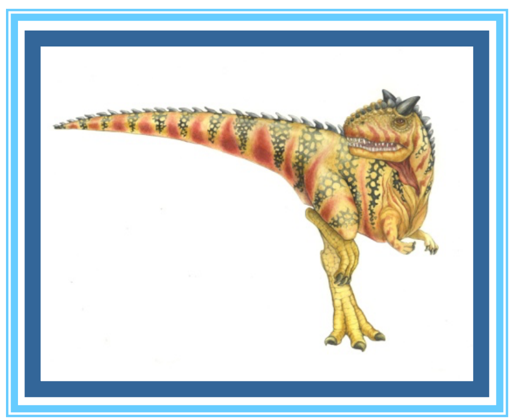
RECAP Chapter 7: Deadlocks



Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously.

- **Mutual exclusion**: only one process at a time can use a resource
- **Hold and wait**: a process holding at least one resource is waiting to acquire additional resources held by other processes
- **No preemption**: a resource can be released only voluntarily by the process holding it, after that process has completed its task
- **Circular wait**: there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, …, P_{n-1} is waiting for a resource that is held by P_n, and P_n is waiting for a resource that is held by P_0.

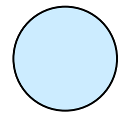
Resource-Allocation Graph

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

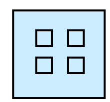
- V is partitioned into two types:
  - $P = \{P_1, P_2, \ldots, P_n\}$, the set consisting of all the processes in the system
  - $R = \{R_1, R_2, \ldots, R_m\}$, the set consisting of all resource types in the system
- **request edge** – directed edge $P_i \rightarrow R_j$
- **assignment edge** – directed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

Resource-Allocation Graph (Cont.)

