SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

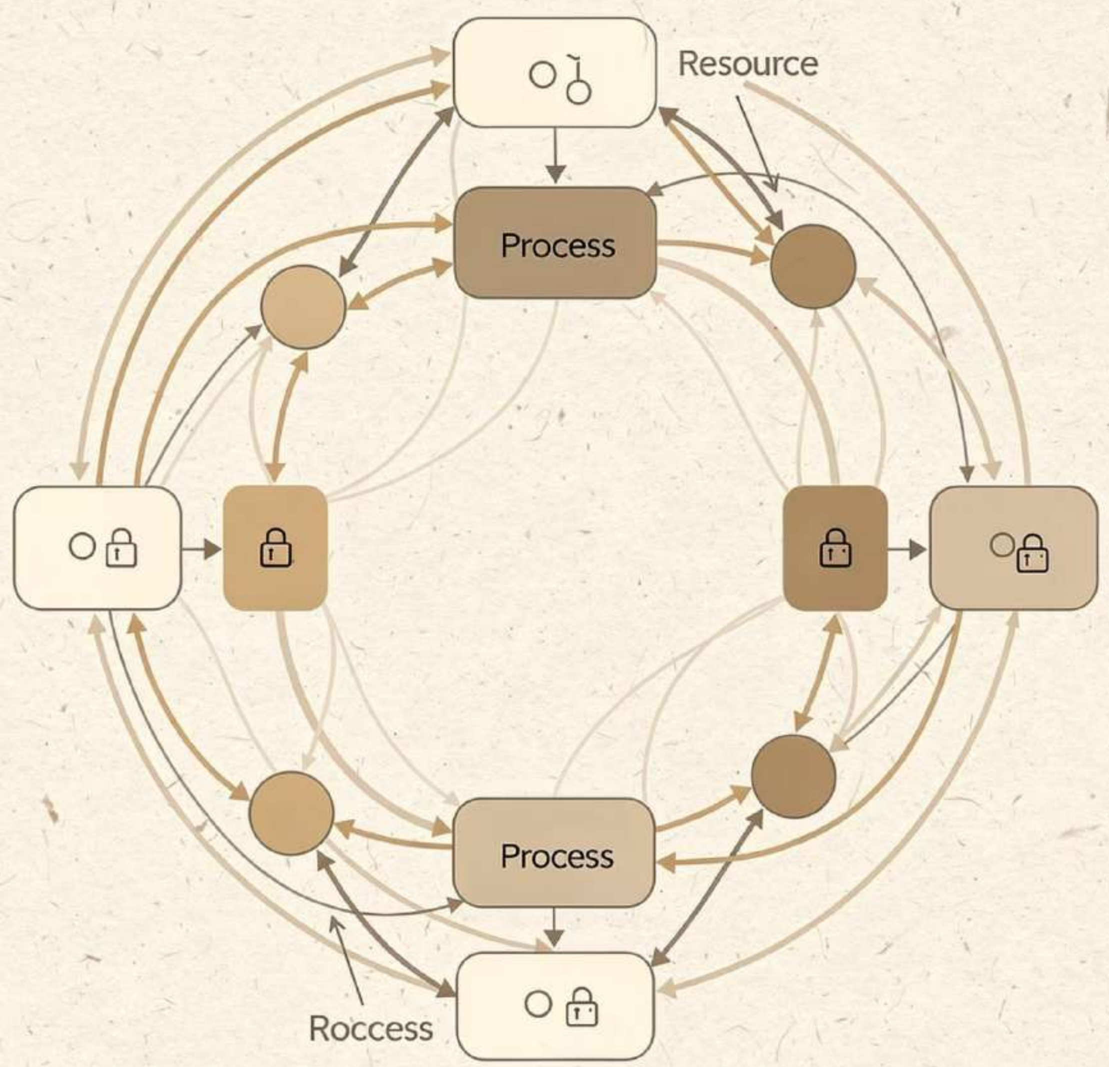
Department of Computer Science and Engineering

