SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : OPERATING SYSTEMS

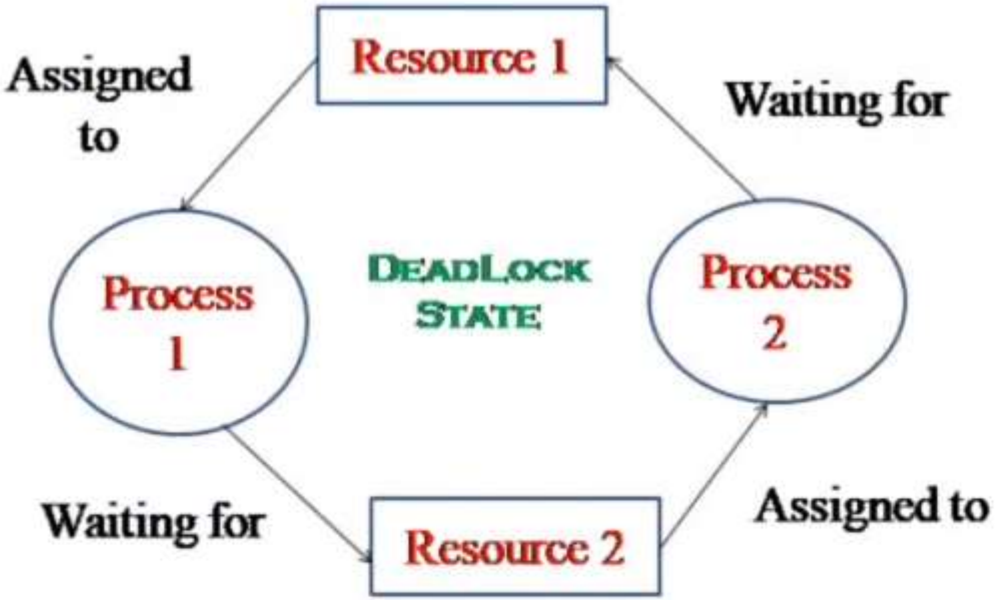
II YEAR/ IV SEMESTER

UNIT – II PROCESS SCHEDULING AND SYNCHRONIZATION

Topic: Deadlock System model – Deadlock characterization

Dr.B.Vinodhini
Associate Professor
Department of Computer Science and Engineering

Dead Lock



Dead lock is situation where a set of processes are blocked because each process is holding a resource and waiting for another resource and waiting for another resource acquired by some other process

