SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

www.snsgroups.com

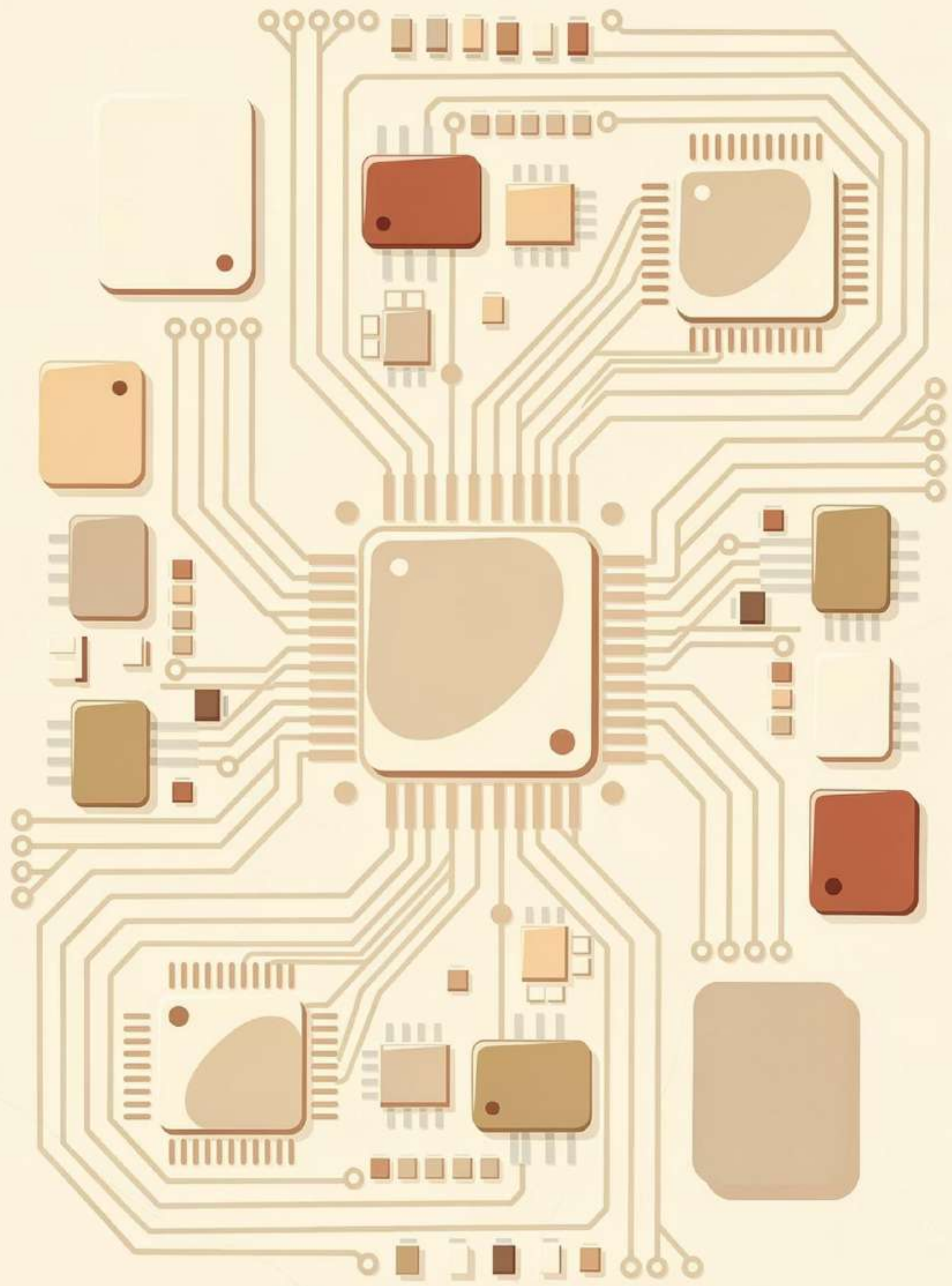
Department of Computer Science and Engineering

23CST206–OPERATING SYSTEMS AND VIRTUALIZATION

B.E- CSE /IV SEMESTER

UNIT – II PROCESS MANAGEMENT

Topic 9: Deadlock Prevention & Avoidance



Deadlock Prevention & Avoidance in Operating Systems

Understanding critical strategies for resource management in concurrent systems

THE PROBLEM

What is Deadlock?

- A deadlock occurs when processes in a system are blocked, each waiting for resources held by others creating a circular wait with no resolution.
- The Classic Scenario

