
Buy image at http://yayimages.com/image/fileid/4511200
YAY: Quality Images, Low Prices
IMAGES

Deadlock Detection and Recovery

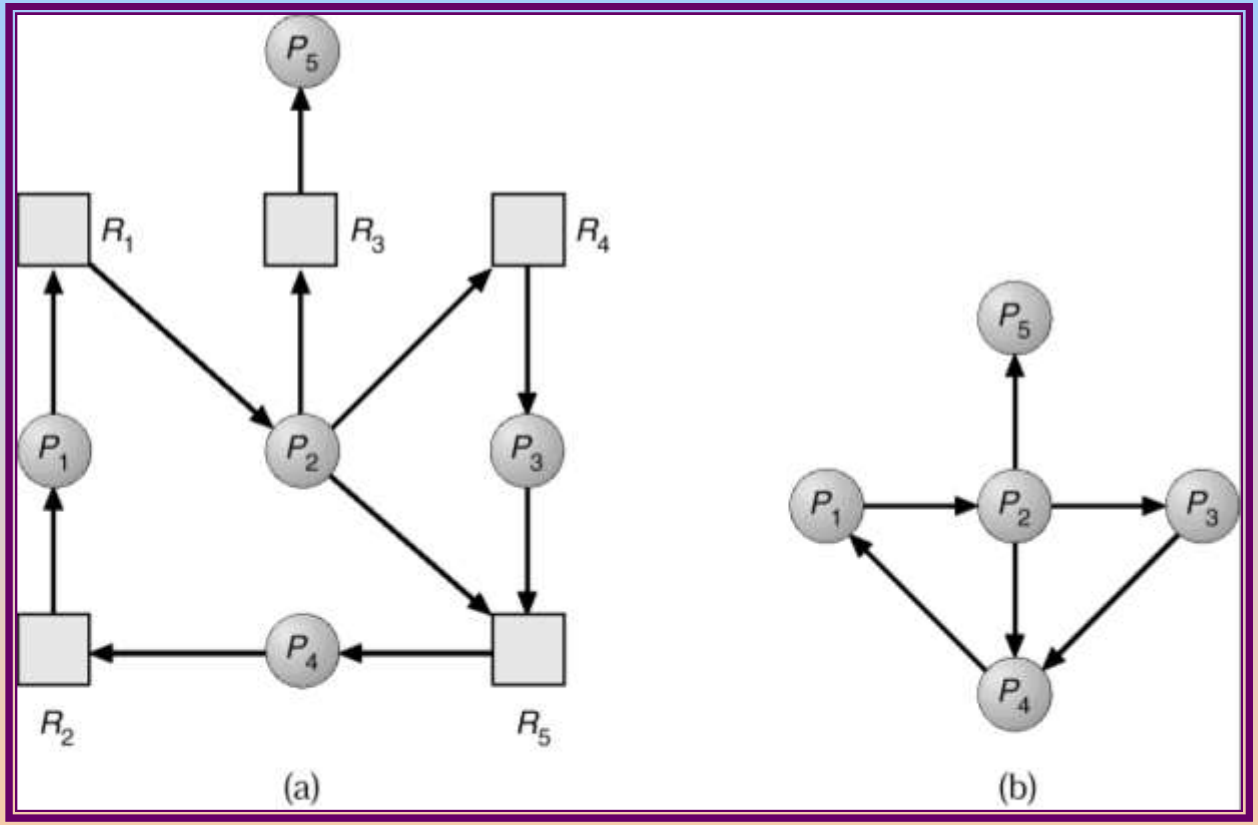
Deadlock Detection

- If a system does not employ either deadlock-prevention or a deadlock avoidance algorithm, then deadlock situation may occur
- In these environments the system must provide
  - An algorithm that examines the state of the system to determine whether a deadlock has occurred
  - An algorithm to recover from a deadlock

Single Instance of Each Resource Type

- Maintain *wait-for* graph
  - Nodes are processes.
  - $P_i \rightarrow P_j$ if P_i is waiting for P_j.
- Periodically invoke an algorithm that searches for a cycle in the graph.
- An algorithm to detect a cycle in a graph requires an order of n^2 operations, where n is the number of vertices in the graph.