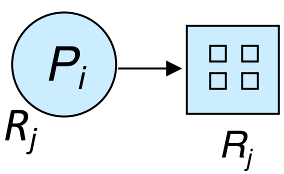
- Process



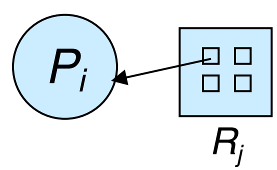
- Resource Type with 4 instances



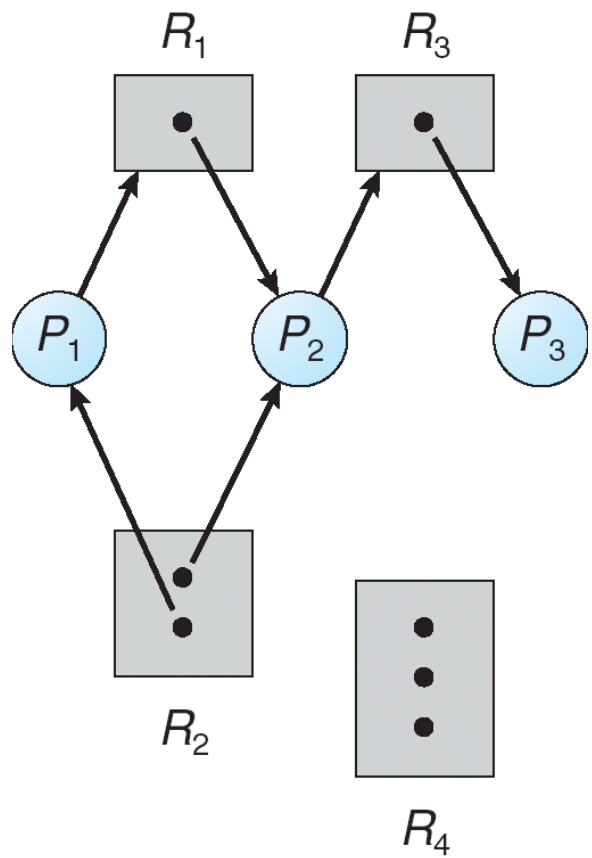
- P_i requests instance of R_j



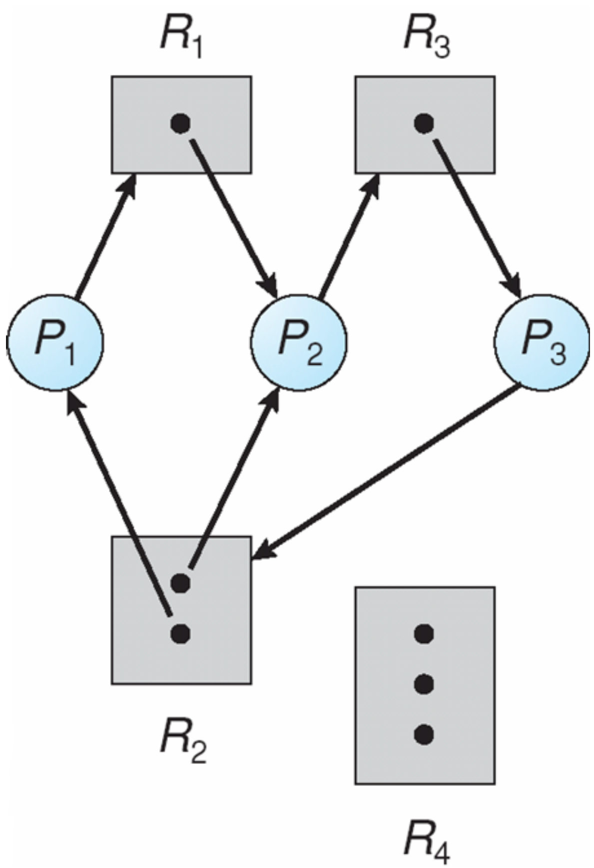
- P_i is holding an instance of R_j



Example of a Resource Allocation Graph

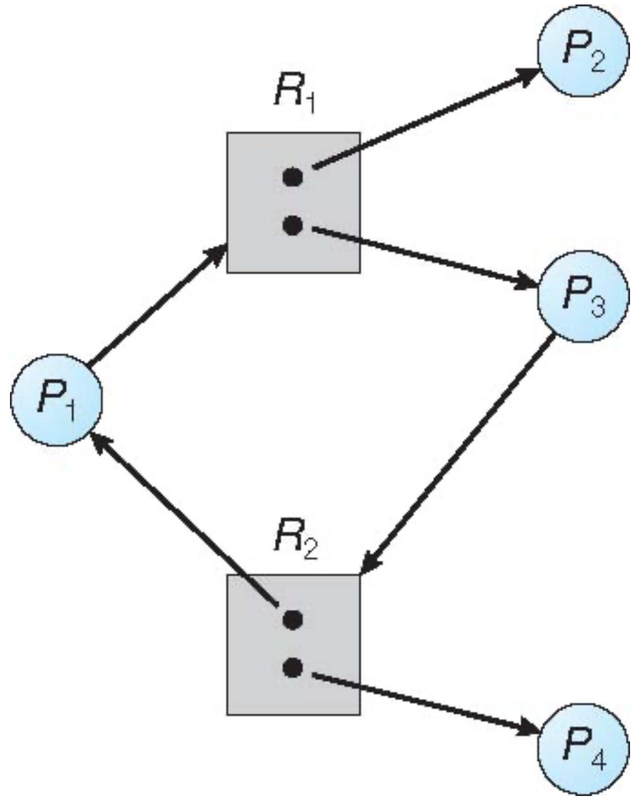


Resource Allocation Graph With A Deadlock



Is there a deadlock?
How can we detect a deadlock?
Is it a iff condition?