Process A holds Resource 1, needs Resource 2

Process B holds Resource 2, needs Resource 1

Neither can proceed → system frozen

Four Necessary Conditions for Deadlock

Mutual Exclusion

Resources cannot be shared—only one process uses a resource at a time

Hold and Wait

Processes hold resources while waiting for additional ones

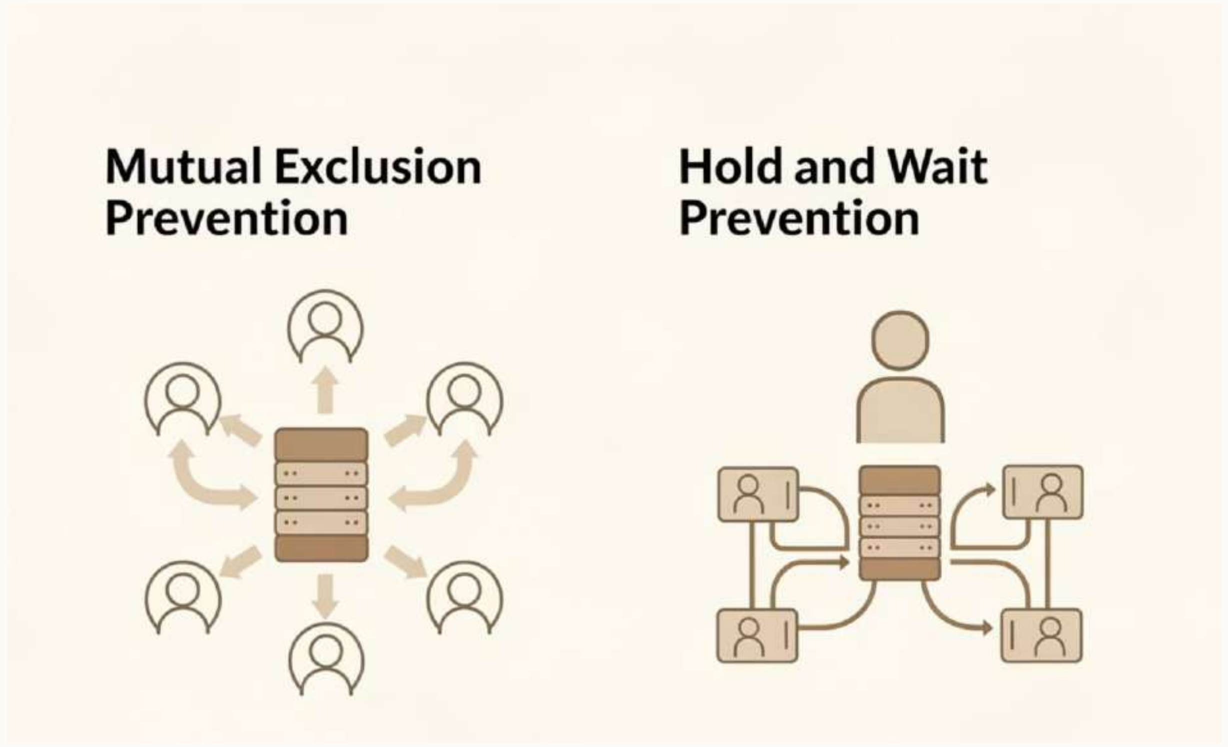
No Preemption

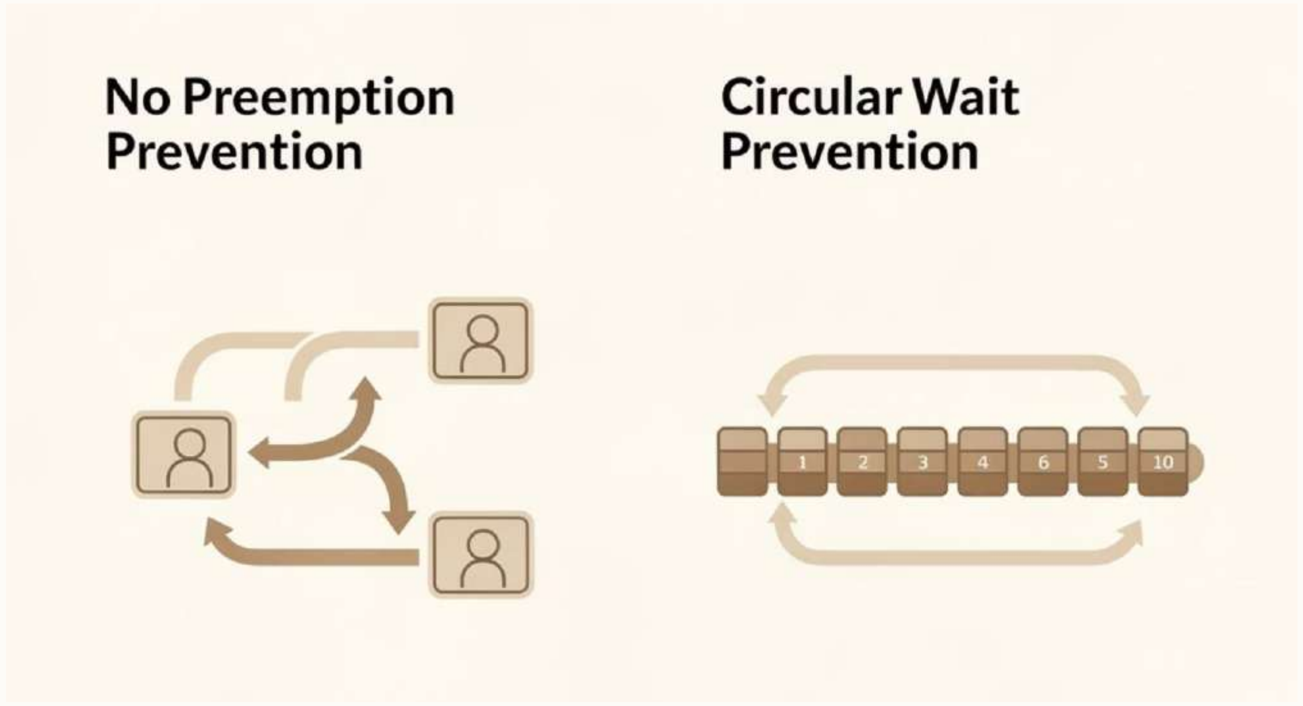
Resources cannot be forcibly taken—only released voluntarily

Circular Wait

Chain of processes, each waiting for a resource held by the next

Deadlock Prevention Strategies



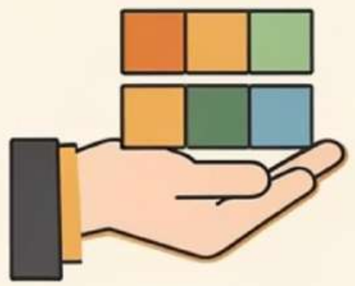


Deadlock Prevention Strategies

SNS INSTITUTIONS
www.snsgroups.com

Break Mutual Excleution

Make resources share shlad possible (e.g.) read-only files)



