22 TUESDAY • JANUARY

Resource - Allocation Graph

- Deadlock can be described more precisely in terms of a directed graph called a system resource-allocation graph.
- This graph consists of a set of vertices V and a set of edges E.
- The set of vertices V is partitioned into two different types of nodes: P = {P1, P2, ..., Pn}, the set consisting of all the active processes in the system, and R = {R1, R2, ..., Rm}, the set consisting of all resource types in the system.

* A directed edge from processes Pi to resource type Rj is denoted by Pi → Rj, it signifies that process Pi has requested an instance of resource type Rj and is currently waiting for that resource.
* A directed edge from resource type Rj to process Pi is denoted by Rj → Pi, it signifies that an instance of resource type Rj has been allocated to process Pi
* A directed edge Pi → Rj is called a request edge; a directed edge Rj → Pi is called an assignment edge.

- We represent each process P_i as a circle and each resource type R_j as a rectangle.
- Since resource type R_j may have more than one instance, we represent each such instance as a dot within the rectangle.
- Note that a request edge points to only the rectangle R_j, whereas an assignment edge must also designate one of the dots in the rectangle.

The resource-allocation graph shown in figure below depicts the following situation:

- The sets P, R and E:
  - $P = \{P_1, P_2, P_3\}$
  - $R = \{R_1, R_2, R_3, R_4\}$
  - $E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$

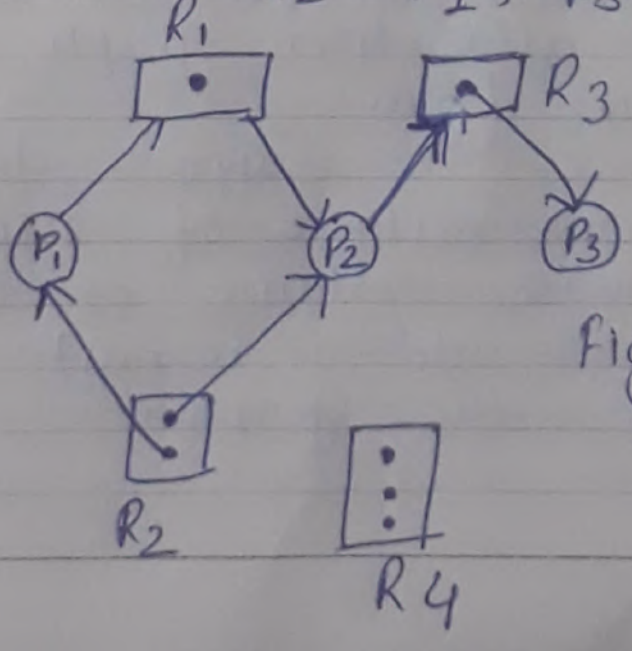

Fig: Resource-allocation graph.

- Resource instances :

→ One instance of resource type R_1.
→ Two instance of resource type R_2.
→ One " " " R_3.
→ Three " " " R_4.

- Process states :

→ Process P_1 is holding an instance of resource type R_2 and is waiting for an instance of resource type R_3.

→ Process P_2 is holding an instance of R_1 and an instance of R_2 and is waiting for an instance of R_3.

→ Process P_3 is holding an instance of R_3.

- To illustrate this concept, we return to the resource-allocation graph depicted on previous figure.

Suppose that process P_3 requests an instance of resource type R_2. Since no resource instance is currently available, we add a request edge $P_3 \rightarrow R_2$ to the graph.