Deadlock Characterization

1.Mutual Exclusion

2.No Preemption

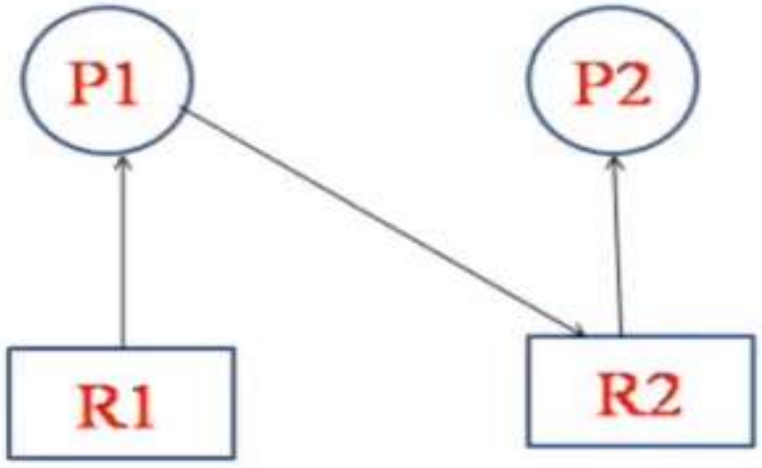
3.Hold&Wait

4.Circular Wait & Resource wait

Deadlock Characterization
Mutual Exclusion

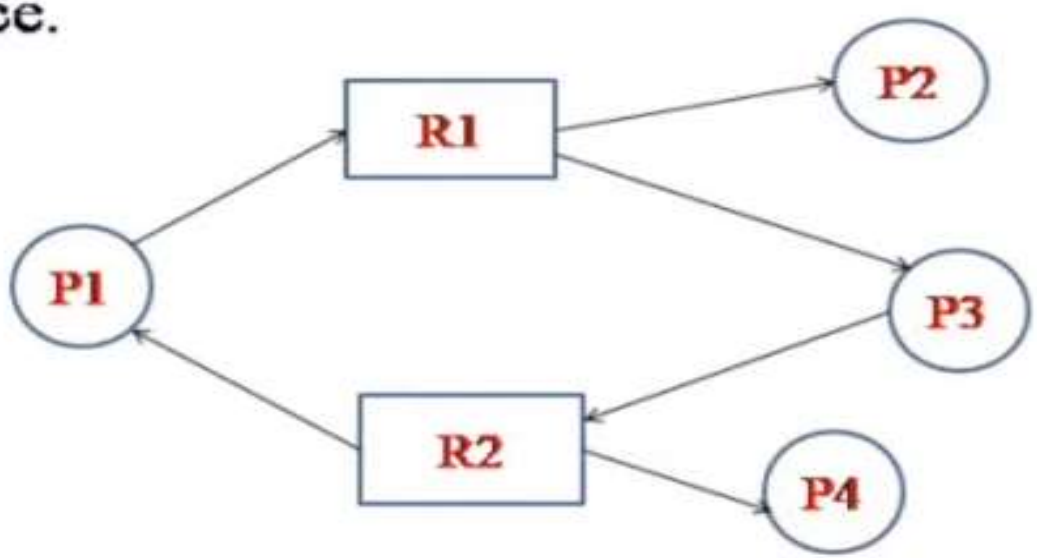
- ✓ One or more than one resource are non-sharable(Only one process can use at a time)

For Example :-



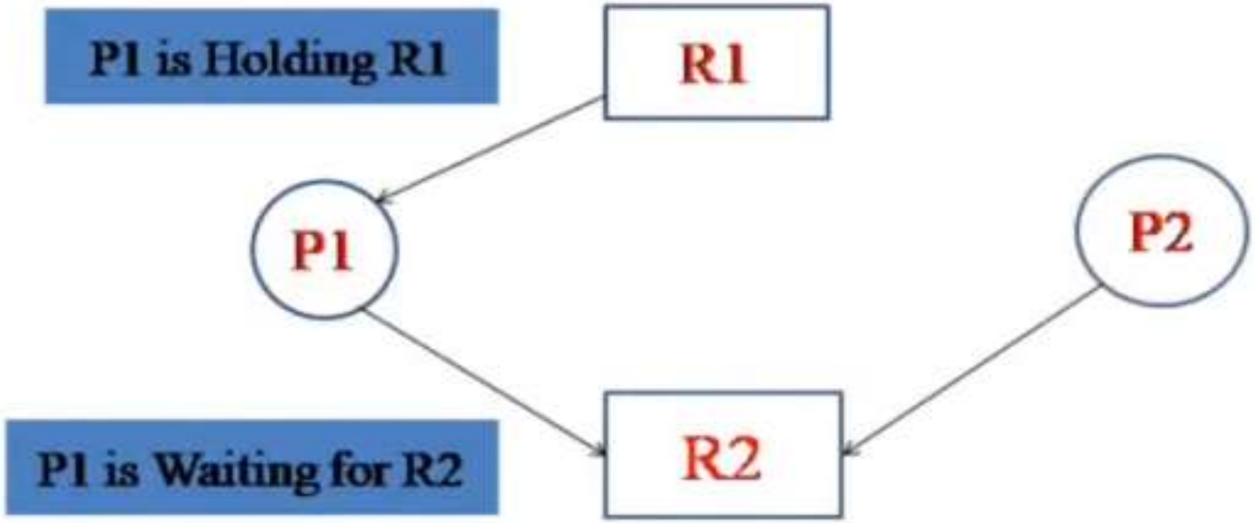
Deadlock Characterization
No Preemption

- ✓ Once a process has obtained a resources the system cannot remove it,from the process control until the process has finished using the resource.