Eliminate Excestion

Require processes to request all resources upriblet (e.g, execution

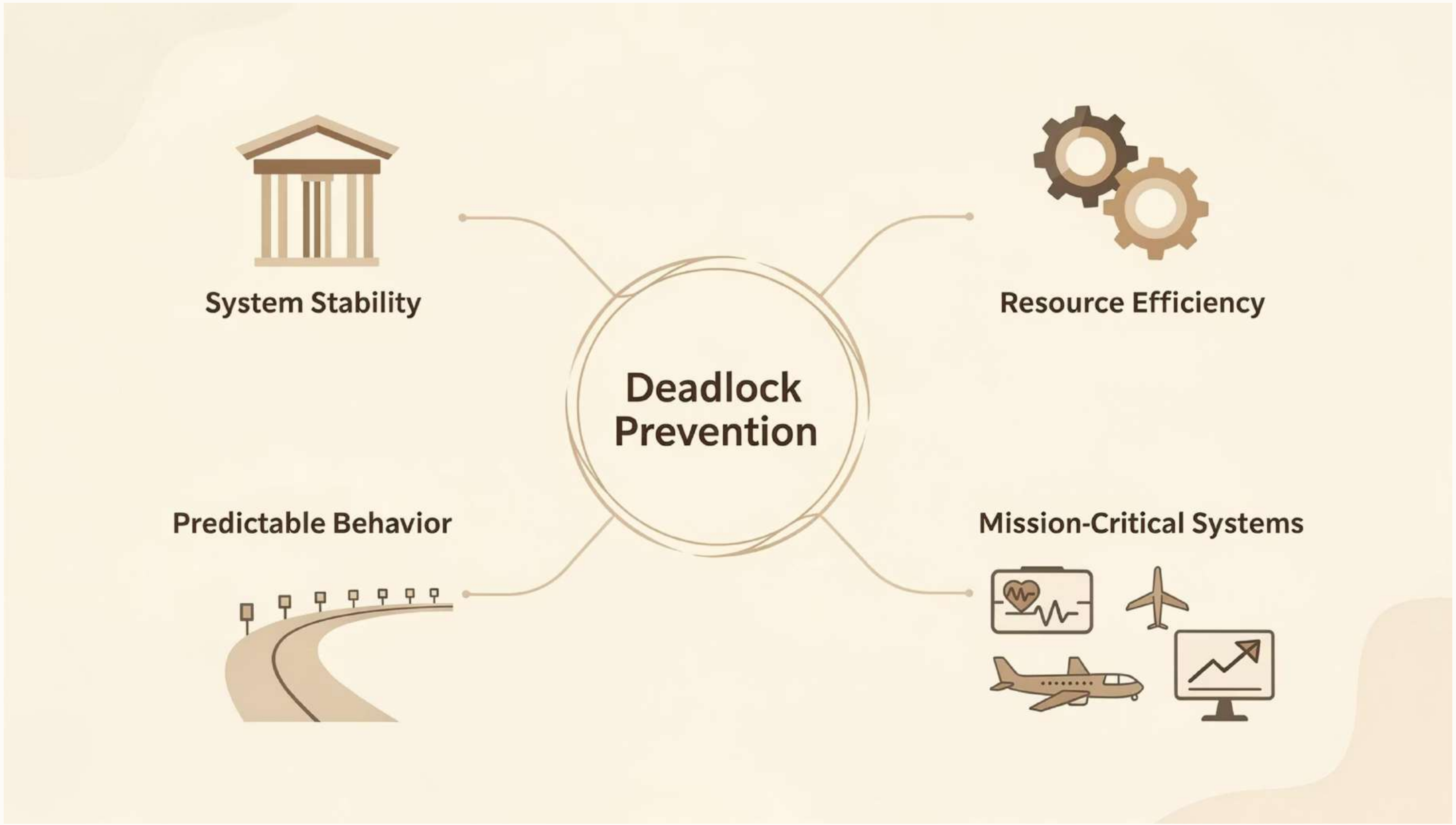
Eliminate Hold & Wait

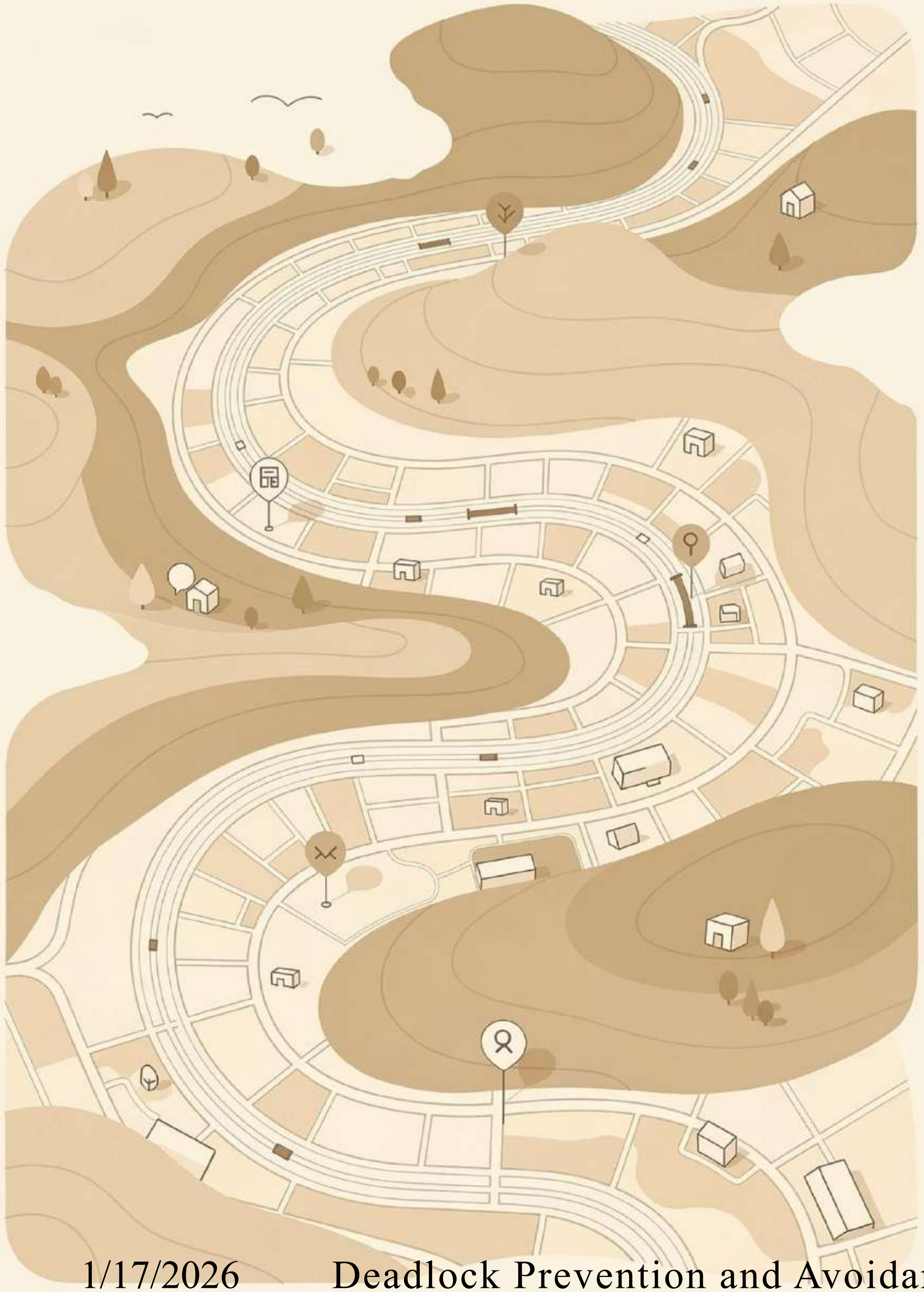
Require processes to reques blre upfromt befrorean execution