- At this point, two minimal cycles exist in the system:

$P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$

$P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_2$

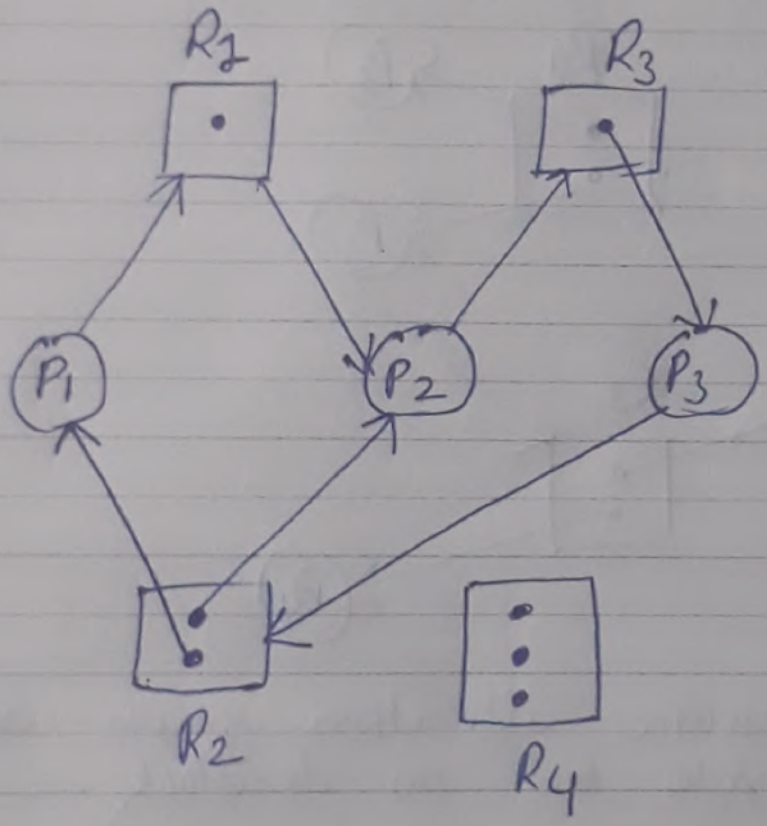

Fig: Resource-allocation graph with a deadlock.

- Processes P_1, P_2, and P_3 are deadlocked. process P_2 is waiting for the resource R_3, which is held by process P_3. Process P_3 is waiting for either process P_1 or process P_2 to release resource R_2. In addition, Process P_1 is waiting for process P_2 to release resource R_1.

Now Consider the resource-allocation graph in this figure. In this example, we also have a cycle:

$$P_1 \rightarrow R_1 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$$

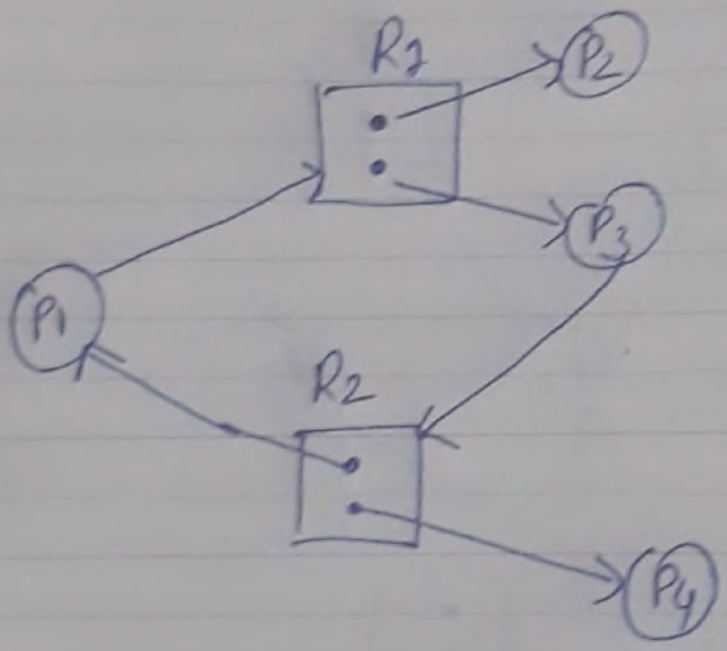

Fig: Resource-allocation graph with a cycle but no deadlock.