Table of Contents

Deadlock

- Deadlock is a fundamental problem in distributed systems.
- A process may request resources in any order, which may not be known a priori and a process can request resource while holding others.
- If the sequence of the allocations of resources to the processes is not controlled.
- A deadlock is a state where a set of processes request resources that are held by other processes in the set.

Deadlock Detection:

1. Resource Allocation Graph (RAG) Algorithm:

- Deadlock detection typically involves constructing a resource allocation graph based on the current resource allocation and request status.
- The RAG algorithm identifies cycles in the graph, indicating the presence of a potential deadlock.
- However, the RAG algorithm suffers from scalability issues in large systems due to the overhead of maintaining the graph.

2. Resource-Requesting Algorithms:

- Another approach is to periodically check the state of resource requests and allocations to identify potential deadlocks.
- This approach involves tracking the resource allocation state and examining resource requests to detect circular waits.
- However, this method may have high overhead and can only identify deadlocks when they occur during the detection phase.

Deadlock Resolution:

1. Deadlock Prevention:

- Prevention involves ensuring that at least one of the necessary conditions for deadlock (mutual exclusion, hold and wait, no preemption, circular wait) is not satisfied.
- By carefully managing resource allocation and enforcing certain policies, deadlocks can be avoided altogether.
- However, prevention methods can be complex, restrictive, and may limit system performance or resource utilization.

2. Deadlock Avoidance:

- Avoidance involves dynamically analyzing resource requests and allocations to ensure that the system avoids entering an unsafe state where a deadlock can occur.
- Resource allocation is made based on resource requirement forecasts and resource availability to prevent circular waits.
- Avoidance requires a safe state detection algorithm to determine if a resource

allocation will lead to a deadlock.
- However, avoidance techniques may suffer from increased overhead and may limit system responsiveness.

3. Deadlock Detection with Recovery:

- Deadlock detection algorithms can be used to periodically check the system's state for potential deadlocks.
- Once a deadlock is detected, recovery mechanisms can be employed to resolve the deadlock.
- Recovery may involve aborting one or more processes, rolling back their progress, and reallocating resources to allow the system to continue.
- However, recovery mechanisms can be complex and may result in data loss or system instability.