is no any deadlock here.

Topic : Deadlock Detection

When resources have multiple instance:

Deadlock detection is done using a specific algorithm

Banker's

Topic : Deadlock Detection Algorithm

	Allocation	Request	Available
	A B C	A B C	A B C
P_0	0 1 0	0 0 0	~~0 0 0~~
P_1	2 0 0	2 0 2	after P0 ~~0 1 0~~
P_2	3 0 3	0 0 0	after P2 ~~3 1 3~~
P_3	2 1 1	1 0 0	after P1 5 1 3
P_4	0 0 2	0 0 2	after P3 7 2 4

after P4 7 2 6

find a process P_i with
$Request_i \leq Available$

$\Downarrow$

all processes completed
hence no deadlock.

$$\underbrace{\text{when no any process has } Req_i \le Available}_{\text{There will be deadlock}}$$

Topic : Deadlock Detection Algorithm

1. Let Work and Finish be vectors of length m and n respectively.
   Initialize Work= Available. For i=0, 1, ..., n-1, if Requesti = 0, then Finish[i] = true; otherwise, Finish[i]= false.
2. Find an index i such that both
   (a) Finish[i] == false
   (b) Requesti <= Work
   If no such i exists go to step 4.
3. Work= Work+ Allocationi
   Finish[i]= true
   Go to Step 2.
4. If Finish[i]== false for some i, 0<=i<n,
   then the system is in a deadlocked state.
   Moreover, if Finish[i]==false the process Pi is deadlocked.

Ques)

	Allocation A	Allocation B	Request A	Request B	Available A	Available B
P1	2	1	4	3	1	0
P2	1	0	2	2	after P3: 1	1
P3	0	1	1	0		
P4	1	1	2	1		

⇓

no any other process can run further

⇓

Deadlock ← (P1, P2, P4)

Topic : Detection-Algorithm Usage

When should the deadlock detection be done? Frequently, or infrequently?

Topic : Detection-Algorithm Usage

1. Do deadlock detection after every resource allocation
2. Do deadlock detection only when there is some clue

Topic : Recovery From Deadlock

There are three basic approaches to recovery from deadlock:

1. Inform the system operator and allow him/her to take manual intervention
2. Terminate one or more processes involved in the deadlock
3. Preempt resources.

Topic : Process Termination

Terminate all processes involved in the deadlock

Terminate processes one by one until the deadlock is broken

Topic : Process Termination

Many factors that can go into deciding which processes to terminate next:

1. Process priorities.
2. How long the process has been running, and how close it is to finishing.
3. How many and what type of resources is the process holding
4. How many more resources does the process need to complete
5. How many processes will need to be terminated
6. Whether the process is interactive or batch

Topic : Resource Preemption

Important issues to be addressed when preempting resources to relieve deadlock:

1. Selecting a victim
2. Rollback
3. Starvation

[NAT]

#Q. Consider a system with 3 processes A, B and C. All 3 processes require 4 resources each to execute. The minimum number of resources the system should have such that deadlock can never occur?

	Requirement	Allocation	Available
A	4	3	1
B	4	3	
C	4	3	
		9	

Total = 9 + 1

= 10

Assume n no. of processes $P_1 \dots P_n$

each process P_i needs R_i no. of resources to complete execution

min. no. of resources for which deadlock never occurs $= \sum_{i=1}^{n} (R_i - 1) + 1$

no. of resources for which deadlock never occurs $\geq \sum_{i=1}^{n} (R_i - 1) + 1$

max. no. of resources for which deadlock can occur $= \sum_{i=1}^{n} (R_i - 1)$

[NAT]

#Q. Consider a system with 4 processes A, B, C and D. All 4 processes require 6 resources each to execute. The maximum number of resources the system should have such that deadlock may occur?

		Allocatn
A	6	5
B	6	5
C	6	5
D	6	5
		20

Ans = 20

[NAT]

Ans = 2

#Q. Consider a system with 3 processes that share 4 instances of the same resource type. Each process can request a maximum of K instances. Resource instances can be requested and released only one at a time. The largest value of K that will always avoid deadlock is ____.

$$3 * (k-1) + 1 \leq 4$$

$$3 * (k-1) \leq 3$$

$$k-1 \leq 1$$

$$k \leq 2$$

$$\boxed{k_{max} = 2}$$

Topic : Types of Locks

1. Spinlock → busy waiting of a process
2. Livelock →
3. Deadlock → when 2 or more processes are blocked permanently.
4. ~~Semaphores~~
5. Reentrant Locks → A process can take same lock again.

while(——){ };

wait (s);

blocked → in block state

Livelock → 2 or more processes run on CPU concurrently but never proceeds further.

ex:- Binary semaphores

$S1 = \cancel{1}\ 0$

$S2 = \cancel{1}\ 0$

P1

wait(S1)
→ wait(S2)

signal(S1)
signal(S2)

P2

wait(S2)
→ wait(S1)

signal(S2)
signal(S1)

if execution

P1	P2	P1	P2	..
wait(S1)	wait(S2)			

2 mins Summary

Topic Deadlock Avoidance

Topic Banker's Safety Algorithm

Topic Banker's Resource Request Algorithm

Happy Learning

THANK - YOU