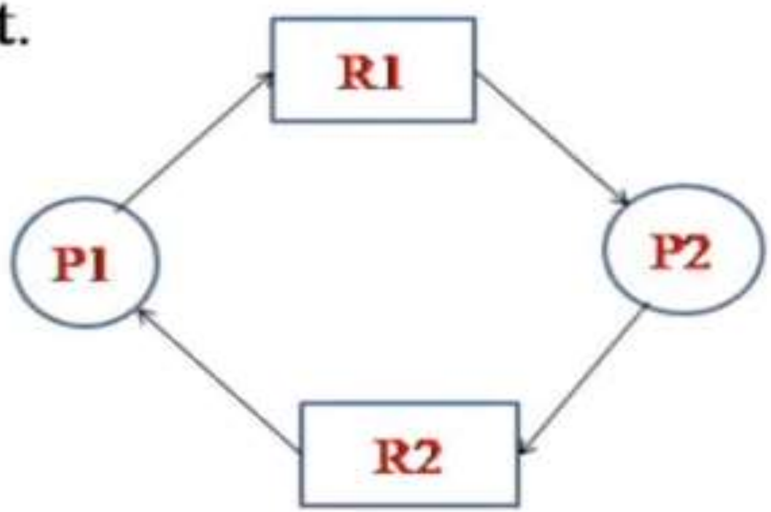
Deadlock Characterization
Hold & Wait

- ✓ A process holding at least one resource is waiting to acquire additional resources held by other process.



Deadlock Characterization
Hold & Wait

✓ The processes in the system form a circular list or chain where each process in the list is waiting for a resource held by the next process in the list.



Resource-Allocation Graph

- A set of vertices *V* and a set of edges *E*.

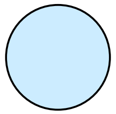
- V is partitioned into two types:
  - $P = \{P_1, P_2, \ldots, P_n\}$, the set consisting of all the **processes** in the system
  - $R = \{R_1, R_2, \ldots, R_m\}$, the set consisting of all **resource** types in the system

- **request edge** – directed edge $P_i \rightarrow R_j$

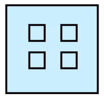
- **assignment edge** – directed edge $R_j \rightarrow P_i$

Resource-Allocation Graph (Cont.)