23CST206–OPERATING SYSTEMS AND VIRTUALIZATION

B.E- CSE /IV SEMESTER

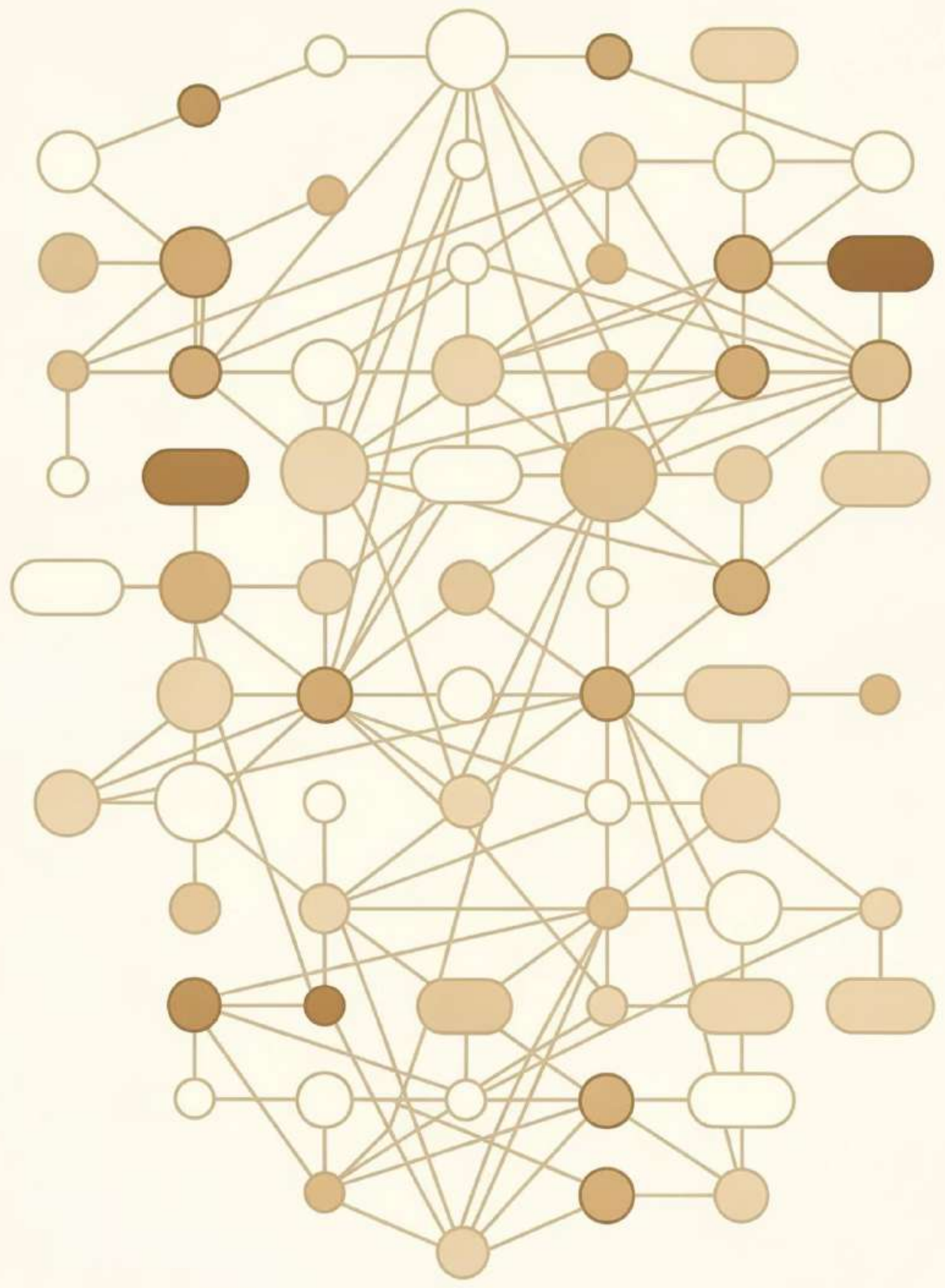
UNIT – II PROCESS MANAGEMENT

Topic 9: Deadlock Detection and Recovery



Deadlock Detection and Recovery

Overcoming Dead Ignorance in Operating Systems



DETECTION METHOD 1

Deadlock Detection: Single Instance Resources

01

Model System State

Use Resource Allocation Graph (RAG) to represent processes and resources

02

Detect Cycles

