OPERATING SYSTEM “DEADLOCK”

Kanika
Assistant Professor
CSE, MAIT
Maharaja Agrasen University Baddi

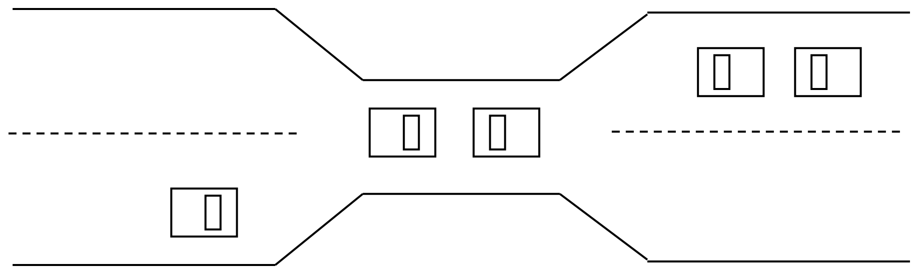
Introduction

- Deadlock
- Deadlock Prevention
- Deadlock Avoidance
- Deadlock Detection & Recovery

DEADLOCK

If two or more processes are waiting on happening of some event, which never happens, then we say these processes are involved in deadlock then that state is called Deadlock.

Bridge Crossing Example



- Traffic only in one direction.
- Each section of a bridge can be viewed as a resource.
- If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback).
- Several cars may have to be backed up if a deadlock occurs.

DEADLOCK NECESSARY CONDITIONS

ALL of these four **must** happen simultaneously for a deadlock to occur:

Mutual exclusion

One or more than one resource must be held by a process in a non-sharable (exclusive) mode.

Hold and Wait

A process holds a resource while waiting for another resource.

No Preemption

There is only voluntary release of a resource - nobody else can make a process give up a resource.

Circular Wait

Process A waits for Process B waits for Process C .... waits for Process A.

RESOURCE ALLOCATION GRAPH

A mathematical way to determine if a deadlock has, or may occur.

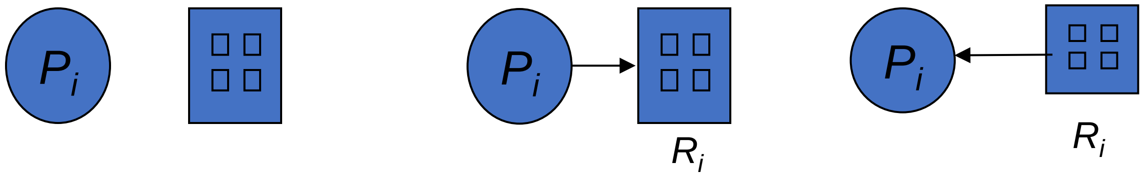
G = (V, E) The graph contains nodes and edges.

V Nodes consist of processes = { P1, P2, P3, ...} and resource types { R1, R2, ...}

E Edges are (Pi, Rj) or (Ri, Pj)

An arrow from the **process** to **resource** indicates the process is **requesting** the resource. An arrow from **resource** to **process** shows an instance of the resource has been **allocated** to the process.

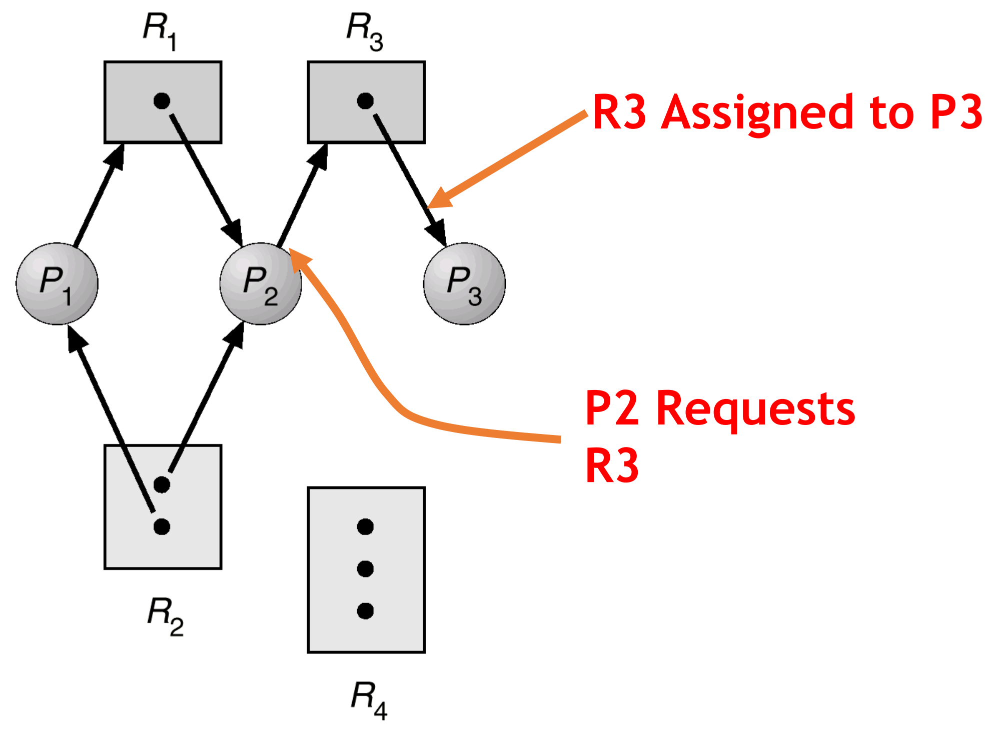
Process is a circle, resource type is square; dots represent number of instances of resource in type. Request points to square, assignment comes from dot.