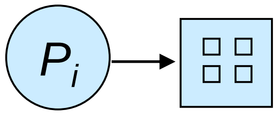
- Process



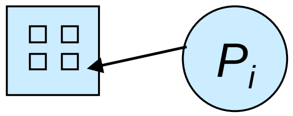
- Resource Type with 4 instances



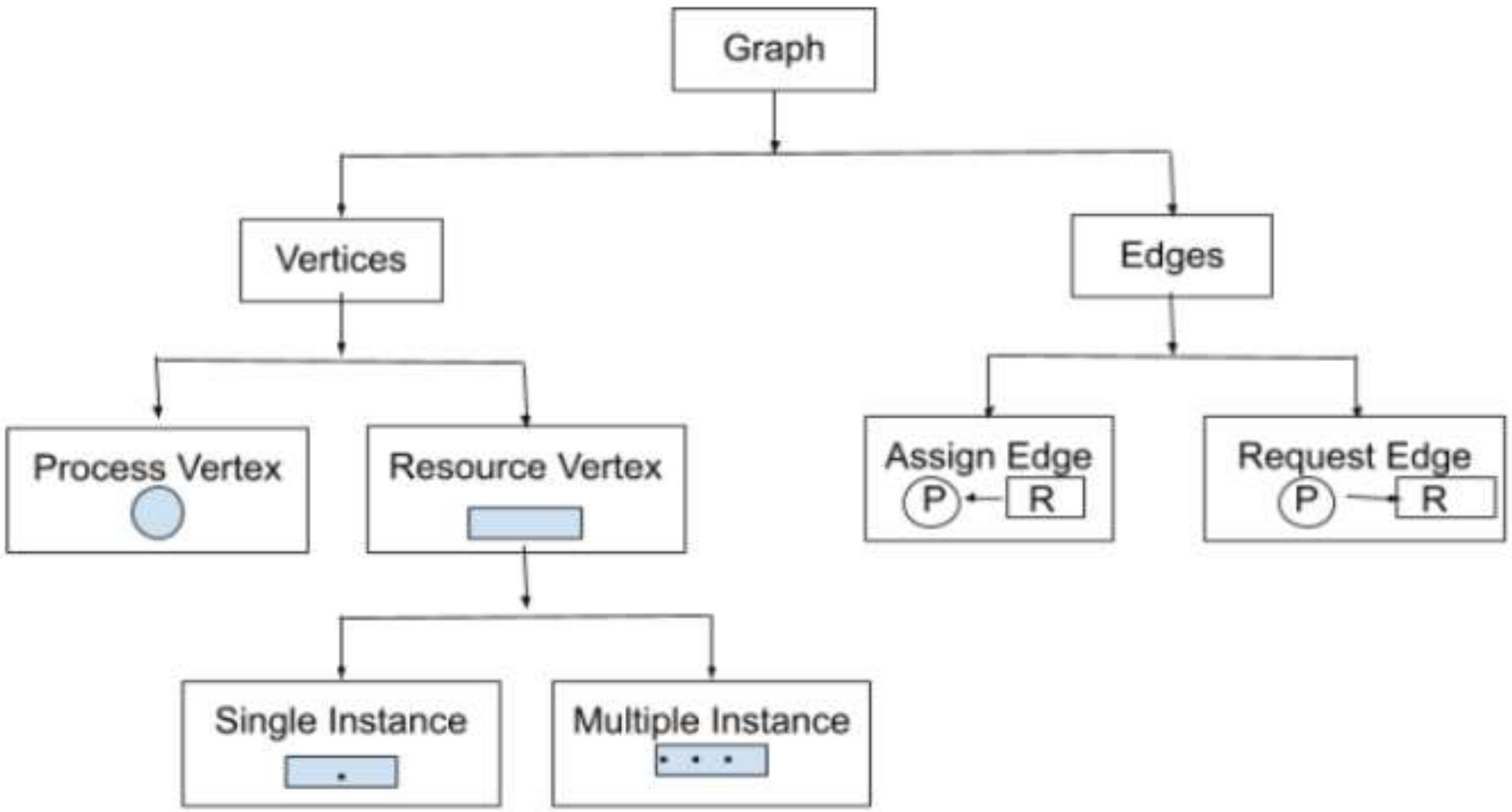
- P_i requests instance of R_j



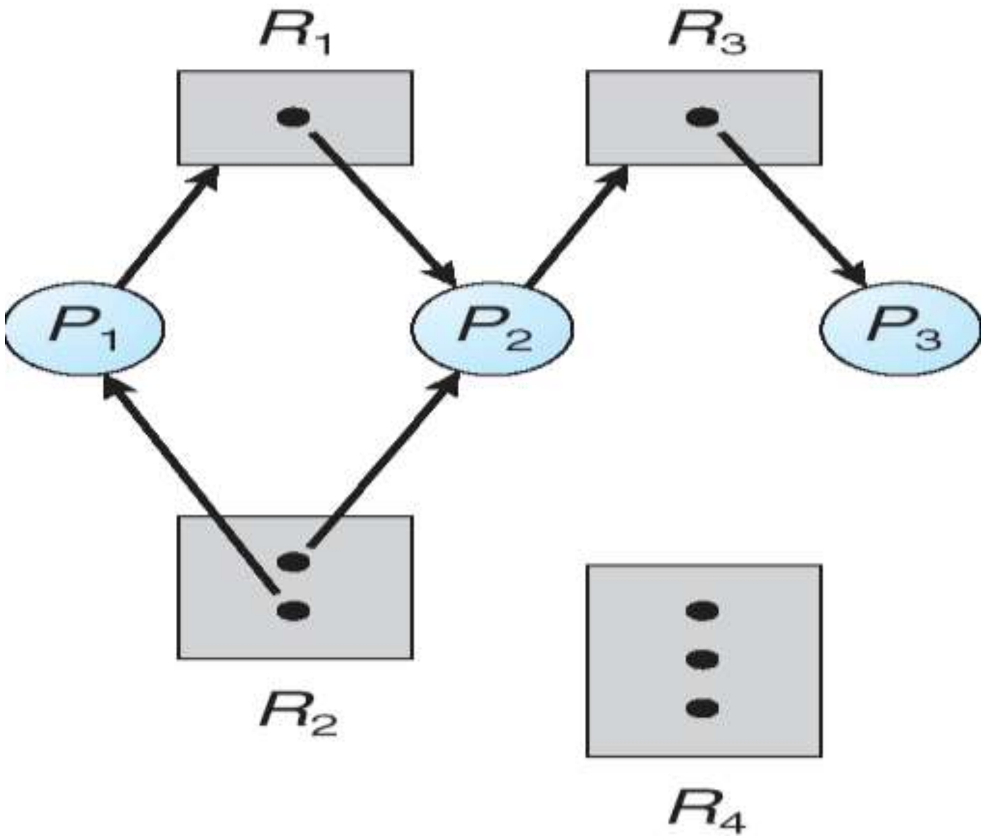
- P_i is holding an instance of R_j



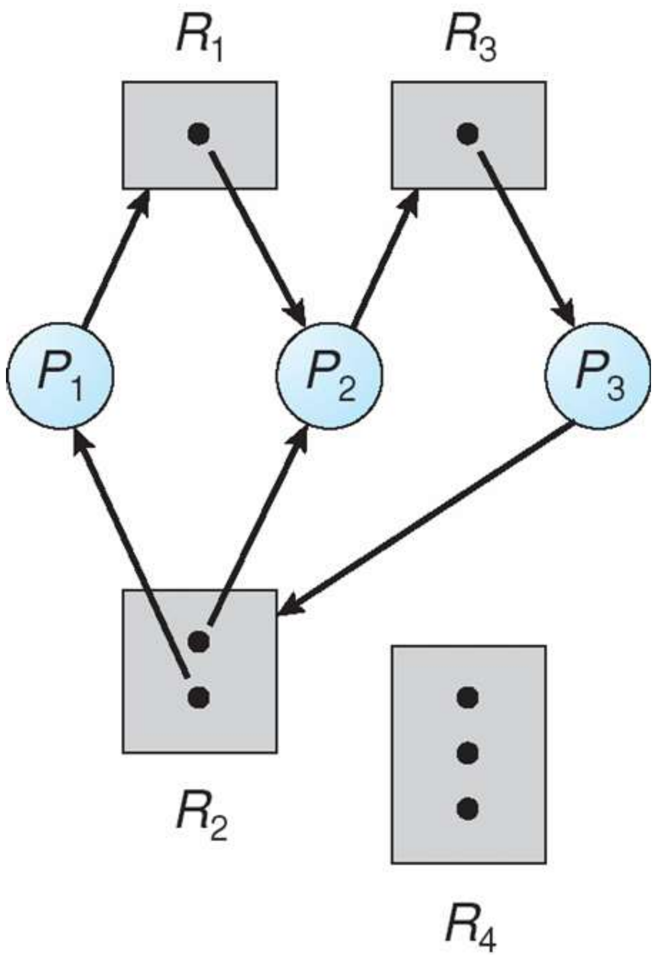
Resource-Allocation Graph



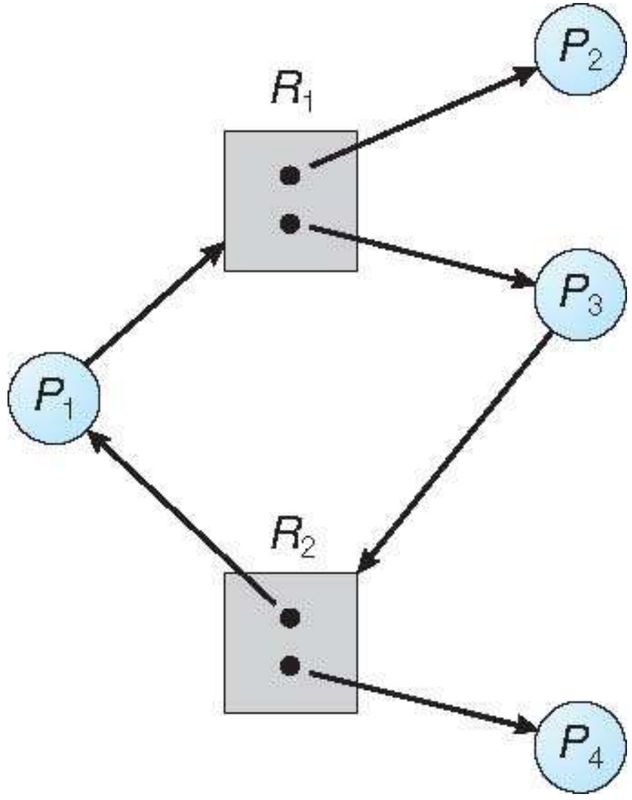
Example of a Resource Allocation Graph