Graph With A Cycle But No Deadlock



Basic Facts

- If graph contains no cycles $\Rightarrow$ no deadlock
- If graph contains a cycle $\Rightarrow$
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, possibility of deadlock

Methods for Handling Deadlocks

- Ensure that the system will ***never*** enter a deadlock state:
  - Deadlock prevention: Avoid all four conditions simultanesouly holding
  - Deadlock avoidance: Allow resource allocation after checking that it wont lead to a possibility of a deadlock
- Allow the system to enter a deadlock state and then recover
- Ignore the problem and pretend that deadlocks never occur in the system; used by most operating systems, including UNIX

Deadlock Avoidance

Requires that the system has some additional ***a priori*** information available

- Simplest and most useful model requires that each process declare the ***maximum number*** of resources of each type that it may need
- The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition
- Resource-allocation *state* is defined by the number of available and allocated resources, and the maximum demands of the processes

Handling Deadlocks in Distributed Systems

What we deal with in this course ?

- Detect deadlocks
  - -Don’t detect false (phantom)deadlocks
  - -Don’t miss detecting an existing deadlock
- Recover from deadlocks

Example- Deadlock

Process A

- Lock A
- Lock B
- Transfer
- Unlock A
- Unlock B

Process B

- Lock B
- Lock A
- Transfer
- Unlock B
- Unlock A

To the Distributed Setting

- A distributed program is composed of a set of n asynchronous processes p1, p2, . . . , pi, . . . , pn that communicate by message passing over the communication network.
- Without loss of generality we assume that each process is running on a different processor.
- The processors do not share a common global memory and communicate solely by passing messages over the communication network

To The Distributed Setting

- There is no physical global clock in the system to which processes have instantaneous access.

- The communication medium may deliver messages out of order, messages may be lost garbled or duplicated due to timeout and retransmission, processors may fail and communication links may go down.

To The Distributed Setting

- We make the following assumptions:
  - The systems have only reusable resources.
  - Processes are allowed to make only exclusive access to resources.
  - **There is only one copy of each resource**
- Hence a system is deadlocked iff there exists a cycle or knot in WFG

Resource Request Models

The manner in which a task asks for resources can be

Single Resource request Model: can ask at a time only for one resource. can thus have at most one pending resource request

AND Resource request Model: Can request multiple resources. Task resumed only when all are acquired.

Example: (R1 AND R2 AND R3)

OR Resource request Model: Can request multiple resources. Task can resume when any one of the request resources becomes available

Example: (R1 *or* R2) or (R1 *or* R2 *or* R3)

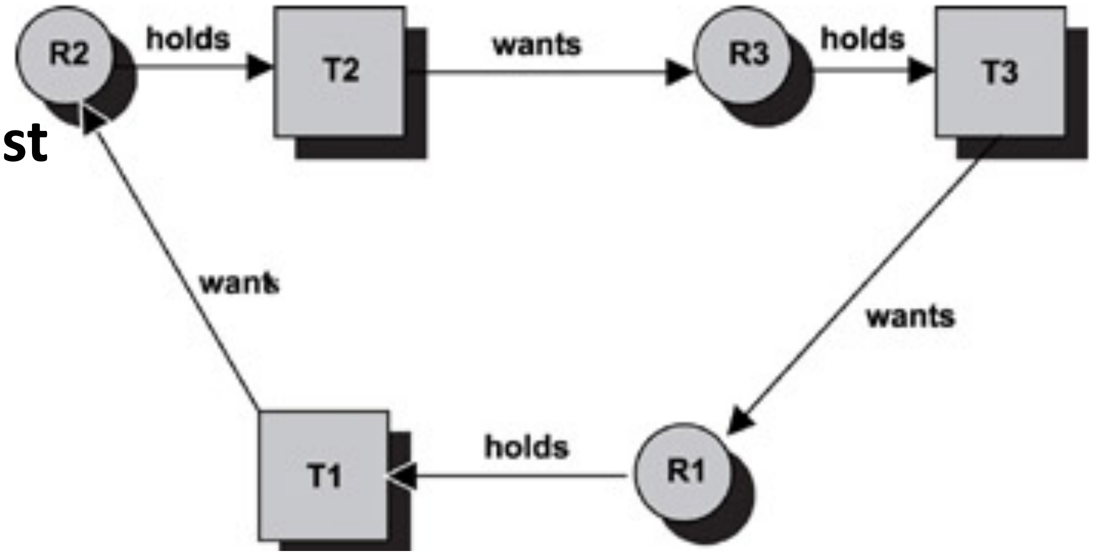
AND OR Resource request Model: Any combination of AND and OR.