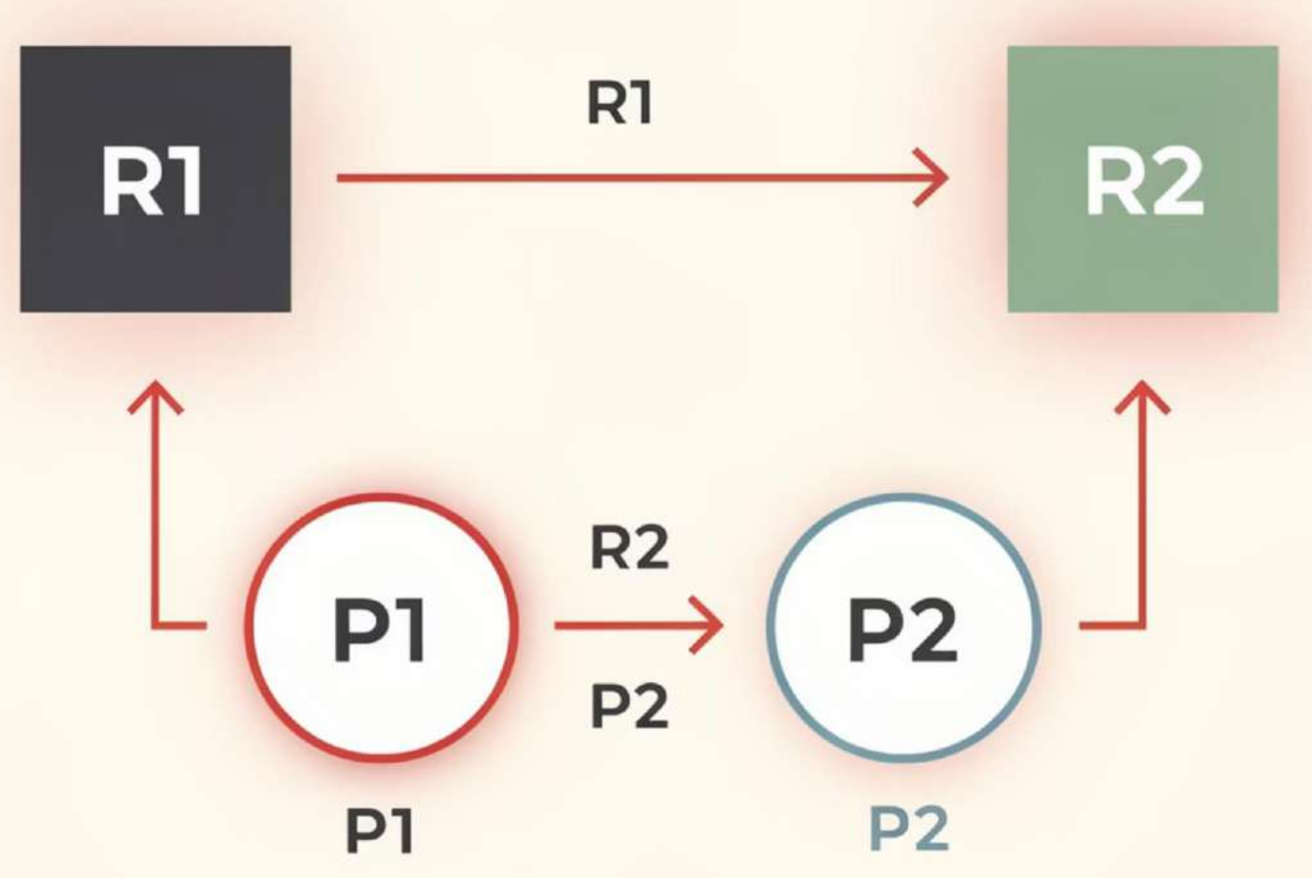
Presence of a cycle in RAG indicates deadlock

03

Confirm Deadlock

Cycle R1 → P1 → R2 → P2 confirms deadlock in single-instance systems

Deadlock Dipet: Single Instance Resources



Resource Allocation Graph

DETECTION METHOD 2

Deadlock Detection: Multiple Instance Resources

Why Cycle Detection Isn't Enough

In systems with multiple resource instances, cycle detection alone is insufficient to confirm deadlock.