Resource Allocation Graph With A Deadlock



Graph With A Cycle But No Deadlock

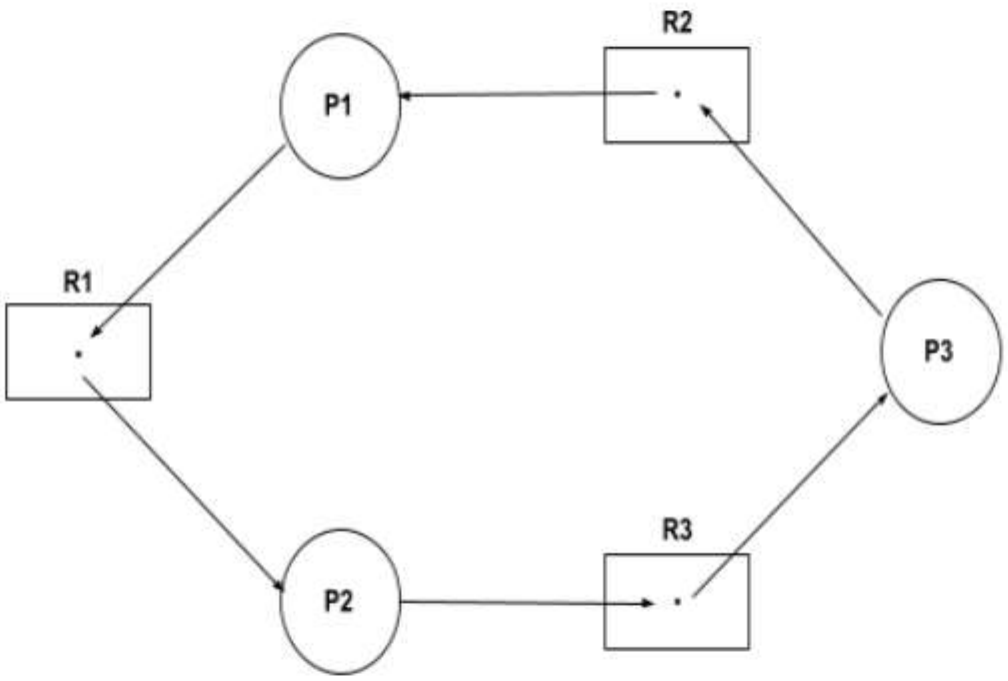


If graph contains **no cycles** ⇒ **no deadlock**

If graph contains a **cycle** ⇒

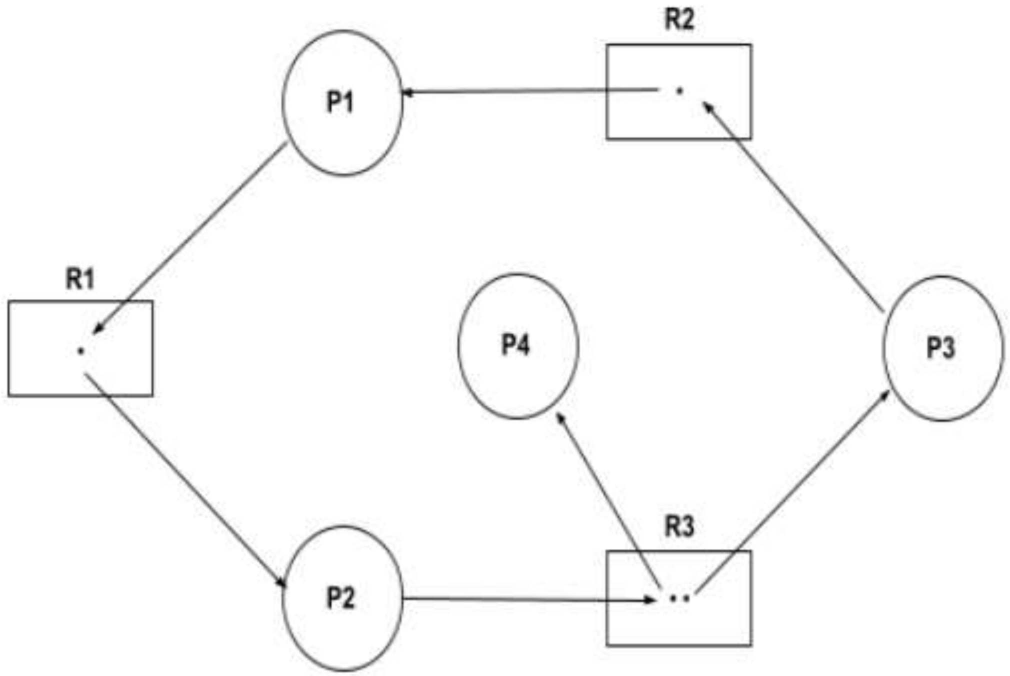
- if only **one instance per resource type**, then deadlock
- if several instances per resource type, possibility of deadlock

Single Instances RAG



Process	Allocation Resource			Request Resource		
	R1	R2	R3	R1	R2	R3
P1	0	1	0	1	0	0
P2	1	0	0	0	0	1
P3	0	0	1	0	1	0

Multiple Instances RAG



Process	Allocation			Request		
	Resource			Resource		
	R1	R2	R3	R1	R2	R3
P1	0	1	0	1	0	0
P2	1	0	0	0	0	1
P3	0	0	1	0	1	0
P4	0	0	1	0	0	0

Algorithm to check deadlock

1. First, find the currently available instances of each resource.
2. Check for each process which can be executed using the allocated + available resource.
3. Add the allocated resource of the executable process to the available resources and terminate it.
4. Repeat the 2nd and 3rd steps until the execution of each process.
5. If at any step, none of the processes can be executed then there is a deadlock in the system.

References

1. Silberschatz, Galvin, and Gagne, “Operating System Concepts”, Ninth Edition, Wiley India Pvt Ltd, 2009.
2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition, Pearson Education, 2010.


Thank you