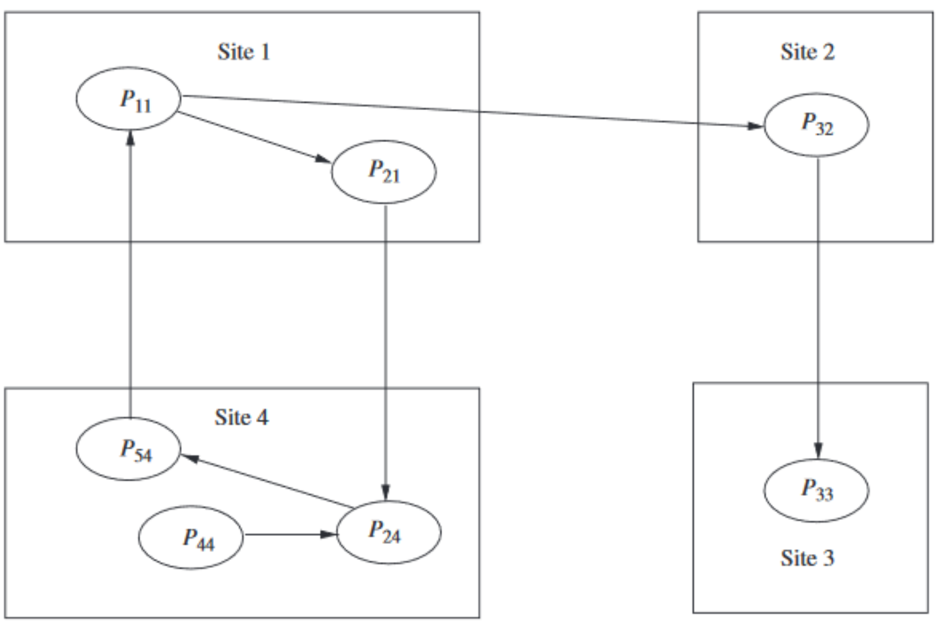
Example: X AND (Y OR Z)

detecting using WFG does not directly wok but algorithms work on doing repeated OR tests over time (as deadlocks dont go away)

Single Resource Request Model

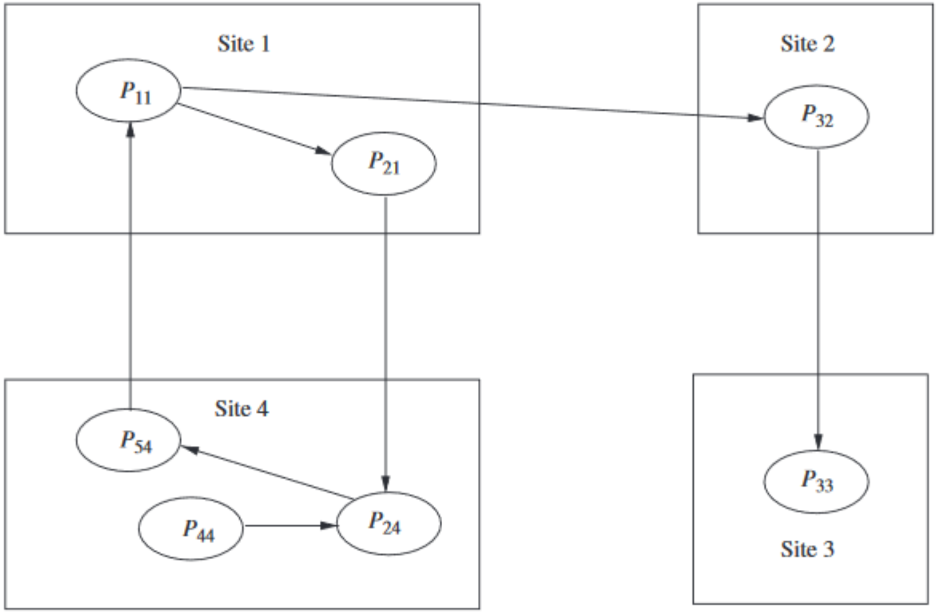
Presence of cycle indicates deadlock?