Wait-For Graph Algorithm

Tracks dependencies between processes to identify circular waits

Periodic Checks

System state scanned regularly to detect deadlock conditions

DETECTION METHOD 2

Deadlock Detection: Multiple Instance Resources

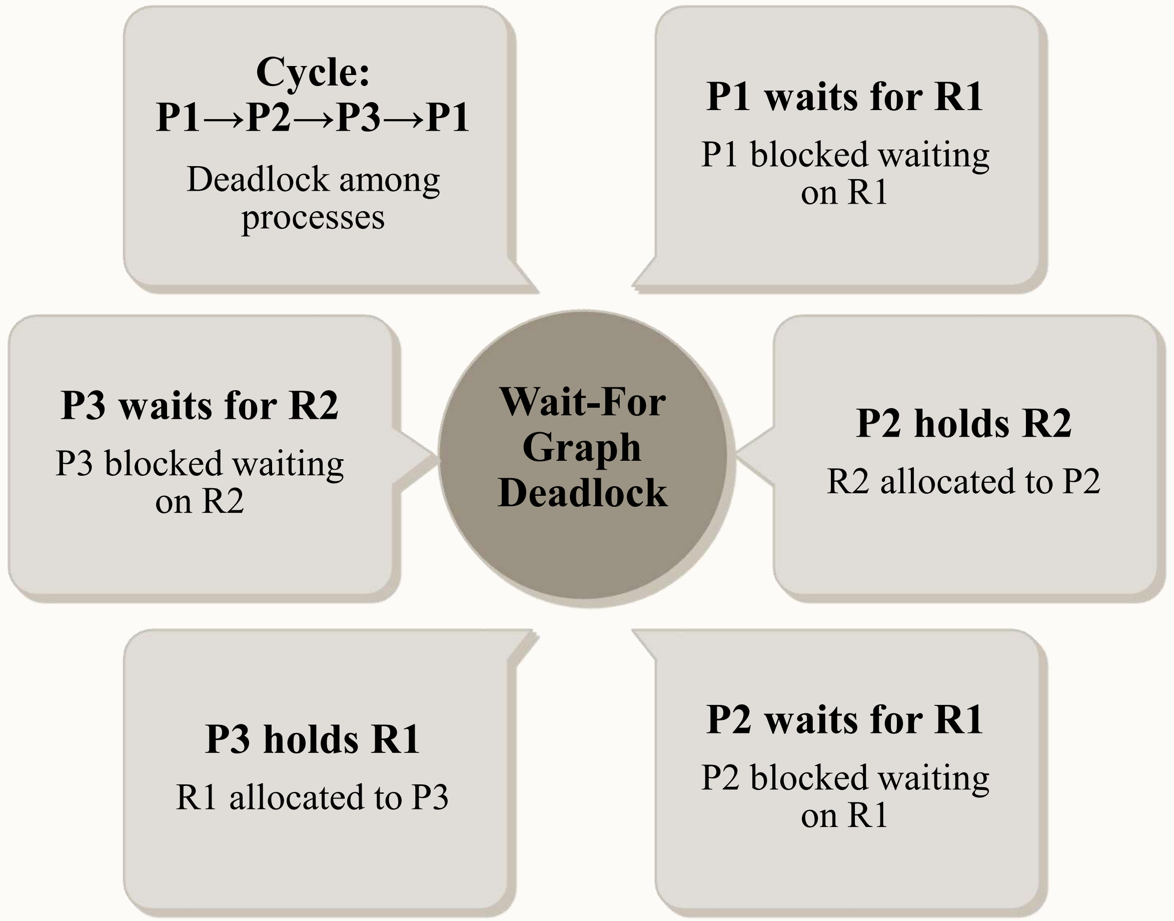
When resources have multiple instances, cycle detection in a standard Resource Allocation Graph is insufficient. We need more sophisticated methods.

The Wait-For Graph Algorithm

- Processes as Nodes
- Directed Edges for Waiting State
- Cycle Indicates Deadlock

Periodic Checks

To ensure system stability, the deadlock detection algorithm runs periodically to scan the current state and identify potential deadlocks before they escalate.



Deadlock Detection Algorithms Overview

Resource Allocation Graph

Visual representation of process-resource dependencies.

Wait-For Graph

Tracks which processes are waiting for others.

Periodic Scanning

Regular system state checks identify deadlocks.