Resource-Allocation Graph and Wait-for Graph



Resource-Allocation Graph

Corresponding wait-for graph

SO 2: Determine the deadlock occurrence with several instance

Several Instances of a Resource Type

- *Available:* A vector of length *m* indicates the number of available resources of each type.

- *Allocation:* An *n* x *m* matrix defines the number of resources of each type currently allocated to each process.

- *Request:* An *n* x *m* matrix indicates the current request of each process. If *Request* [*I,j*] = *k*, then process P_i is requesting *k* more instances of resource type. R_j.

Detection Algorithm

1. Let *Work* and *Finish* be vectors of length *m* and *n*, respectively Initialize:
   (a) *Work = Available*
   (b) For *i* = 1,2, …, *n*, if $Allocation_i$ =/= 0, then *Finish*[i] = false;otherwise, *Finish*[i] = *true*.
2. Find an index *i* such that both:
   (a) *Finish*[*i*] == *false*
   (b) $Request_i \leq Work$

   If no such *i* exists, go to step 4.

Detection Algorithm (Cont.)

3. $Work = Work + Allocation_i$
   $Finish[i] = true$
   go to step 2.

4. If $Finish[i]$ == false, for some i, $1 \leq i \leq n$, then the system is in deadlock state. Moreover, if $Finish[i]$ == *false*, then P_i is deadlocked.

Algorithm requires an order of $O(m \times n^2)$ operations to detect whether the system is in deadlocked state.

Example of Detection Algorithm

- Five processes P_0 through P_4; three resource types A (7 instances), *B* (2 instances), and *C* (6 instances).
- Snapshot at time T_0:

	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	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

- Sequence <P_0, P_2, P_3, P_4, P_1> will result in *Finish*[*i*] = true for all *i*.

Example (Cont.)

- P_2 requests an additional instance of type C.

	Request
	A B C
P_0	0 0 0
P_1	2 0 1
P_2	0 0 1
P_3	1 0 0
P_4	0 0 2

- State of system?
  - Deadlock exists, consisting of processes P_1, P_2, P_3, and P_4.

SO 3: Exemplify the ways to recover from deadlock

Recovery from Deadlock: Process Termination

- When deadlock is detected by algorithm then there are two possibilities
- Deal with deadlock manually
- System recover from deadlock automatically
- There are two solutions to break the deadlock
  - Abort one or more process to break the circular wait
  - Preempt some resources from one or more deadlocked processes
- Process Termination

1. Abort all deadlocked processes.

2. Abort one process at a time until the deadlock cycle is eliminated.

Recovery from Deadlock: Process Termination

- In which order should we choose to abort?
  - Priority of the process.
  - How long process has computed, and how much longer to completion.
  - Resources the process has used.
  - Resources process needs to complete.
  - How many processes will need to be terminated.
  - Is process interactive or batch?

Recovery from Deadlock: Resource Preemption

To eliminate deadlocks, we preempt some resources from P and give these resources to other P until deadlock cycle is broken. 3 issues to be considered

- Selecting a victim-determined by the cost factors such as no of resources a deadlock process is holding, amount of time a deadlocked process has consumed during execution

- Rollback – return to some safe state, restart process from that state.

- Starvation – Starvation is where low priority processes get blocked, and high priority process proceeds.

Combined Approach to Deadlock Handling

- Combine the three basic approaches
  - prevention
  - avoidance
  - detection

  allowing the use of the optimal approach for each of resources in the system.

Formative Assessment

- Which of the following graph is used in deadlock detection?
  a. Resource allocation graph
  b. Resource request graph
  c. Wait for graph
  d. Resource allocation graph
- A system is in state if the system can allocate resources to each process in some order and still avoid deadlock.
  a. Safe
  b. Unsafe
  c. Deadlock
  d. Combination of any 2 states