AND Resource request Model

If there is a cycle then there is a deadlock? there is a process deadlocked then will it be part of a cycle?

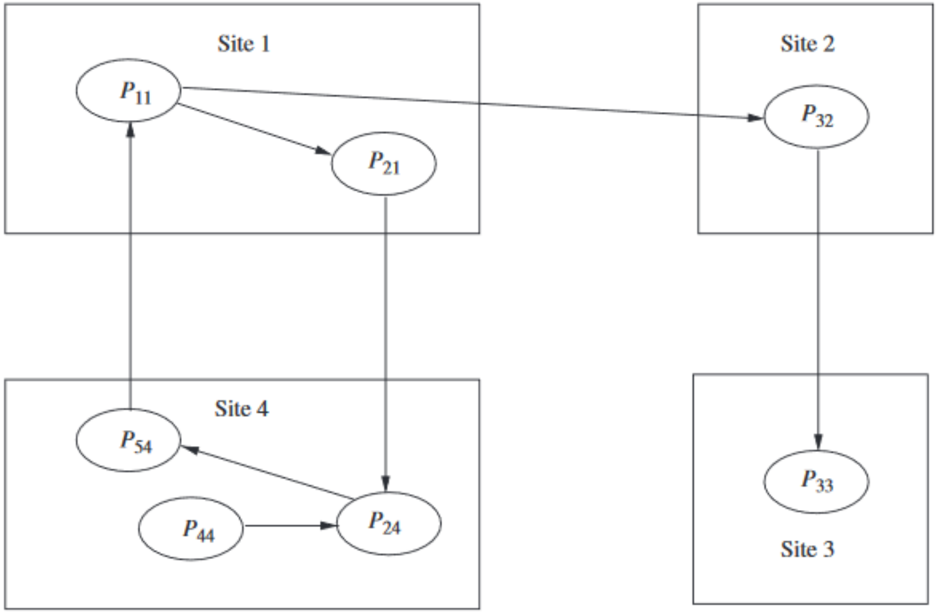


OR Resource request Model

If there is a cycle then there is a deadlock?

A knot(K) consists of a set of nodes such that for every node a in K, all nodes in K and only the nodes in K are reachable from node a.
(SCC with no outgoing edge?)

A knot indicates a deadlock in the OR Model.
Are processes in knot alone deadlocked?



OR Resource request Model

If there is a cycle then there is a deadlock?