RESOURCE ALLOCATION GRAPH

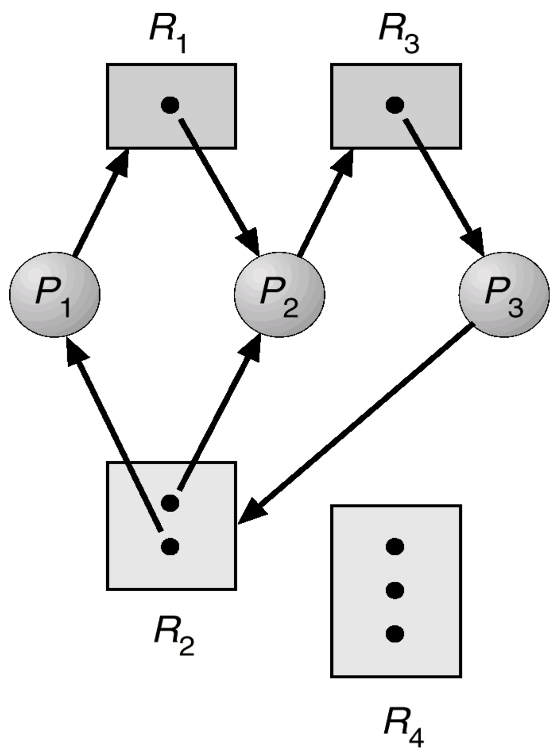
- If the graph contains no cycles, then no process is deadlocked.
- If there is a cycle, then:
  a) If resource types have multiple instances, then deadlock MAY exist.
  b) If each resource type has 1 instance, then deadlock has occurred.

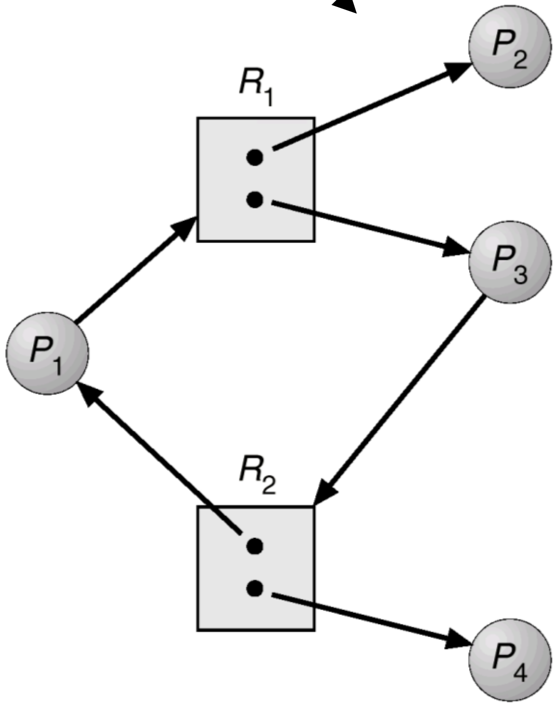
Resource allocation graph



Resource allocation graph with a deadlock.

Resource allocation graph with a cycle but no deadlock.





HOW TO HANDLE DEADLOCKS – GENERAL STRATEGIES

There are three methods:

1) Deadlock Ignorance (Ostrich Method): Ignore Deadlocks

2) Deadlock Prevention: Prevent any one of the 4 conditions from happening.

3) Deadlock Avoidance: Banker's Algorithm Allow deadlock to happen. This requires using both.

4) Deadlock Detection and Recovery: Detect and Recover the deadlock.

Deadlock Prevention

Do not allow one of the four conditions to occur.

Mutual exclusion:

a) Automatically holds for printers and other non-sharables.

b) Shared entities don't need mutual exclusion.

c) Prevention not possible, since some devices are intrinsically non-sharable.

Hold and wait:

a) Collect all resources before execution.

b) A particular resource can only be requested when no others are being held. A sequence of resources is always collected beginning with the same one.

c) Utilization is low, starvation possible.

