CISC 3320
C24d Deadlock Detection and Recovery

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Acknowledgement

- These slides are a revision of the slides provided by the authors of the textbook via the publisher of the textbook

Outline

- Deadlock Detection
- Recovery from Deadlock

Deadlock Detection and Recovery

- Allow system to enter deadlock state
- When detecting deadlock, recover from it
  - Detection algorithm
  - Recovery scheme

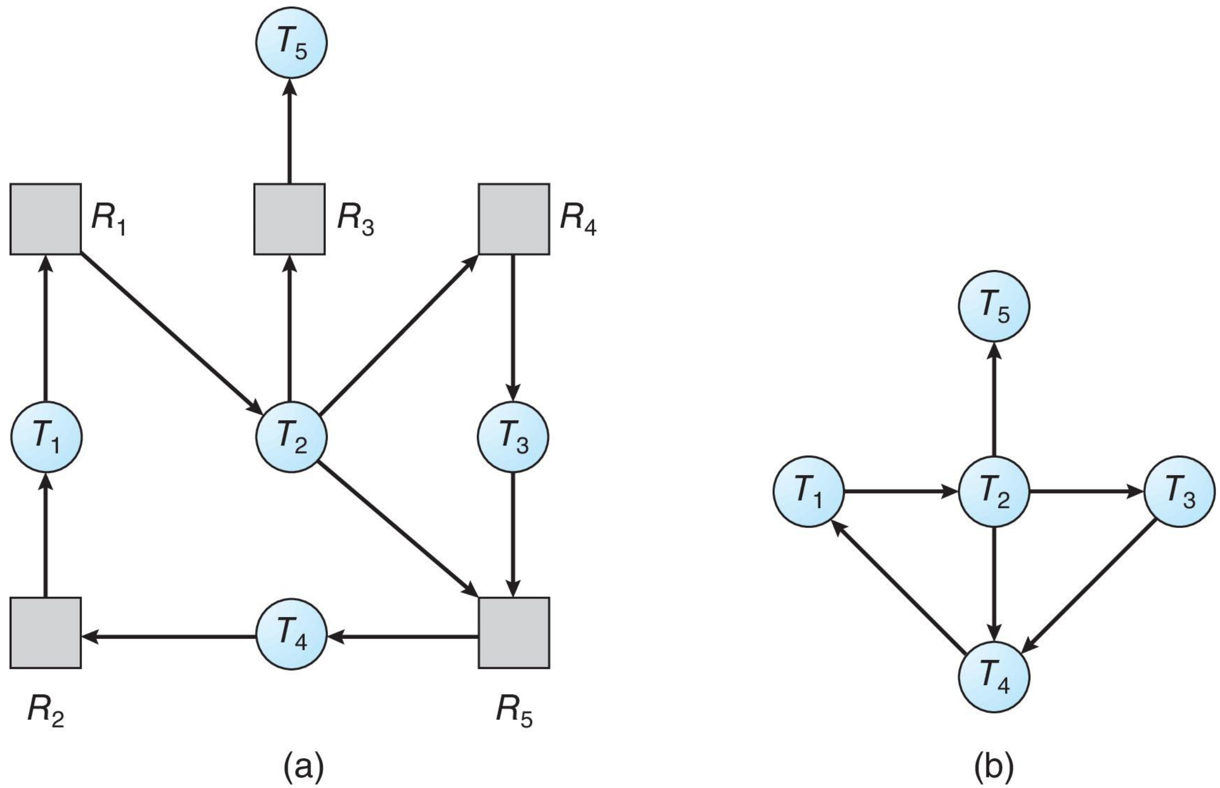
Two Cases

- Single instance of each resource type
- Multiple instances of a resource type

Data Structure for 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. If there is a cycle, there exists a deadlock
- 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

Data Structure for Multiple Instances of a Resource Type

- ***Available***: *A* vector of length $\boldsymbol{m}$ indicates the number of available resources of each type
- ***Allocation***: *An* $\boldsymbol{n \times m}$ matrix defines the number of resources of each type currently allocated to each process
- ***Request***: *An* $\boldsymbol{n \times m}$ 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 $Allocation_i \neq 0$, 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

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

Time Complexity

- Algorithm requires an order of $O(m \times n^2)$ operations to detect whether the system is in deadlocked state
  - Where m: number of resource types, n: number of processes

Detection Algorithm: Example

- 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_1, P_4> will result in ***Finish[i] = true*** for all ***i***

- P_2 requests an additional instance of type C

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

- State of system?
  - Can reclaim resources held by process P_0, but insufficient resources to fulfill other processes; requests
  - Deadlock exists, consisting of processes P_1, P_2, P_3, and P_4

Use Detection Algorithm

- When, and how often, to invoke depends on:
  - How often a deadlock is likely to occur?
  - How many processes will need to be rolled back?
    - one for each disjoint cycle
- If detection algorithm is invoked arbitrarily, there may be many cycles in the resource graph and so we would not be able to tell which of the many deadlocked processes "caused" the deadlock

Questions?

- Detecting deadlock
  - One instance per resource type
    - Resource allocation graph
  - Multiple instance per resource type
    - Similar to the Banker's algorithm

Recovery from Deadlock

- Process termination
- Resource preemption

Process Termination

- Abort all deadlocked processes
- Abort one process at a time until the deadlock cycle is eliminated
- In which order should we choose to abort?
    1. Priority of the process
    2. How long process has computed, and how much longer to completion
    3. Resources the process has used
    4. Resources process needs to complete
    5. How many processes will need to be terminated
    6. Is process interactive or batch?

Recovery from Deadlock: Resource Preemption

- **Selecting a victim** – minimize cost
- **Rollback** – return to some safe state, restart process for that state
- **Starvation** – same process may always be picked as victim, include number of rollback in cost factor

Questions?

- Process termination
- Resource preemption