Deadlock Recovery: Process Termination

1

Abort All Deadlocked Processes

Guaranteed to break deadlock but costly significant loss of work

2

Abort One at a Time

Less costly, requires repeated detection after each termination

RECOVERY STRATEGY 2

Deadlock Recovery: Resource Preemption

Select Victim Process

Choose which process will have resources temporarily taken away

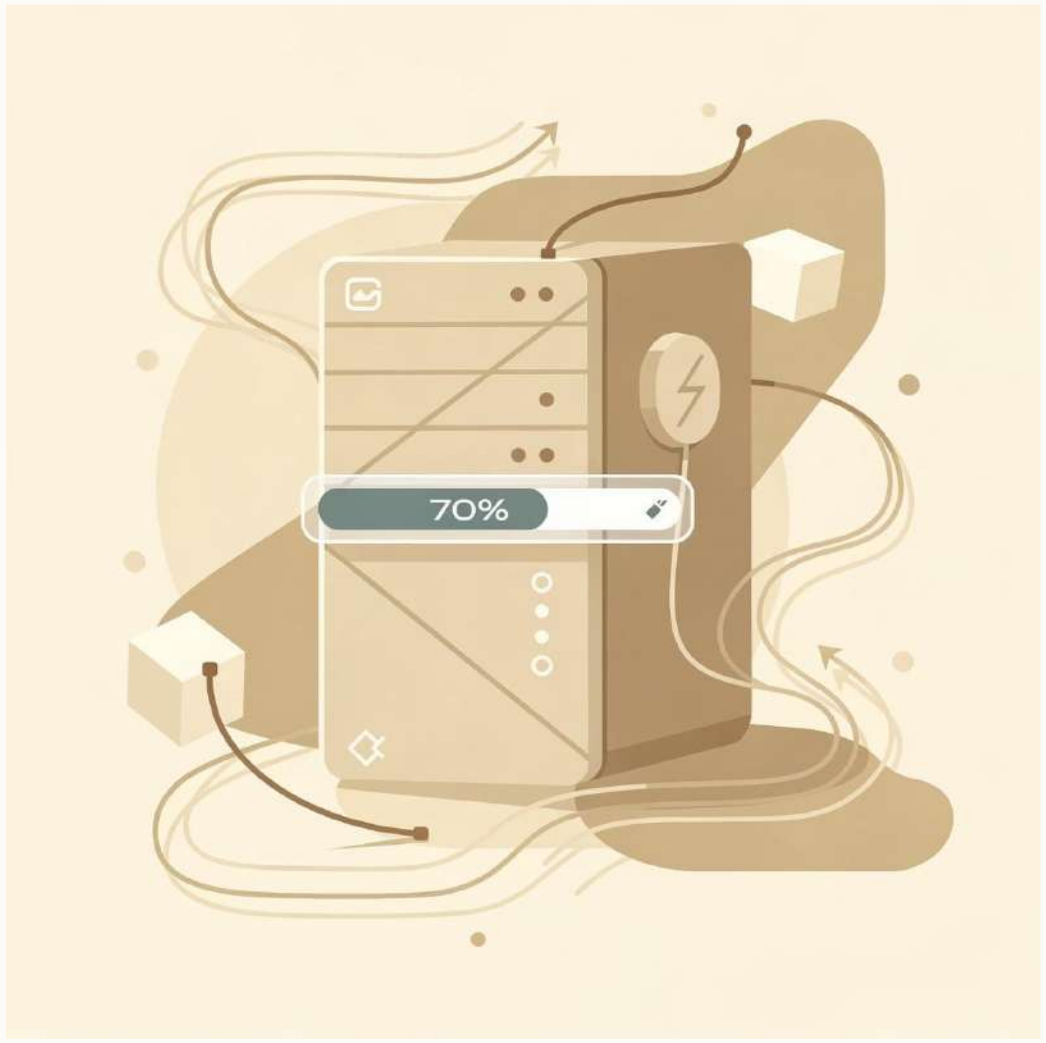
Preempt Resources

Take resources from victim and reassign to waiting processes

Rollback or Restore

Use checkpointing to restore victim process state when resources available

Rollback and Starvation Avoidance



Smart Recovery Strategies

Safe State Rollback

Return processes to checkpointed safe states before deadlock occurred

Priority Schemes

Prevent starvation by tracking how many times a process has been preempted

Balanced Approach

Optimizes system stability and resource utilization simultaneously

Smart Recovery Strategies

Safe State Rollback

Return processes to checkpodited safe states belbock occurred 4C4C4

Priority Schemes

Prevent starvaticn by tracking how many times a w process has been preempted

Balanced Approach

Optimizes system stability and resource utilization simuntanecusly

DESIGN THINKING

Problem Statement

How might we design smarter, less disruptive deadlock management in operating systems?

