CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Deadlock Visualization

Necessary conditions for deadlock

- mutual exclusion
- hold and wait
- no preemption
- circular wait

To detect deadlock, it is helpful to model system resources and processes in a directed graph

Nodes are one of two types:

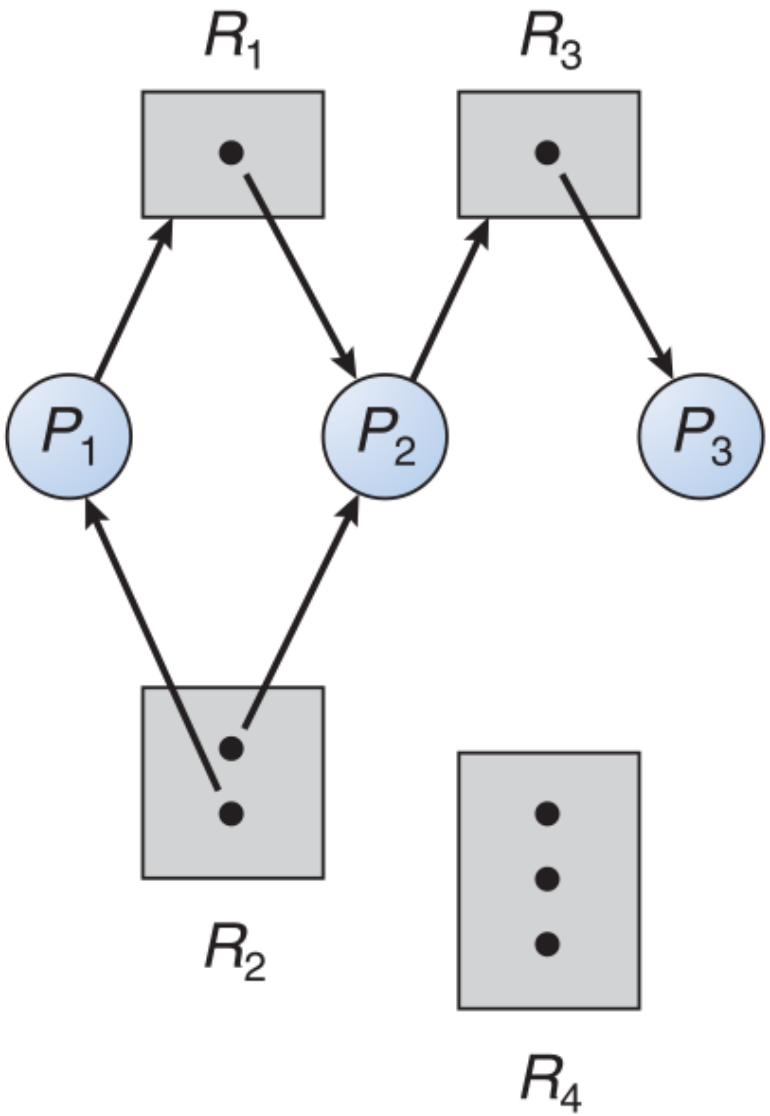
- processes (shown as circles)
- resources (shown as rectangles)

Resources can have multiple instances, which show up as dots inside rectangle

Edges are directed:

- resource -> process indicates process posseses resource
- process -> resource indicates process needs resource, but does not have it

So, arrow from process to resource implies that process is waiting on that resource