Elainicral Exleution

Require processes to resources.supfnotmt beafroreyrless

Why Deadlock Prevention Matters





Deadlock Avoidance

1

Dynamic Assessment

System continuously analyzes resource allocation patterns before granting requests

2

Safe State Analysis

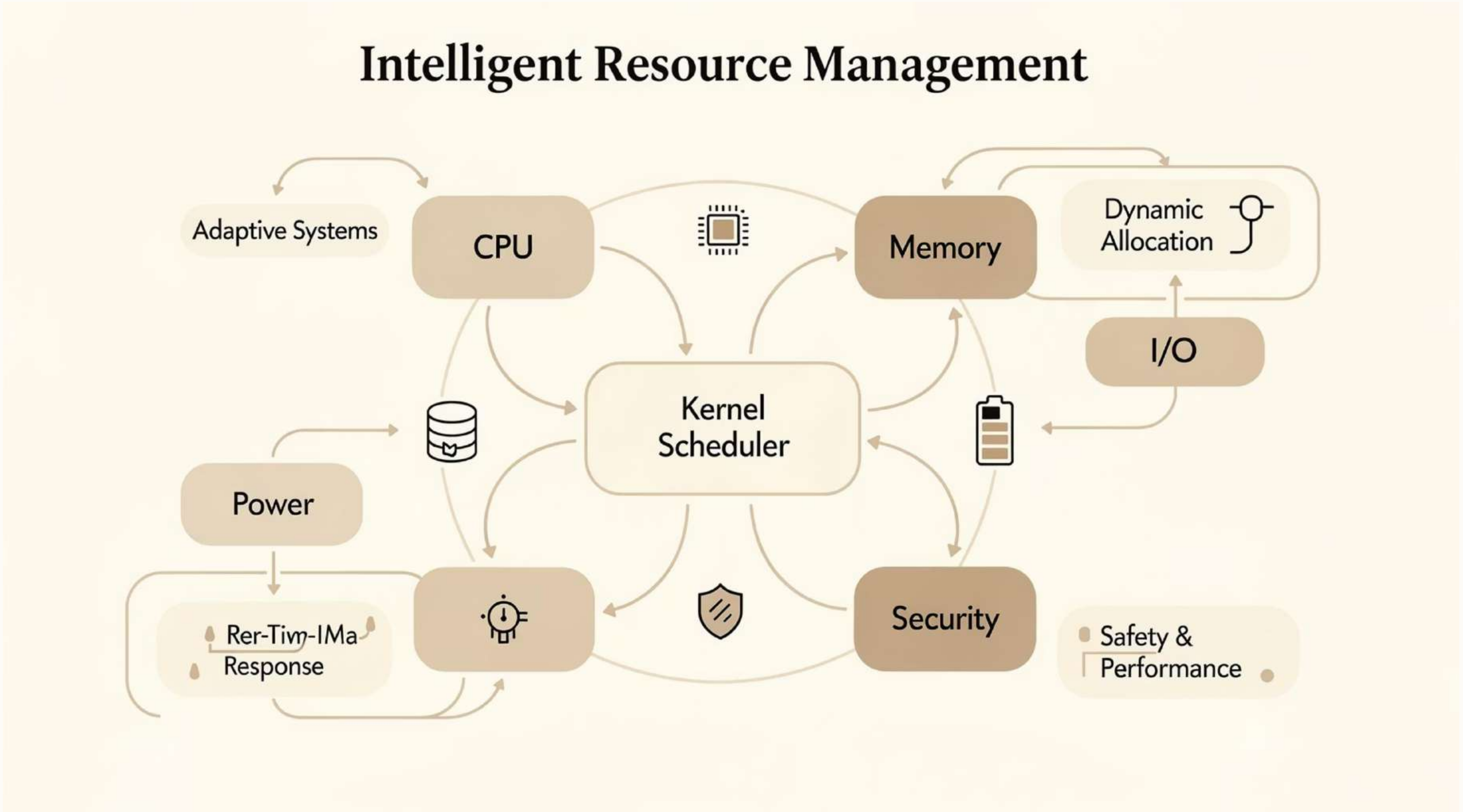
Ensures system remains in "safe state" where all processes can complete

3

Strategic Denial

May temporarily deny safe requests to prevent future unsafe states

The Role of Avoidance in Modern OS



DESIGN THINKING

Problem Definition

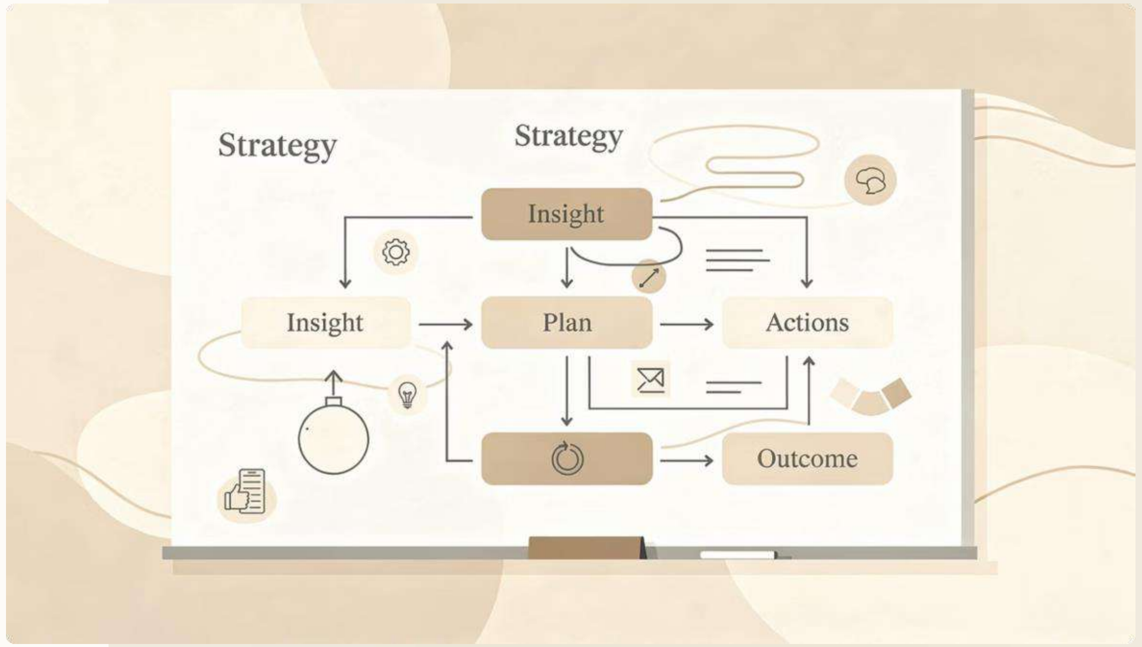
How might we manage limited system resources among competing processes to ensure continuous progress while maximizing efficiency?