Current Challenge

Deadlocks cause system freezes and resource wastage

Inefficiency Issue

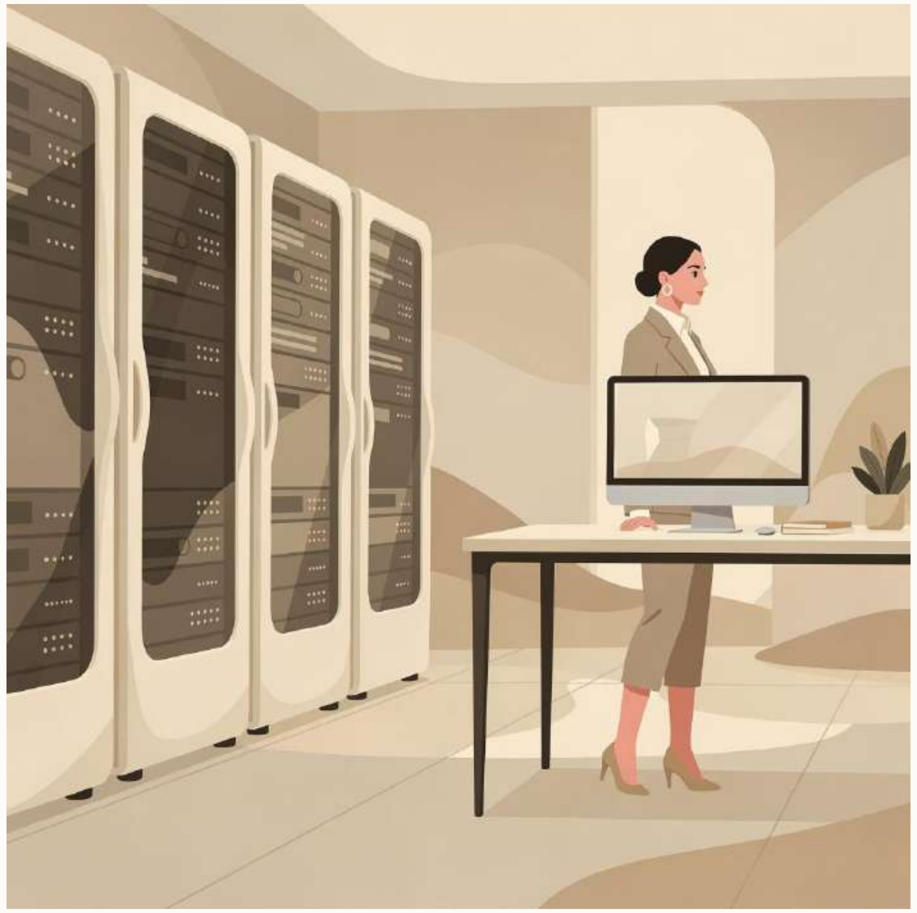
Existing detection and recovery methods can be costly or disruptive

Innovation Opportunity

Need for intelligent, adaptive solutions

EMPATHIZE

Understanding User and System Impact



End User Experience

- Application freezes
- System slowdowns
- Lost productivity
- Frustration and downtime

Administrator Challenges

- Complex recovery decisions
- Balancing multiple processes
- Minimizing disruption
- Maximizing throughput

DEFINE

Key Challenges in Deadlock Management

1 **Accurate Detection**

Identifying deadlocks accurately without imposing excessive computational overhead on the system

2 **Minimal Data Loss**

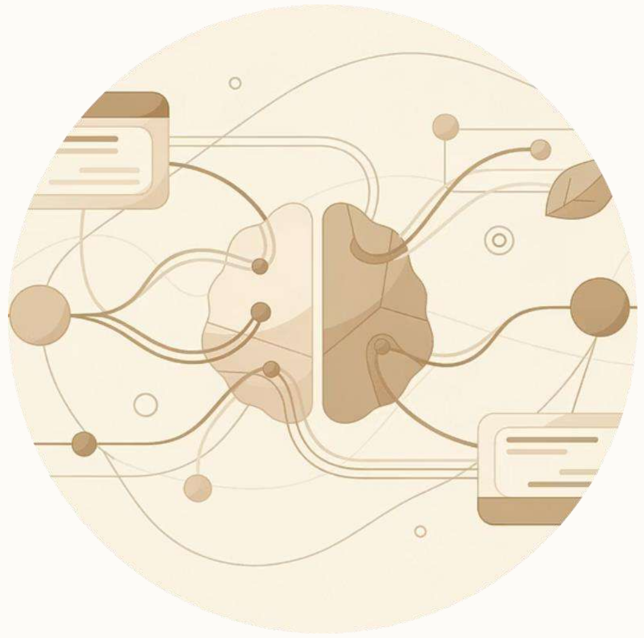
Recovering from deadlocks while preserving as much process work and system state as possible

3 **Fair Resource Allocation**

Balancing resource allocation fairness across processes with overall system performance goals

