Deadlocks (3)

Dr. Jun Zheng

CSE325 Principles of Operating Systems

10/11/2019

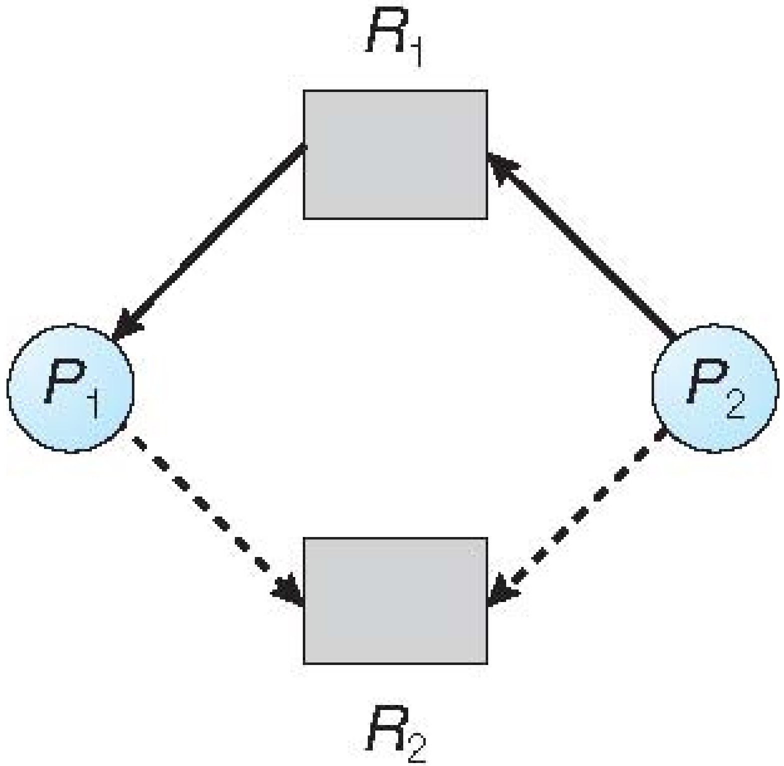
NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Resource-Allocation Graph Scheme

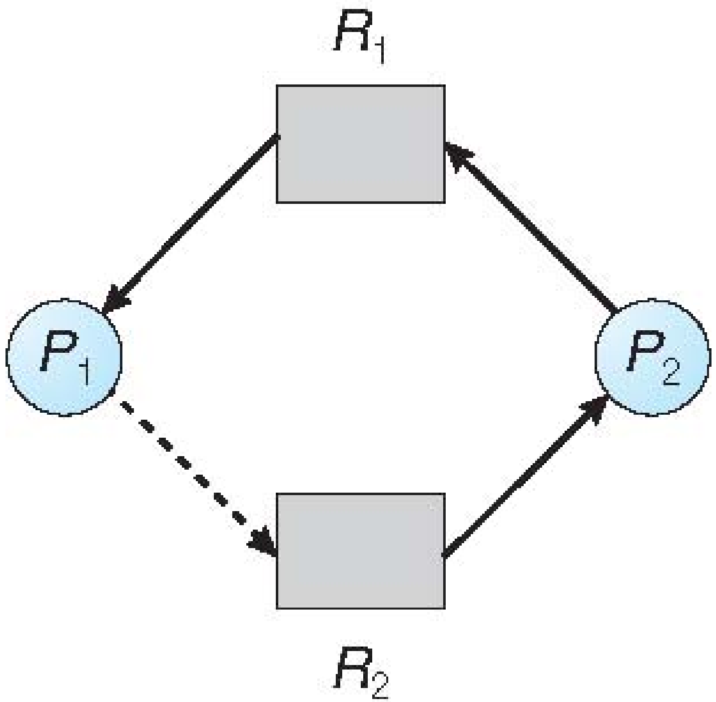
- **Claim edge** $P_i \rightarrow R_j$ indicated that process P_j may request resource R_j; represented by a dashed line
- Claim edge converts to request edge when a process requests a resource
- Request edge converted to an assignment edge when the resource is allocated to the process
- When a resource is released by a process, assignment edge reconverts to a claim edge
- Resources must be claimed *a priori* in the system