Prevention Challenge

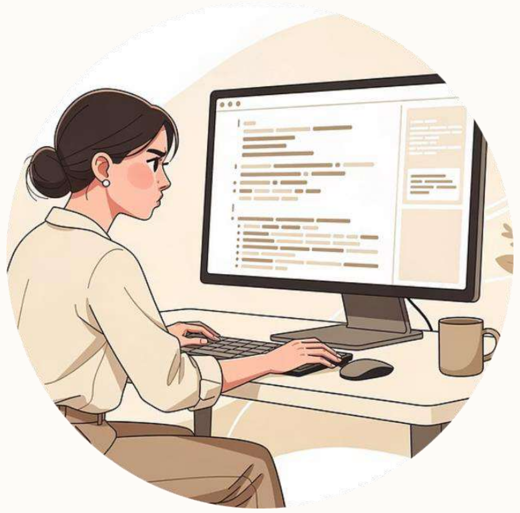
Design constraints that eliminate deadlock conditions without crippling system performance

Avoidance Challenge

Create algorithms that predict unsafe states and make real-time resource allocation decisions

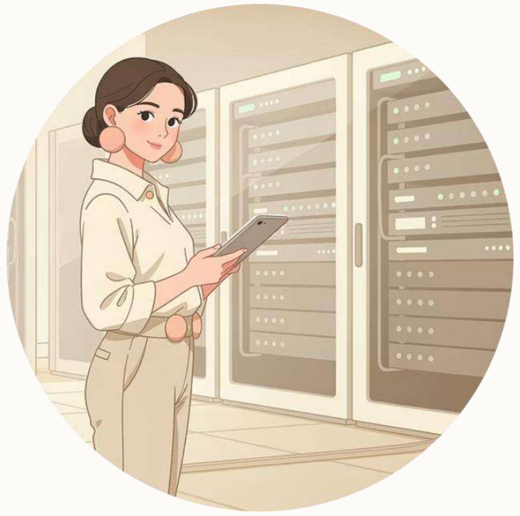


Empathize: Understanding Stakeholders



System Developers

Need simple, reliable mechanisms that don't complicate code

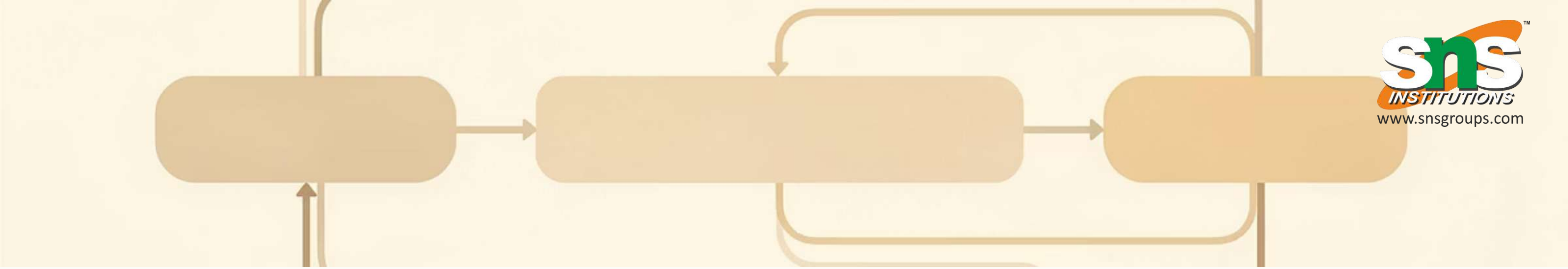


Database Admins

Require transaction integrity without sacrificing performance

End Users

Expect responsive systems that never freeze or lose data



Define & Ideate: Solution Approaches

Resource Ordering

Assign global priority to resources; always request in ascending order

1

Wait-Die Scheme

Timestamp-based: older processes wait, younger processes die and restart

3

2

Banker's Algorithm

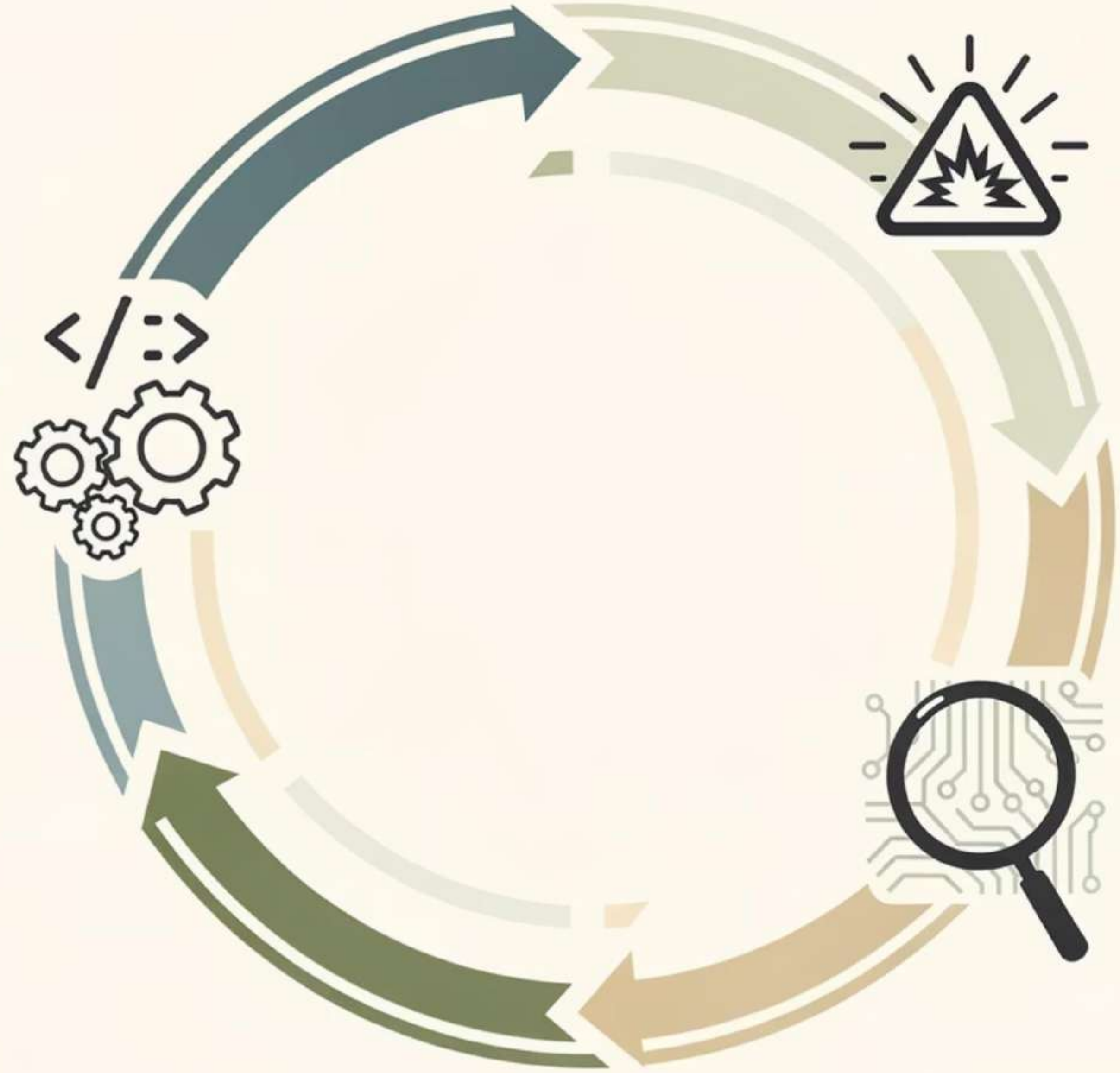
Track max needs; grant requests only if safe state maintained