IDEATE

Innovative Approaches to Deadlock Handling



Predictive Analytics

Anticipate deadlocks before they occur using AI and machine learning

Adaptive Algorithms

Resource allocation that learns from historical system behavior patterns

Hybrid Recovery

Combine preemption and selective termination for optimal results

PROTOTYPE

Conceptual Model for Smarter Deadlock Recovery

Real-Time Monitoring

Integrate ML to detect patterns

Checkpointing

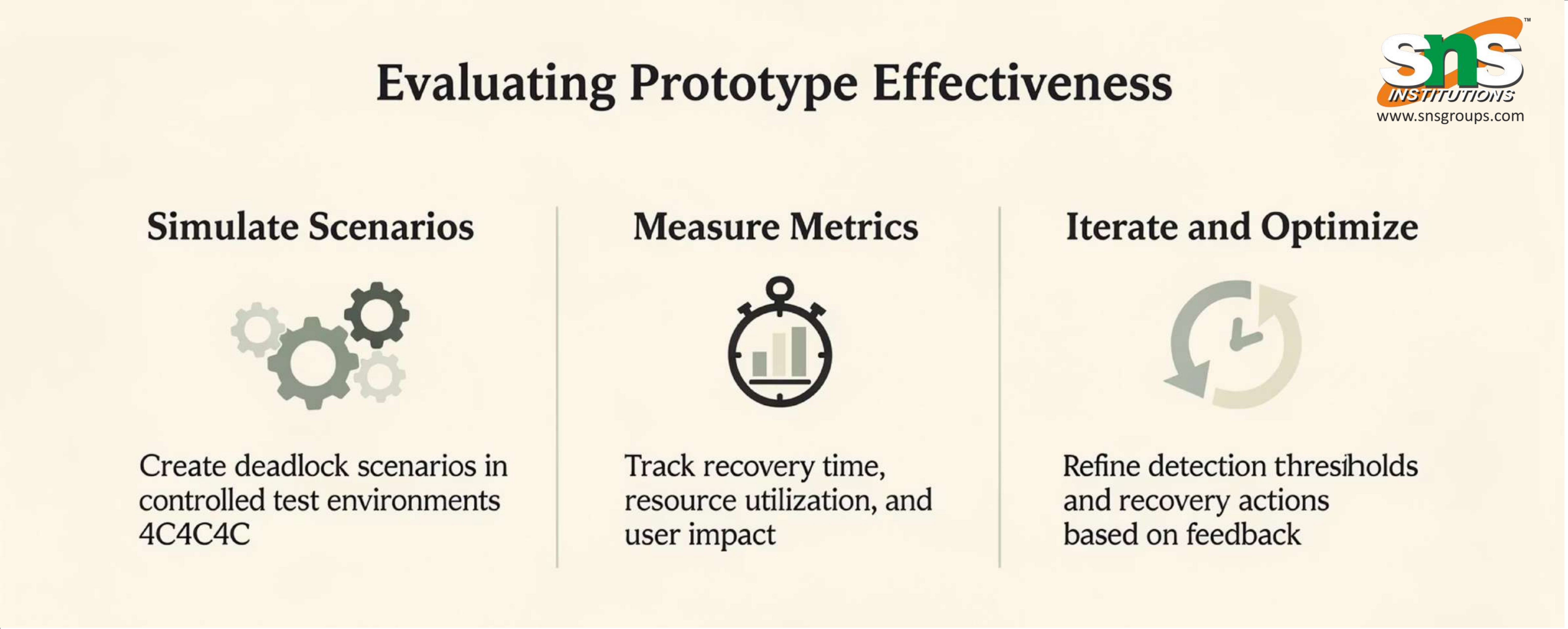
Quick rollback capability

Adaptive Recovery

Learn and optimize actions

User Notification

Prioritize critical processes



Evaluating Prototype Effectiveness

Simulate Scenarios

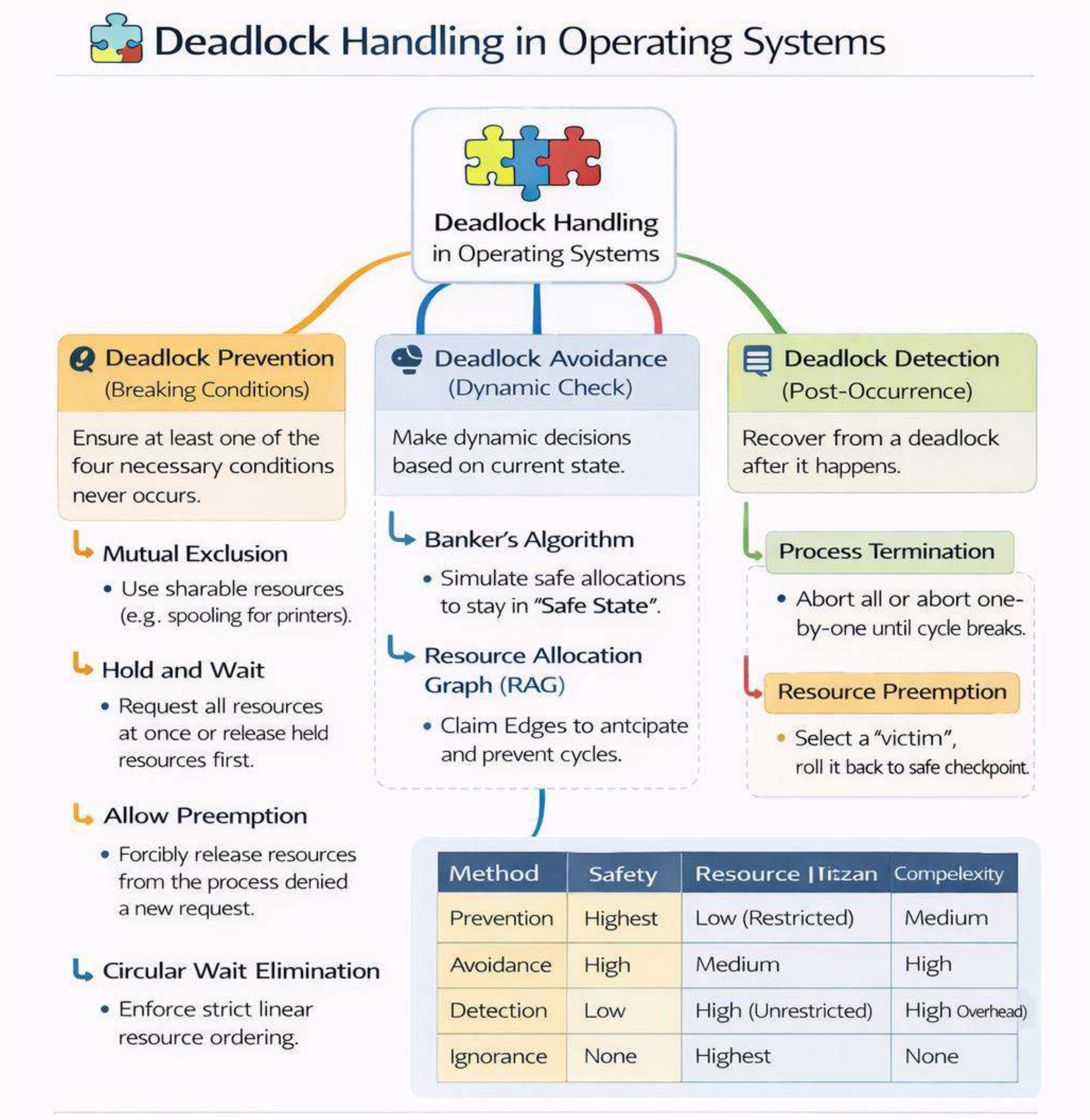
Create deadlock scenarios in controlled test environments 4C4C4C

Measure Metrics

Track recovery time, resource utilization, and user impact

Iterate and Optimize

Refine detection thresiholds and recovery actions based on feedback



Deadlock Handling in Operating Systems

Deadlock Handling in Operating Systems

Deadlock Prevention (Breaking Conditions)

Ensure at least one of the four necessary conditions never occurs.

- **Mutual Exclusion**
  - Use sharable resources (e.g. spooling for printers).
- **Hold and Wait**
  - Request all resources at once or release held resources first.
- **Allow Preemption**
  - Forcibly release resources from the process denied a new request.
- **Circular Wait Elimination**