No preemption:

a) Release any resource already being held if the process can't get an additional resource.

b) Allow preemption - if a needed resource is held by another process, which is also waiting on some resource, steal it. Otherwise wait.

Circular wait:

a) Each of these prevention techniques may cause a decrease in utilization and/or resources. For this reason, prevention isn't necessarily the best technique.

b) In general way we can easily implement the Prevention.

Deadlock Avoidance

Prior knowledge of resource request is needed, to determine if we are entering an "unsafe" state.

Possible states are:

Deadlock	No forward progress can be made.
Unsafe state	A state that **may** allow deadlock.
Safe state	A state is safe if a sequence of processes exist such that there are enough resources for the first to finish, and as each finishes and releases its resources there are enough for the next to finish.

The rule is simple: If a request allocation would cause an unsafe state, do not honor that request.

NOTE: All deadlocks are unsafe, but all unsafe are NOT deadlocks.

Deadlock Avoidance

Let's assume a very simple model: each process declares its maximum needs. In this case, algorithms exist that will ensure that no unsafe state is reached.

Do these examples:

Consider a system with: five processes, P0 → P4, three resource types, A, B, C.

Type A has 10 instances, B has 5 instances, C has 7 instances.

At time T0 the following snapshot of the system is taken.

Max Needs = allocated + can-be-requested

Is the system in a safe state?

	←	Aloc	→		←	Req	→		←	Avail	→
	A	B	C		A	B	C		A	B	C
P0	0	1	0		7	4	3		3	3	2
P1	2	0	0		0	2	0				
P2	3	0	2		6	0	0				
P3	2	1	1		0	1	1				
P4	0	0	2		4	3	1				

Deadlock Avoidance

Let's assume a very simple model: each process declares its maximum needs. In this case, algorithms exist that will ensure that no unsafe state is reached.

Do these examples:

Consider a system with: five processes, P0 → P4, three resource types, A, B, C.

Type A has 10 instances, B has 5 instances, C has 7 instances.

At time T0 the following snapshot of the system is taken.

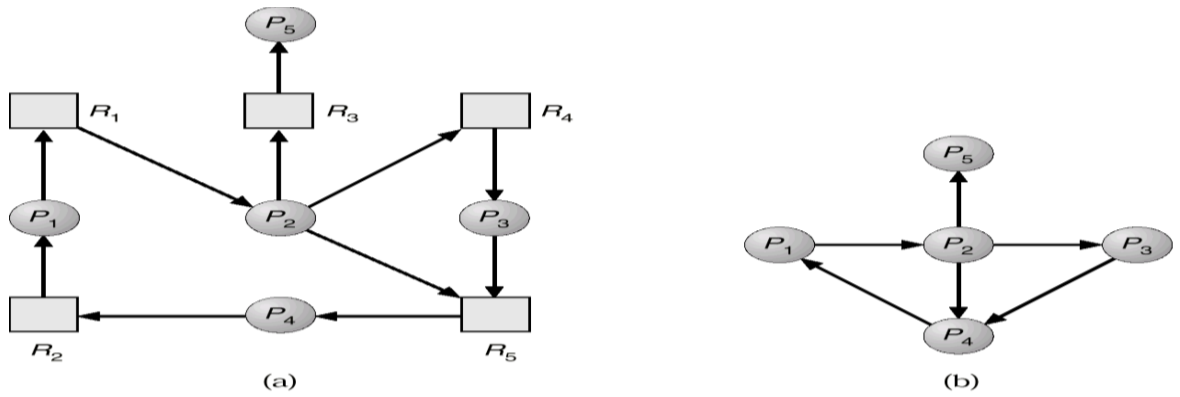
Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P1	0	1	0	7	5	3	3	3	2
P2	2	0	0	3	2	2			
P3	3	0	2	9	0	2			
P4	2	1	1	2	2	2			
P5	0	0	2	4	3	3			

Deadlock Detection

Need an algorithm that determines if deadlock occurred.

SINGLE INSTANCE OF A RESOURCE TYPE

- Wait-for graph == remove the resources from the usual graph and collapse edges.
- An edge from p(j) to p(i) implies that p(j) is waiting for p(i) to release.



Deadlock Recovery

So, the deadlock has occurred. Now, how do we get the resources back and gain forward progress?

PROCESS TERMINATION:

- Could delete all the processes in the deadlock -- this is expensive.
- Delete one at a time until deadlock is broken (time consuming).
- Select who to terminate based on priority, time executed, time to completion, needs for completion, or depth of rollback
- In general, it's easier to preempt the resource, than to terminate the process.

Deadlock Recovery

RESOURCE PREEMPTION:

- Select a victim - which process and which resource to preempt.
- Rollback to previously defined "safe" state.
- Prevent one process from always being the one preempted (starvation).

THANK YOU