4

Wound-Wait Scheme

Older processes preempt younger; younger processes wait

Prototype & Test



Stress Testing

- Push system to extreme conditiomany concurrent processes
- Measure Performance
- Track throughout, latency, and resource utilization

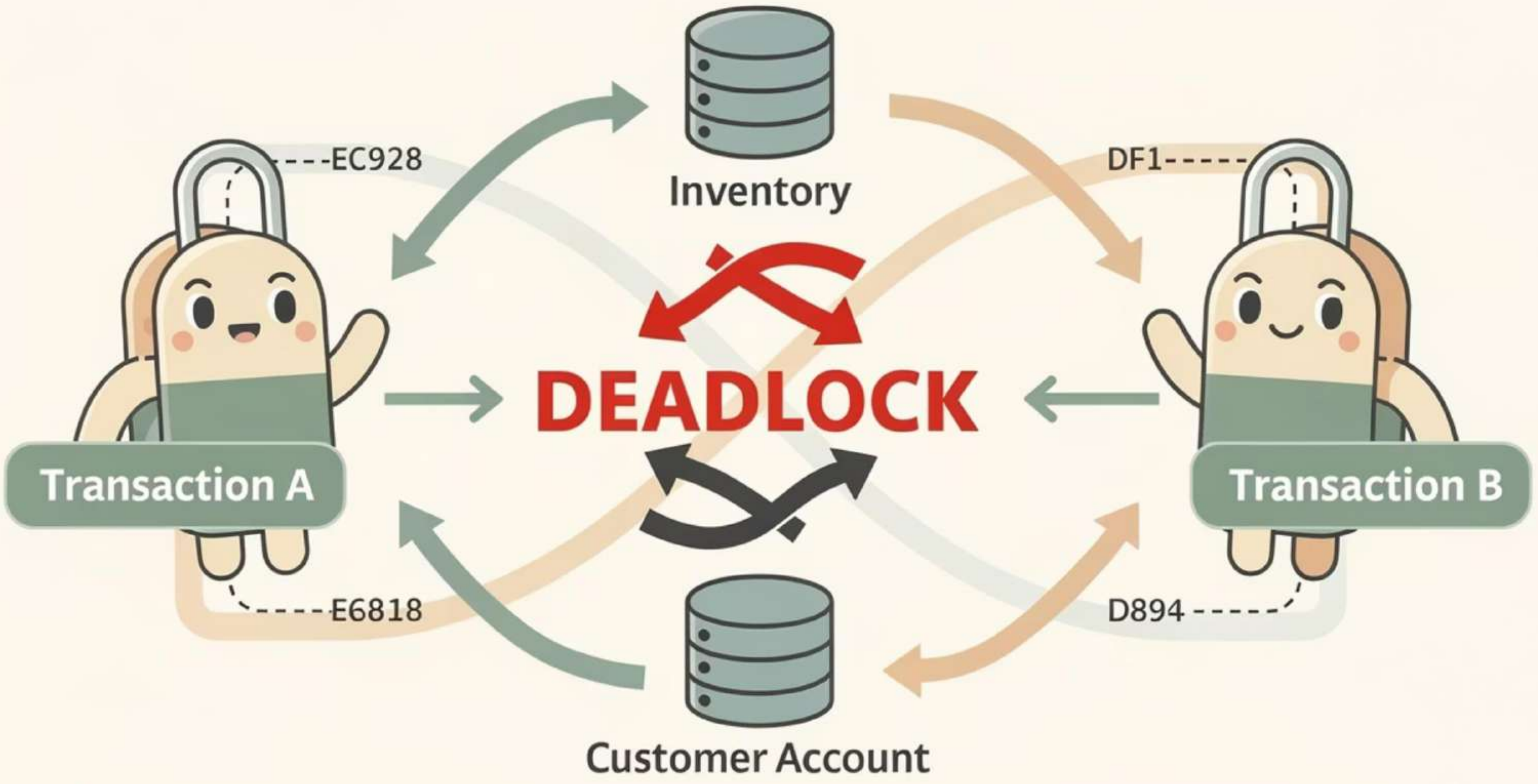
Impleentation Testing

- Simulate Workload
- Create test scenarios with