R_1
R_3
P_1
P_2
P_3
R_2
R_4

Edges in graph will

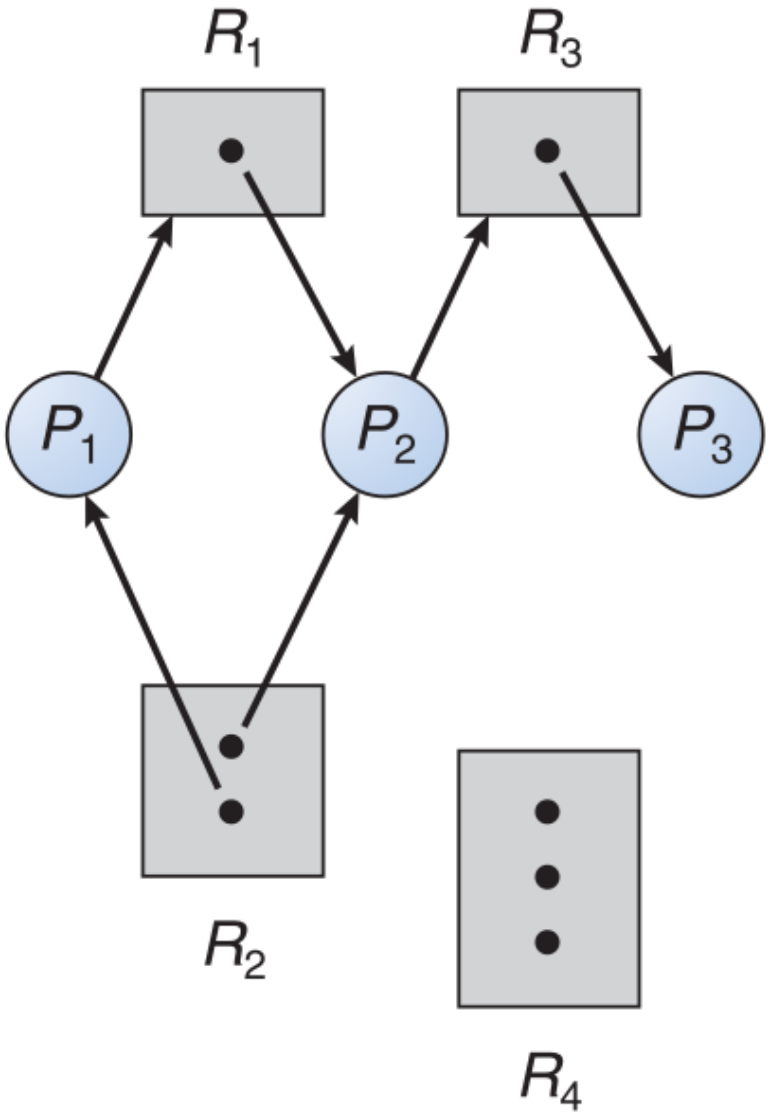
- appear when process requests resource
- change directions instantly when resource is acquired
- disappear when resource is released

These graphs allow us to detect a circular wait

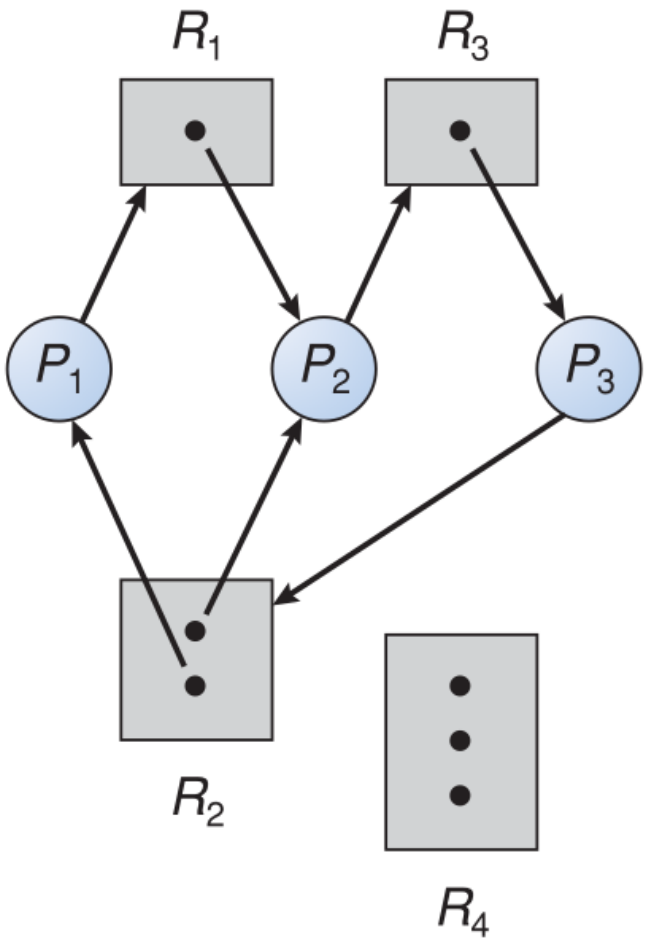
Circular wait shows up as a cycle in the graph

When we can assume our first three conditions for deadlock apply, a circular wait indicates that we have a *possible* deadlock