  - Enforce strict linear resource ordering.

Deadlock Avoidance (Dynamic Check)

Make dynamic decisions based on current state.

- **Banker's Algorithm**
  - Simulate safe allocations to stay in "Safe State".
- **Resource Allocation Graph (RAG)**
  - Claim Edges to antcipate and prevent cycles.

Deadlock Detection (Post-Occurrence)

Recover from a deadlock after it happens.

- **Process Termination**
  - Abort all or abort one-by-one until cycle breaks.
- **Resource Preemption**
  - Select a "victim", roll it back to safe checkpoint.

Method	Safety	Resource |Iitzan	Compelexity
Prevention	Highest	Low (Restricted)	Medium
Avoidance	High	Medium	High
Detection	Low	High (Unrestricted)	High Overhead)
Ignorance	None	Highest	None

Puzzle: Identify the Deadlock Scenario

Given Scenario

Three processes **P1, P2, and P3** and three resources **R1, R2, and R3**

1 Process P1
Holds R1 and waits for R2

2 Process P2
Holds R2 and waits for R3

3 Process P3
Holds R3 and waits for R1

Is this a deadlock? Why or why not?

Puzzle Solution

Circular Wait Detected

$P1 \rightarrow R2 \rightarrow P2 \rightarrow R3 \rightarrow P3 \rightarrow R1 \rightarrow P1$

All Four Coffman Conditions Met

1. **Mutual Exclusion:** Resources held exclusively
2. **Hold and Wait:** Processes hold resources while waiting
3. **No Preemption:** Resources cannot be forcibly taken
4. **Circular Wait:** Closed chain of waiting processes

THANK YOU

OPERATING SYSTEMS

DEADLOCK MANAGEMENT

SYSTEM DESIGN