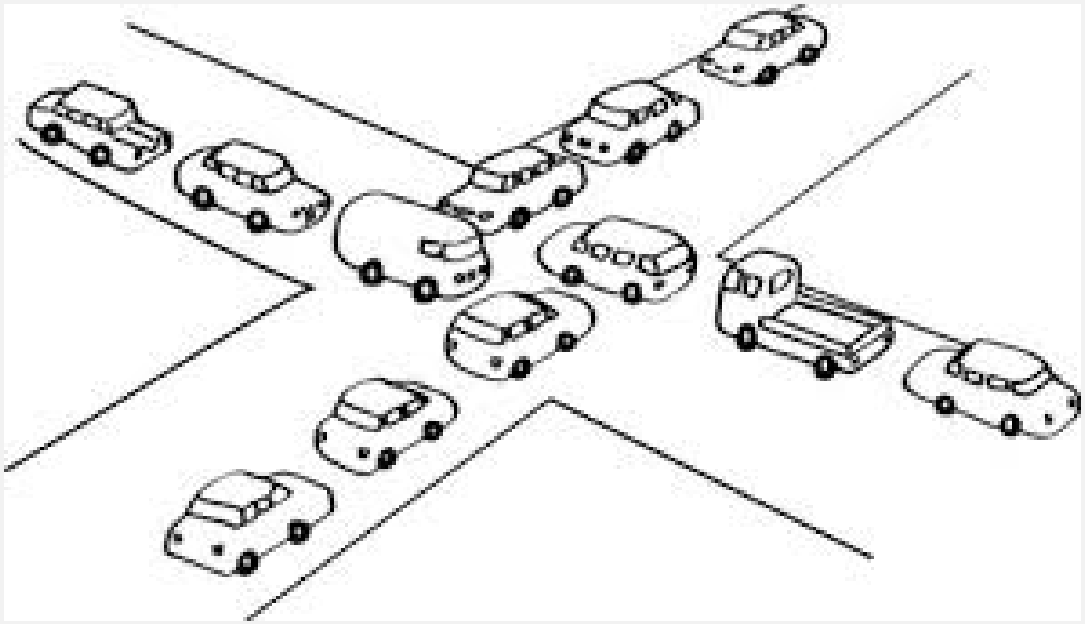
Deadlocks



Learning Objectives

- Understand what is a deadlock and what causes deadlocks?
- How to deal with a deadlock: prevention, avoidance, detection and recovery
- Understand the Banker's Algorithm for deadlock avoidance.

Overview

Deadlock is a state in which each process is waiting fron another process to take action, such as sending a message or releasing a lock. Thus no process is able to make any progress.

Necessary Conditions for a Deadlock

Coffman conditions: A deadlock situation on a resource can arise if and only if all of the following conditions hold simultaneously in a system:

- Mutual exclusion: The resources involved must be unshareable.
- Hold and wait: A process is currently holding at least one resource and requesting additional resources which are being held by other processes.
- No preemption: A resource can be released only voluntarily by the process holding it.
- Circular wait: For a set of processes: $P_i, 1 \leq n$, P_1 is waiting for P_2, which is waiting for P_3 and so on up to P_n,which is waiting for P_1.

Dealing with Deadlocks

Deadlock is a global condition. An individual program generally cannot detect a deadlock, which implies that the deadlocks have to be handled by the operating system. The four basic approaches are:

- Prevention.
- Avoidance.
- Detection and Recovery.
- Manual intervention by the operator.

Banker's Algorithm

- We assume that each process declares its maximum claim on each resource type at the time of creation.
- The strategy is to keep the system in a *safe state*, that is, if every process were to exercise its maximum claim, then there would still be some sequence of allocations and deallocations that would enable the system to satisfy every process's requests. A system state in which this guarantee cannot be made is an *unsafe state*.
- A system can be in an unsafe state without deadlocking. A safe state guarantees that there cannot be a deadlock, but an unsafe state implies that the matter is out of the hands of the resource allocator and will be determined by the actions of the processes.

Banker's Algorithm (contd.)

A simple example with one resource. Total units available = 10.

Initial state of system:

Process	*Current*	*Max*
A	0	6
B	0	5
C	0	4
D	0	7

Is this a safe state?

Process	*Current*	*Max*
A	1	6
B	1	5
C	2	4
D	4	7

Is this a safe state?

Process	*Current*	*Max*
A	1	6
B	2	5
C	2	4
D	4	7

Banker's Algorithm (contd.)

- A set of n processes using a set of m resources.
- Let alloc be a table in which row i represents process p_i $(0 \leq i < n)$ and column j represents resources r_j $(0 \leq j < m)$. The entry alloc[i,j] represents the number of units of resource r_j held by the process p_i.
- The table maxc represents the maximum claim on resource r_j by process p_i.
- The table C represents the number of units in the system.
- Given the above data structures, the number of available resources can be computed as:

$$\text{avail[j]} = c_j - \sum_{0 \leq i \leq n} \text{alloc[i,j]}.$$

Banker's Algorithm (contd.)

1. Copy the alloc[i, j] table to a table named alloc'.
2. Given C, maxc, and alloc', compute the avail vector. First take the column sums for alloc': alloc'[*,j]. Then compute: avail[j] = c_j - alloc'[*, j]
3. Find p_i such that maxc[i,j] - alloc'[i,j] $\leq$ avail[j] for $0 \leq j < m$ and $0 \leq i < n$. If no such p_i exists, then the state is unsafe–halt the algorithm. If alloc'[i,j] is 0 for all i and j, the state is safe—halt the algorithm.
4. Set alloc'[i,j] to 0 to indicate that p_i could exercise its maximum claim. Then deallocate all resources to represent that p_i is not permanently blocked in the state that is being analyzed. Go back to Step 2.

The worst-case runtime: $\Theta(mn^2)$

Detection and Recovery

- Make a graph of resources and processes for use in detection. If there is only one of each resource, then a cycle in the resource graph represents a deadlock. Otherwise the conditions for deadlock are more complicated. How do we detect a cycle in a graph?
  Use depth-first search and look for a back-edge in the depth-first search tree. Can be done in time linear in the size of the graph. See Chapter 22 of the CLRS Algorithms book!
- Recovery will involve preempting processes or even destroying processes until the deadlock is resolved. The brute force approach is to destroy all processes by rebooting the machine.
- *Check-pointing* can help in saving the computation done by processes in case they are killed by the operating system. Checkpoints are used extensively in database management systems and other mission-critical applications.