- varyicing ressource demands

Refine Algorithm

- Optimize based on real-world bottlaecks discovered

Database Transaction Management



Prevention

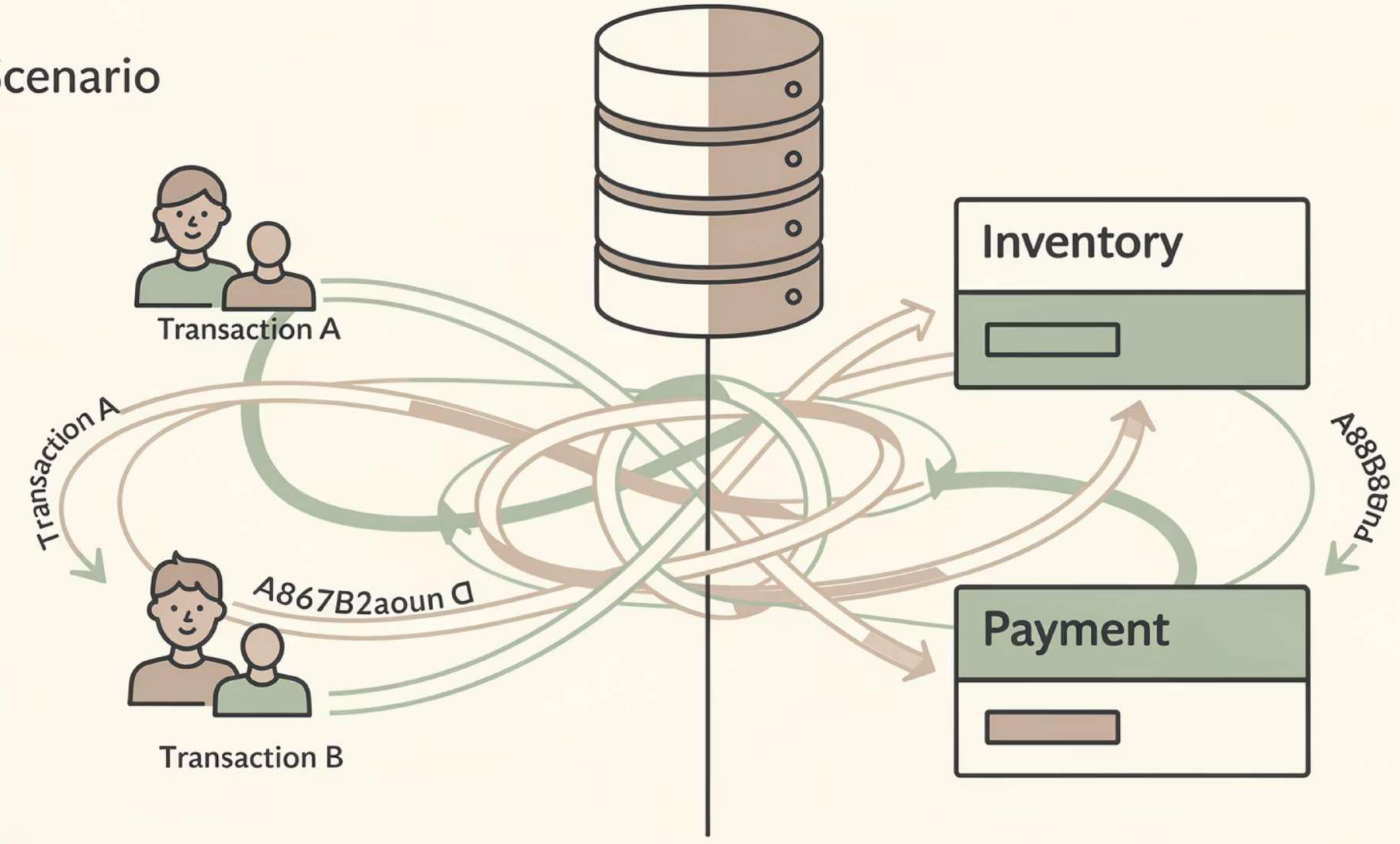
Thre locsis in tneagioln tiodes, pereso tio: se lipentha.

Customer Account

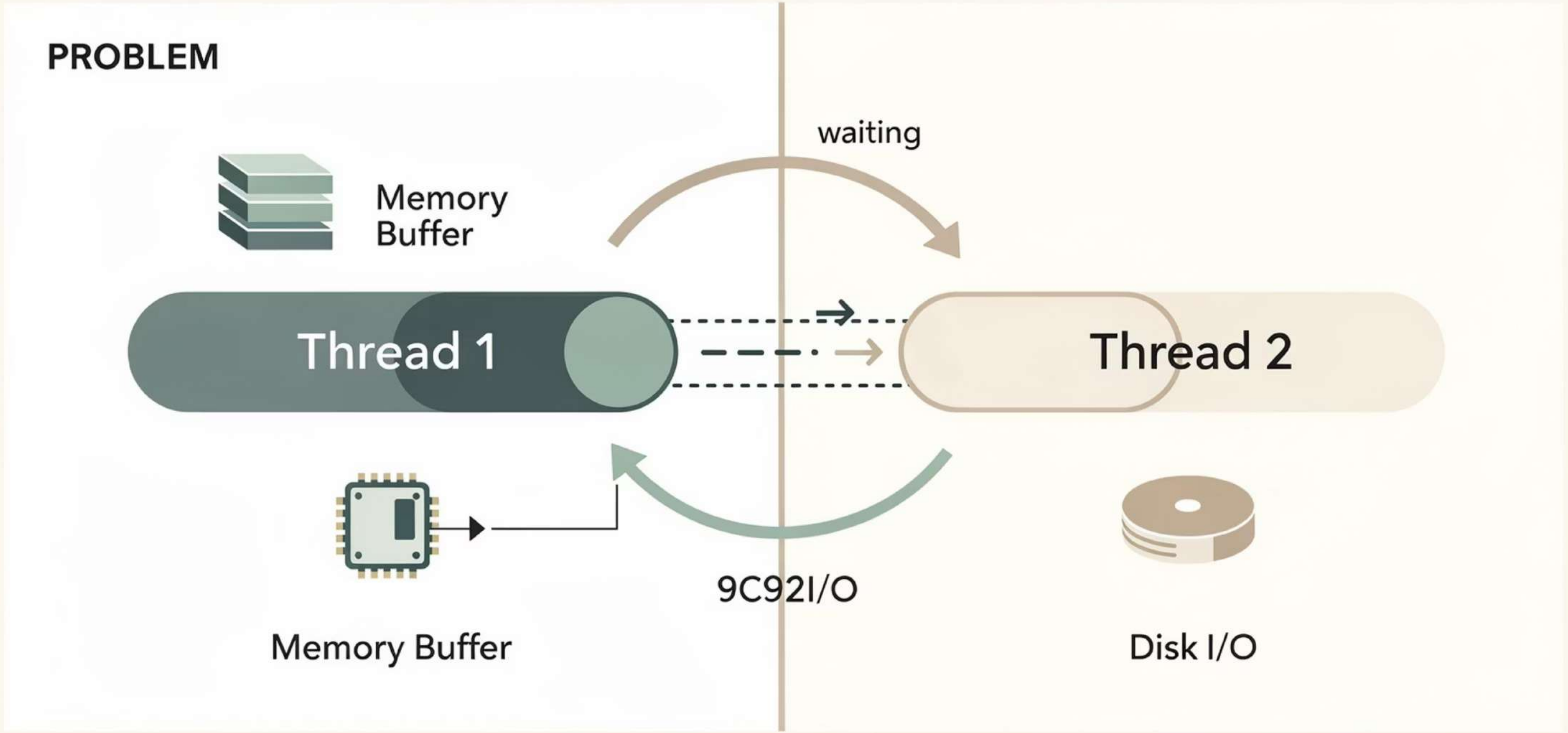
Dase isapeiongis aпy opposite order pore deloiand oose pyendiay.