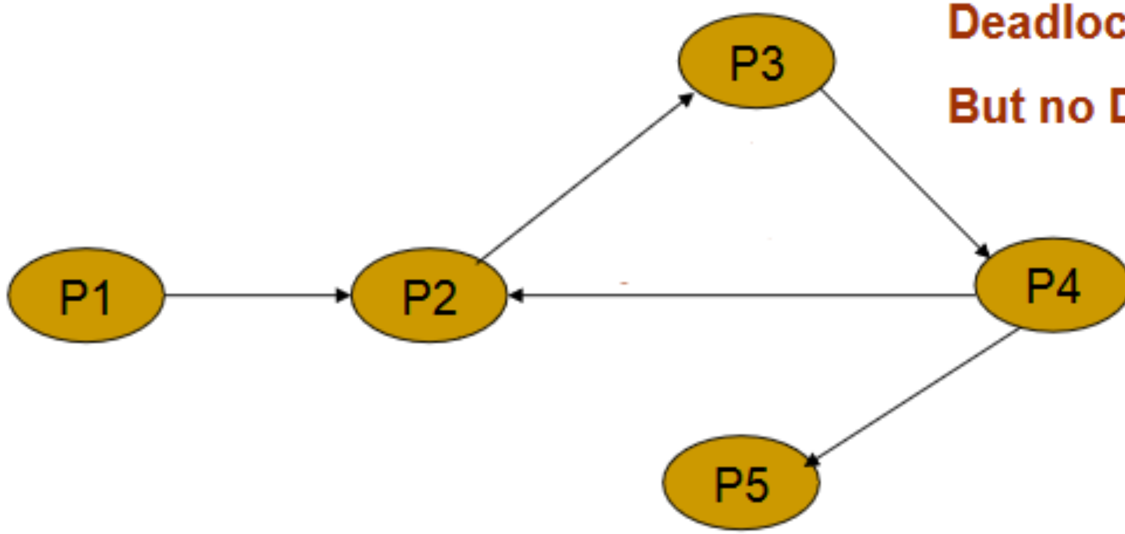
No, P33, then P32 and then P11 will run

A knot(K) consists of a set of nodes such that for every node a in K, all nodes in K and only the nodes in K are reachable from node a. (SCC with no outgoing edge?)

A knot indicates a deadlock in the OR Model.
Are processes in knot alone deadlocked? No..P44 too is

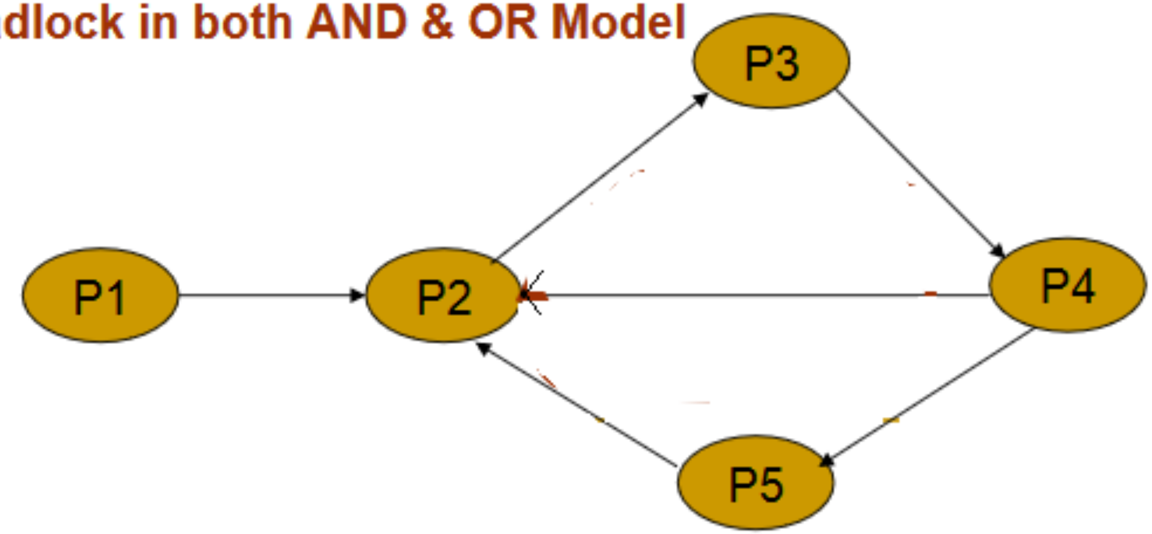
Deadlock in AND Model

But no Deadlock in OR Model



Cycle but no Knot

Deadlock in both AND & OR Model



Cycle & Knot

Deadlock conditions

- The condition for deadlock in a system using the AND condition is the existence of a *cycle*.
- The condition for deadlock in a system using the OR condition is the existence of a *knot*.

Hence a system is deadlocked iff there exists a cycle or knot in WFG