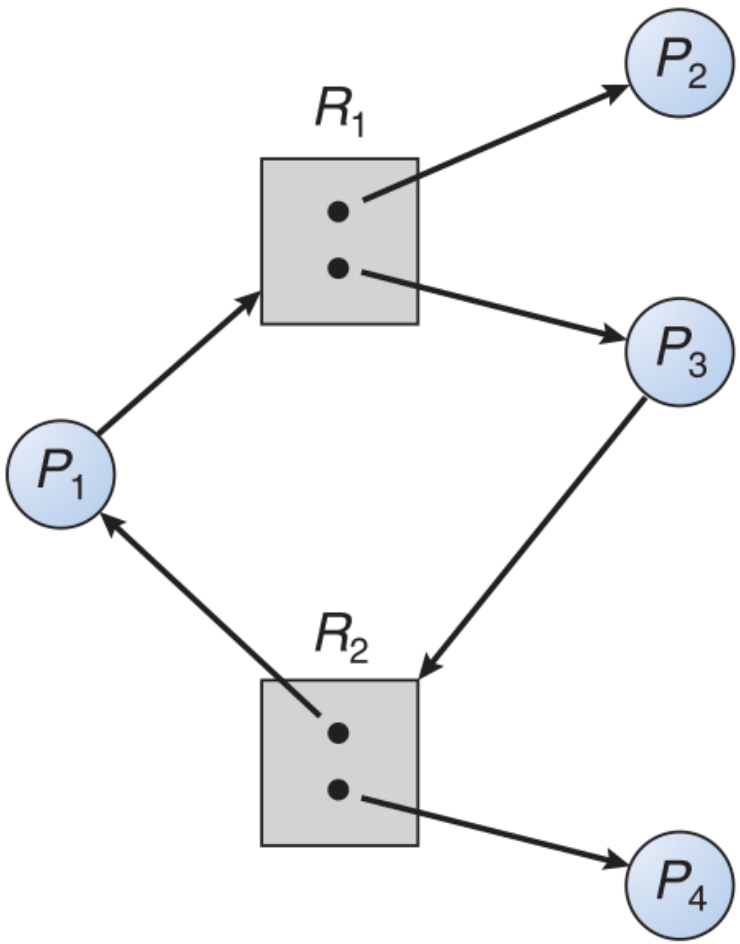
Original image


R_1
R_3
P_1
P_2
P_3
R_2
R_4

Deadlock



Cycle, but no deadlock

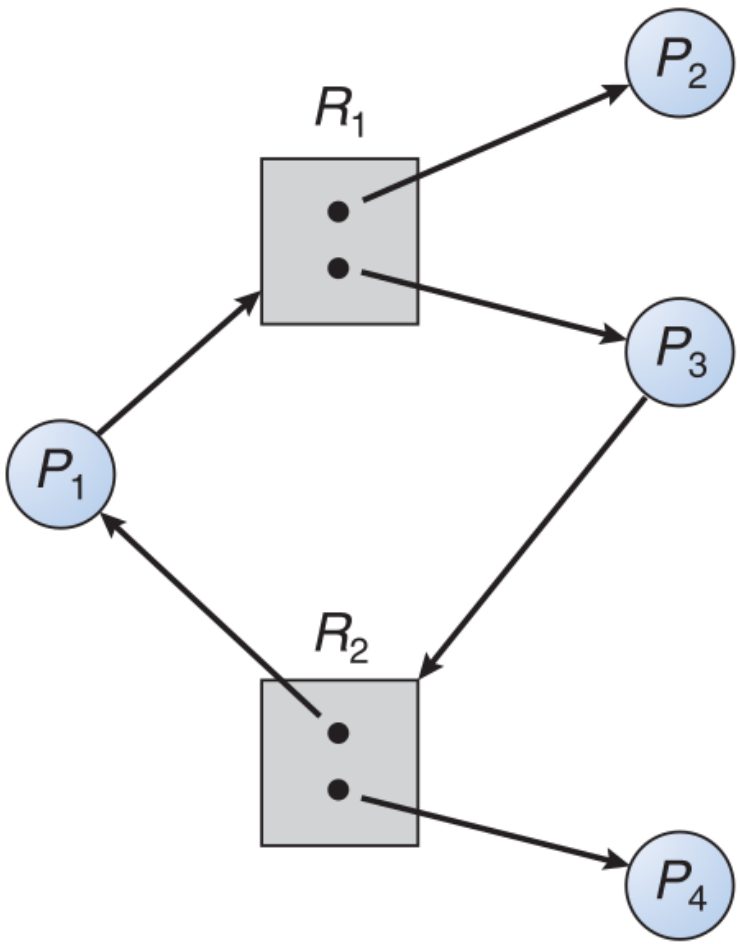


If just one instance of each resource exists (or just one instance of resources involved in cycle), then a cycle implies deadlock

Otherwise, cycle may or may not be deadlock

Be sure to recall the difference between a necessary and a sufficient condition

Add edge(s) to create deadlock



Ways of dealing with deadlock in the OS

- prevent or avoid deadlock entirely,
- detect deadlock and recover, or
- ignore the problem

OS can prevent deadlock by constraining how requests for resources can be made

Deadlocks can be avoided if the OS knows in advance which resources a process will request

Recovering from deadlock requires periodically checking system state for deadlocks and recovering if they exist

Recovering can include preempting a resource or restarting a process

Undetected deadlocks degrade system performance and can lead to even more deadlocks

However, most modern OSes do ignore the problem

Deadlocks are rare enough in practice that the expense of avoiding or recovering from them exceeds the gain