Review

Chapter 3. Deadlock

- Deadlock Strategies
  - Just ignore
  - Detection and Recover
    - Deadlock Detection Algorithm
    - Recover

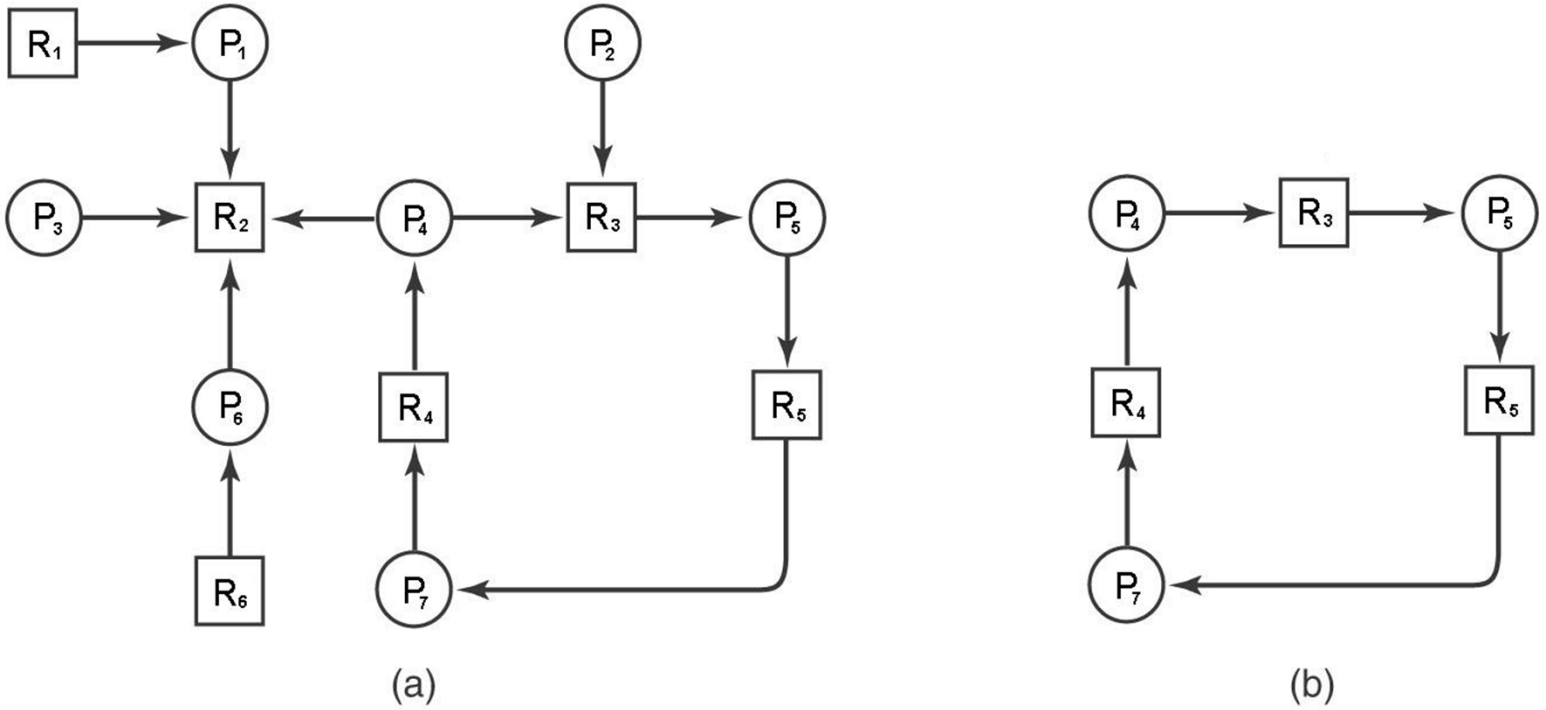
Deadlock Condition

A deadlock situation can arise if and only if the following four conditions hold simultaneously in a system. (Coffman et al.)

1. Mutual exclusion
2. Hold and wait
3. No preemption
4. Circular wait

Deadlock Detection and Recovery
(Detection with one resource of each type)

A cycle – means *deadlock*



Deadlock Detection and Recovery
(Detection with Multiple resources of each type)

We need matrices for deadlock detection algorithm.

- Existing resource matrix – At present, the number of resources per each type.
- Available resource matrix – At present, the number of resources per each type available, with some of the resources are assigned to processes (not available).
- Current allocation matrix – At present, which resources are currently held by processes
- Request matrix – At present, how many resources are needed for processes to finish their job.