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Resource-Allocation Graph



Unsafe State In Resource-Allocation Graph



NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Resource-Allocation Graph Algorithm

- Suppose that process P_i requests a resource R_j
- The request can be granted only if converting the request edge to an assignment edge does not result in the formation of a cycle in the resource allocation graph

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Banker's Algorithm

- Multiple instances
- Each process must a priori claim maximum use
- When a process requests a resource it may have to wait
- When a process gets all its resources it must return them in a finite amount of time

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Data Structures for the Banker's Algorithm

Let n = number of processes, and m = number of resources types.

- **Available**: Vector of length m. If available $[j] = k$, there are k instances of resource type R_j available
- **Max**: n x m matrix. If $Max\ [i,j] = k$, then process P_i may request at most k instances of resource type R_j
- **Allocation**: n x m matrix. If Allocation$[i,j] = k$ then P_i is currently allocated k instances of R_j
- **Need**: n x m matrix. If $Need[i,j] = k$, then P_i may need k more instances of R_j to complete its task
  $Need\ [i,j] = Max[i,j] - Allocation\ [i,j]$

Safety Algorithm

- 1. Let ***Work*** and ***Finish*** be vectors of length *m* and *n*, respectively. Initialize:
  - ***Work = Available***
  - ***Finish* [*i*] = *false* for *i* = 0, 1, …, *n*- 1**
- 2. Find an ***i*** such that both:
  - (a) ***Finish* [*i*] = *false***
  - (b) ***Need**$_i$* $\leq$ ***Work***
  - If no such ***i*** exists, go to step 4
- 3. ***Work = Work + Allocation**$_i$*
  Finish*[*i*] = *true
  go to step 2
- 4. If ***Finish* [*i*] == *true*** for all ***i***, then the system is in a safe state

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Resource-Request Algorithm for Process P_i

- ❿ $\boldsymbol{Request_i}$ = request vector for process $\boldsymbol{P_i}$. If $\boldsymbol{Request_i}$ **[*j*] = *k*** then process $\boldsymbol{P_i}$ wants ***k*** instances of resource type $\boldsymbol{R_j}$
  - 1. If $\boldsymbol{Request_i} \leq \boldsymbol{Need_i}$ go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim
  - 2. If $\boldsymbol{Request_i} \leq \boldsymbol{Available}$, go to step 3. Otherwise $\boldsymbol{P_i}$ must wait, since resources are not available
  - 3. Pretend to allocate requested resources to $\boldsymbol{P_i}$ by modifying the state as follows:
    - $\boldsymbol{Available = Available - Request_i;}$
    - $\boldsymbol{Allocation_i = Allocation_i + Request_i;}$
    - $\boldsymbol{Need_i = Need_i - Request_i;}$
  - If safe $\Rightarrow$ the resources are allocated to $\boldsymbol{P_i}$
  - If unsafe $\Rightarrow$ $\boldsymbol{P_i}$ must wait, and the old resource-allocation state is restored

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Example of Banker's Algorithm

5 processes P_0 through P_4;

3 resource types:

A (10 instances), B (5instances), and C (7 instances)

Snapshot at time T_0:

	Allocation	*Max*	*Available*
	$A\ B\ C$	$A\ B\ C$	$A\ B\ C$
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Example (Cont.)

The content of the matrix ***Need*** is defined to be ***Max* – *Allocation***

	Need
	A B C
P_0	7 4 3
P_1	1 2 2
P_2	6 0 0
P_3	0 1 1
P_4	4 3 1

The system is in a safe state since the sequence < P_1, P_3, P_4, P_2, P_0> satisfies safety criteria

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Example: P_1 Request (1,0,2)

Check that Request ≤ Available (that is, (1,0,2) ≤ (3,3,2) ⇒ true

	Allocation	*Need*	*Available*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

Executing safety algorithm shows that sequence < P_1, P_3, P_4, P_0, P_2> satisfies safety requirement

Can request for (3,3,0) by P_4 be granted?

Can request for (0,2,0) by P_0 be granted?