Department of Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY
UNIVERSITY OF LUCKNOW
LUCKNOW

CS-501

Dr. Zeeshan Ali Siddiqui
Assistant Professor
Deptt. of C.S.E.

METHODS FOR HANDLING DEADLOCKS
(Part-5)
Banker's Algorithm Example

Methods for Handling Deadlocks

- *Deadlock Prevention*
- ***Deadlock Avoidance***
- *Deadlock Detection*
- *Ignore the problem*

DEADLOCK AVOIDANCE
Continue…

AVOIDANCE ALGORITHMS

Avoidance algorithms

- Single instance of a resource type:
  - *Use a resource-allocation graph*

- Multiple instances of a resource type:
  - *Use the banker's algorithm*

BANKER'S ALGORITHM
Example

Banker's Algorithm: Example

- Let a system with
  - *Processes*: 5 Processes (P0 through P4)
  - *Resource* types: 3 (A, B, and C)
    - Resource type A has 10 instances,
    - Resource type B has 5 instances,
    - Resource type C has 7 instances.

- Suppose that, at time T0, the following snapshot of the system has been taken:

	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	

Banker's Algorithm: Solution 1/5

	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	

- Matrix, *Need = 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

Banker's Algorithm: Solution 2/5

	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	

	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

- Now, by applying *Safety Algorithm*:

- Step 1:
  - Work: = Available: =332,
  - Finish[i]= false, where i=0 to 4

Banker's Algorithm: Solution 3/5

	Allocation	Max	Available	Need
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2		1 2 2
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	2 2 2		0 1 1
P_4	0 0 2	4 3 3		4 3 1

- Step 2:

work

332

1. for i=0, Finish[0]=false,
   Needi <=work;
   Need0 <=work;
   743<=332 ->**False**

Banker's Algorithm: Solution 3/5

- Step 2:

	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	

	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

2. for i=1, Finish[1]=false,
   Needi <=work;
   Need1 <=work;
   122<=332 ->**True**

   if **True** Then:
   Work = Work + Allocation1 ;
   = 332+200=532;

   Finish[1] = **True**;

work

332

work

532

Banker's Algorithm: Solution 3/5

- Step 2:

	Allocation	Max	Available	Need
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2		1 2 2
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	2 2 2		0 1 1
P_4	0 0 2	4 3 3		4 3 1

3. for i=2, Finish[2]=false,
   Needi <=work;
   Need2 <=work;
   600<=532->False

work

532

Banker's Algorithm: Solution 3/5

- Step 2:

	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	

	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

4. for i=3, Finish[3]=false,
   Needi <=work;
   Need3 <=work;
   011<=532 ->**True**

   if **True** Then:
   Work = Work + Allocation3 ;
   =532+211=743;

   Finish[3] = **True**;

work
532

work
743

Banker's Algorithm: Solution 4/5

	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	

	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

5. for i=4, Finish[4]=false,
   Needi <=work;
   Need4 <=work;
   431<=743 ->**True**

   if **True** Then:
   Work = Work + Allocation4 ;
   =743+002=745;

   Finish[4] = **True**;

work
743

work
745

Banker's Algorithm: Solution 4/5

	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	

	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

6. for i=0, Finish[0]=false,
   Needi <=work;
   Need0 <=work;
   743<=745 ->**True**

   if **True** Then:
   Work = Work + Allocation0 ;
   =745+010=755;

   Finish[0] = **True**;

work
745

work
755

Banker's Algorithm: Solution 5/5

	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	

	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

7. for i=2, Finish[2]=false,
Needi <=work;
Need2 <=work;
600<=755 ->**True**

if **True** Then:
Work = Work + Allocation2 ;
=755+302=10 5 7;

Finish[2] = **True**;

Safe Sequence:
<P1, P3, P4, P0, P2>

work
755

work
10 5 7

Banker's Algorithm: Homework Q1

- Process P1 requests one *additional* instance of resource type A, and two instances of resource type C, so Request1 = (1, 0, 2).

- By Applying Resource Allocation Algorithm:

  Request1 ≤ Available

	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	

- That is, that (1, 0, 2) ≤ (3, 3, 2), which is true.