Deadlock Detection and Recovery

(Detection with Multiple resource of each type)

E Resources in existence $(E_1, E_2, E_3, \ldots, E_m)$

A Resources available $(A_1, A_2, A_3, \ldots, A_m)$

C Current allocation matrix

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm} \end{bmatrix}$$

Row n is current allocation to process n

R Request matrix

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1m} \\ R_{21} & R_{22} & R_{23} & \cdots & R_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & R_{n3} & \cdots & R_{nm} \end{bmatrix}$$

Row 2 is what process 2 needs

We have the following equation from the four matrix structures.

$$\sum_{i=1}^{n} C_{i,j} + A_j = E_j$$

Deadlock Detection and Recovery
(Detection with Multiple resource of each type)

- The deadlock detection algorithm is based on comparing vectors.
- Let's define the relation A ≤ B between two vector A and B means that each element of A is less than or equal to the corresponding element of B.

Ex)

A = (1, 2, 0 2), B = (1, 3, 0 2): Is A ≤ B true? Yes

A = (1, 2, 0, 2), B = (2, 0, 0, 0): Is A ≤ B true? No

Deadlock Detection and Recovery

(Detection with Multiple resource of each type)

A Deadlock detection algorithm

1. All processes start with unmarked.
2. Look for an unmarked process P_i, for which the i^{th} row of R is less than or equal to A.
3. If such a process is found, add the i^{th} row of C to A, mark the process and go back to step 1.
4. If no such process exists, the algorithm terminates.

Preview

- Deadlock Avoidance
  - Safe and Unsafe state
  - Banker's Algorithm
- Deadlock Prevention
  - Attack Mutual Exclusion
  - Attack Hold and Wait
  - Attack No-Preemption
  - Attack Circular Wait

Deadlock Avoidance

- The system must be able to decide whether granting a resource is safe or not and only make allocation when it is safe.
- Is there any algorithm that can always avoid deadlock by making the right choice all the time?
  - Yes, when certain information is available in advance.

Deadlock Avoidance
(Safe and Unsafe State)

A state is said to be a safe state

- If it is not deadlocked and
- there is some scheduling order in which every process can run to completion even if all of them suddenly request their maximum number of resources immediately.

Deadlock Avoidance (Safe and Unsafe State)

The State in (a) is Safe

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3

(a)

	Has	Max
A	3	9
B	4	4
C	2	7

Free: 1

(b)

	Has	Max
A	3	9
B	0	–
C	2	7

Free: 5

(c)

	Has	Max
A	3	9
B	0	–
C	7	7

Free: 0

(d)

	Has	Max
A	3	9
B	0	–
C	0	–

Free: 7

(e)

Deadlock Avoidance (Safe and Unsafe State)

The State in (a) is Safe & The State in (b) is Unsafe

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3

(a)

	Has	Max
A	4	9
B	2	4
C	2	7

Free: 2

(b)

	Has	Max
A	4	9
B	4	4
C	2	7

Free: 0

(c)

	Has	Max
A	4	9
B	—	—
C	2	7

Free: 4

(d)

Banker's Algorithm for a Single Resource (Dijkstra)

- Modeled on the way a small-town banker might need to deal with a group of customers.
- What banker's algorithm does is check to see whether granting the request leads to an unsafe state or not.
- If granting the request leads to an unsafe state, the request is denied.

Banker's Algorithm for a Single Resource (Dijkstra)

Ex)

- There are four customers A, B, C, D, each of whom has been granted a certain amount of credit units.

	Has	Max
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10

(a)

	Has	Max
A	1	6
B	1	5
C	2	4
D	4	7

Free: 2

(b)

	Has	Max
A	1	6
B	2	5
C	2	4
D	4	7

Free: 1

(c)

Three resource allocation states: (a) Safe (b) Safe (c) Unsafe

- The banker knows that not all customers will need their maximum credit immediately, so banker has reserved only 10 units rather than 22.

Banker's Algorithm for Multiple Resource

Using three matrices for checking a safe state

- Available resource matrix **(A)** – At present, how many number of resources per each type are available (Note that some resources are assigned to processes and they are not available).
- Request matrix (R) – At present, how many resources (at most) are needed for processes to finish their jobs.
- Current allocation matrix (C) – At present, how many resources are currently held by processes.

Banker's Algorithm for Multiple Resource

The algorithm for checking if a state is safe or not

1. Look for a row of vector R, whose unmet resource needs are all smaller than or equal to A. If no such row exists, the system might have a deadlock later as some processes might never run to completion.
2. Assume the process (of the row) requests all the resources it needs and finishes. Mark that process as terminated and add all its resources to the A vector.
3. Repeat step 1 and 2 until either all processes are terminated (safe state) or until a deadlock occurs (unsafe).

Banker's Algorithm for Multiple Resource

E – Existing resources, P – Possessed resources, Available resources

Process	Tape drives	Plotters	Scanners	CD ROMs
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Scanners	CD ROMs
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still needed

E = (6342)
P = (5322)
A = (1020)

Banker's Algorithm for Multiple Resource

- E = (6, 3, 4, 2)
- A = (1, 0, 2, 0)

$$C = \begin{bmatrix} 3011 \\ 0100 \\ 1110 \\ 1101 \\ 0000 \end{bmatrix}, R = \begin{bmatrix} 1100 \\ 0112 \\ 3100 \\ 0010 \\ 2110 \end{bmatrix}$$

Deadlock Prevention

- Actually, deadlock avoidance is essentially impossible, since it requires information about future requests which is not known.
- Deadlock prevention is a method to attack one of four deadlock condition.
  1. Mutual exclusion
  2. Hold and wait
  3. No preemption
  4. Circular wait

Deadlock Prevention
(Attacking Mutual Exclusion)

Attacking Mutual Exclusion

- The mutual-exclusion condition must hold for non-preemptive resources such as printer, CD writer.
- But preemptive resources do not require mutual exclusion such as read only file.

Deadlock Prevention

(Attacking the hold and wait)

Attacking the hold and wait

Approach 1)

- Each process requests all resources before starting execution.
- If everything is available, all resources (requested by the process) are located and finish its job.
- If some resources are not available, no resources are allocated to the process.

Deadlock Prevention

(Attacking the hold and wait)

Problem with Approach 1)

- Many processes do not know how many resources they will need before start execution.
- If possible, Banker's algorithm can be applied.
- Resources cannot be used optimally

Ex)

- P_1 running for its completion, holding resources R_1, R_2, and R_3.
- P_1 currently using R_1, but R_2 and R_3 are idle.
- P_2 needs only R_2 to finish its job, but R_2 is held by P_1.

Deadlock Prevention

(Attacking the hold and wait)

Attacking the hold and wait

Approach 2)

- Allows a process to request resources only when the process has none.
- To get a new resource, first, release all the resources currently holds and request all at time same time.

Disadvantage with Approach 2

- Starvation is possible – a process that needs several popular resources may have to wait indefinitely since at least one of the resources that it needs is always allocated to some other process.

Deadlock Prevention
(Attacking No Preemption)

Attacking No Preemption

Approach 1)

- If a process holding some resources requests another resources that cannot be immediately allocated it, then all resources currently being hold are preempted.
- Preempted resources are intentionally released and enter the available resources list.
- The process will be restarted only when it can regain its old resources as well as the new ones that it is requesting

Deadlock Prevention

(Attacking No Preemption)

Attacking No Preemption

Approach 2)

- If a process requests some resources, first checks whether they are available or not
- If they are, allocate them to the process.
- If they are not available, check whether they are allocated to some other process that is waiting for additional resources.
- If so, preempt the desired resources from the waiting process and allocate them to the requesting process.
- If the resources are not either available or held by waiting process, the requesting process must wait.

Deadlock Prevention
(Attacking the Circular Wait Condition)

Approach 1)

- A process can hold only one resource.
- If a process needs another resource, the process needs to release the first one.
- But sometimes this approach is not acceptable

Approach 2)

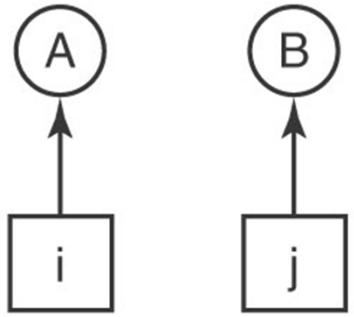
- Global number is provided to each resource
- A process can request resources whenever they want, but all requests must made in numerical order.

Deadlock Prevention

(Attacking the Circular Wait Condition)

1. Imagesetter
2. Scanner
3. Plotter
4. Tape drive
5. CD Rom drive

(a)



(b)

- Only deadlock can occur if the process A makes a request of resource j and the process B makes a request of resource i. but it can never happen since either i > j or i < j.