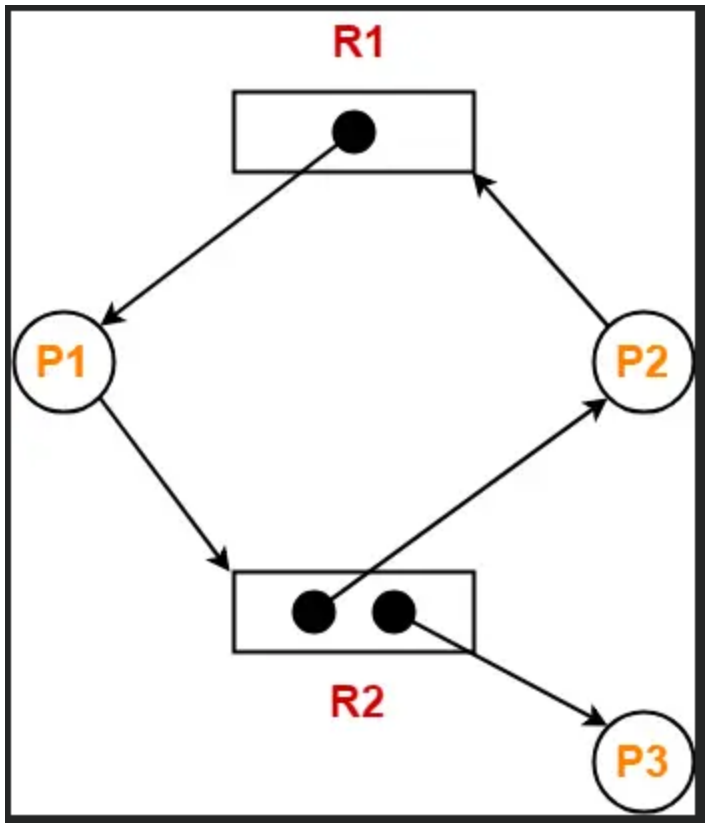
- We then *pretend* that this request has been fulfilled, and we arrive at the following new state:

- *Find the safe sequence.*

	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	

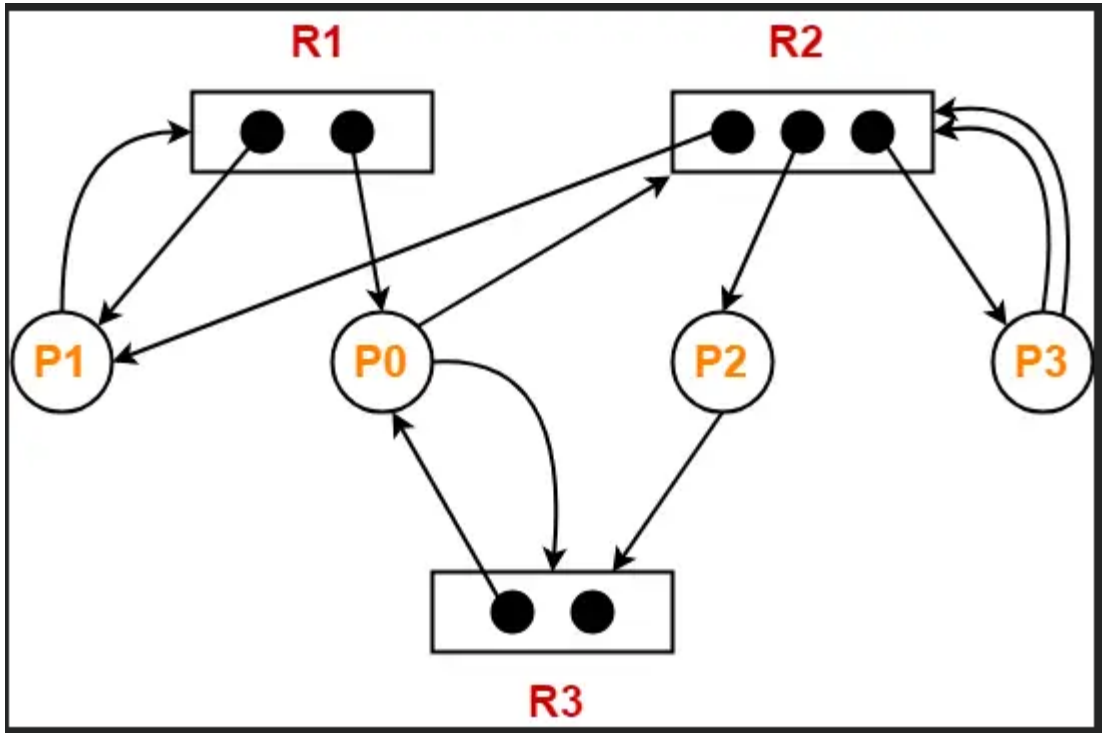
Banker’s Algorithm: Homework Q2

- Take a look at the figure's resource allocation graph. Check for deadlocks in the system; if not, find a safe sequence.



Banker’s Algorithm: Homework Q3

- Take a look at the figure's resource allocation graph. Check for deadlocks in the system; if not, find a safe sequence.



References

1. Silberschatz, Galvin and Gagne, “Operating Systems Concepts”, Wiley.

2. William Stallings, “Operating Systems: Internals and Design Principles”, 6th Edition, Pearson Education.

3. D M Dhamdhere, “Operating Systems: A Concept based Approach”, 2nd Edition, TMH.

Thank You.