Example 1: Database Transaction Management

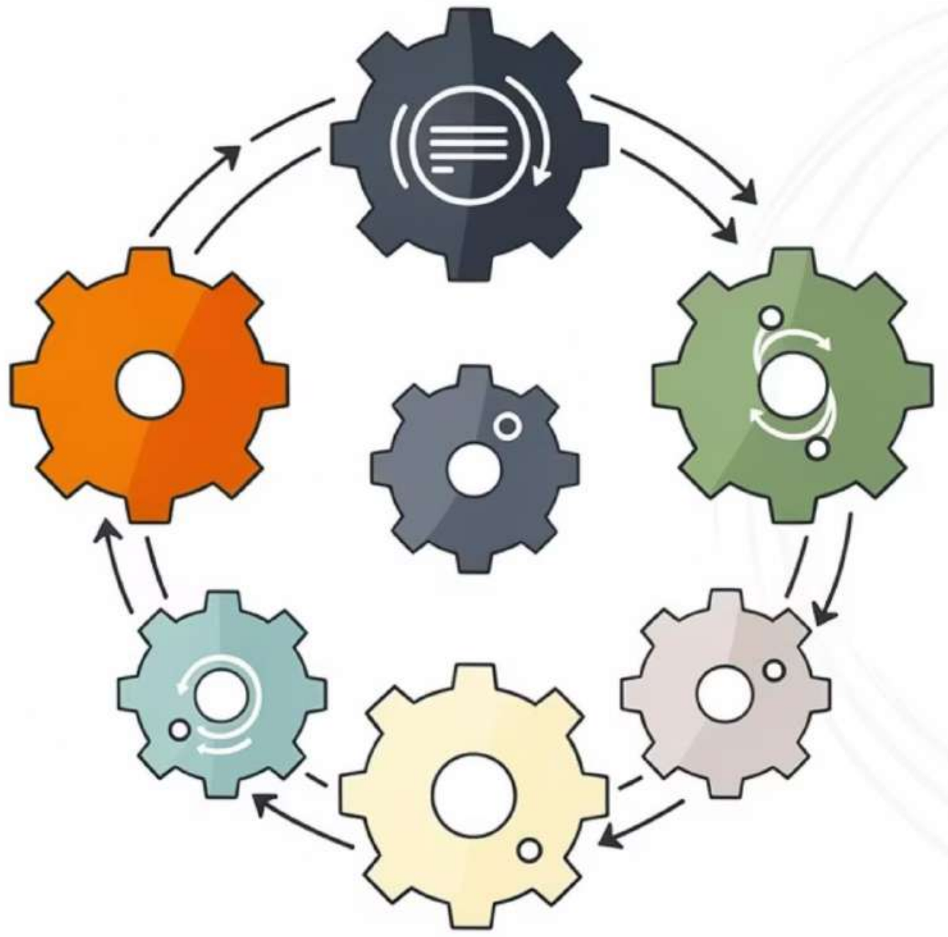
The Scenario

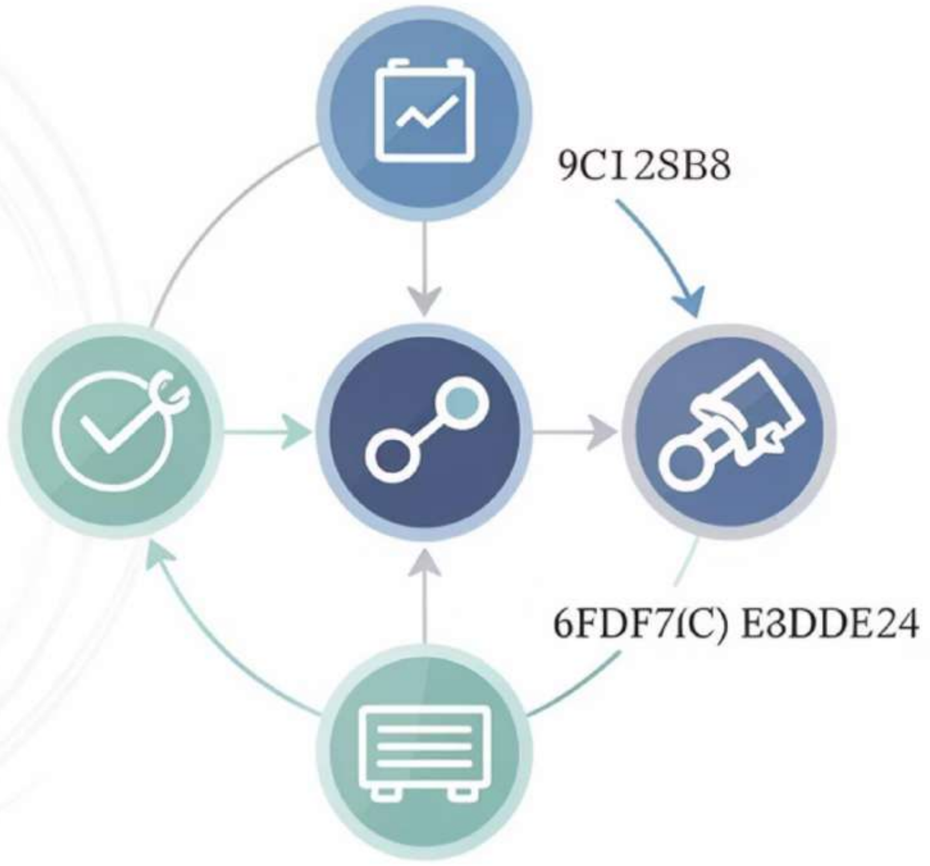


Deadlock in Video Editing: A Case Study



Deadlock Prevention & Avoidance





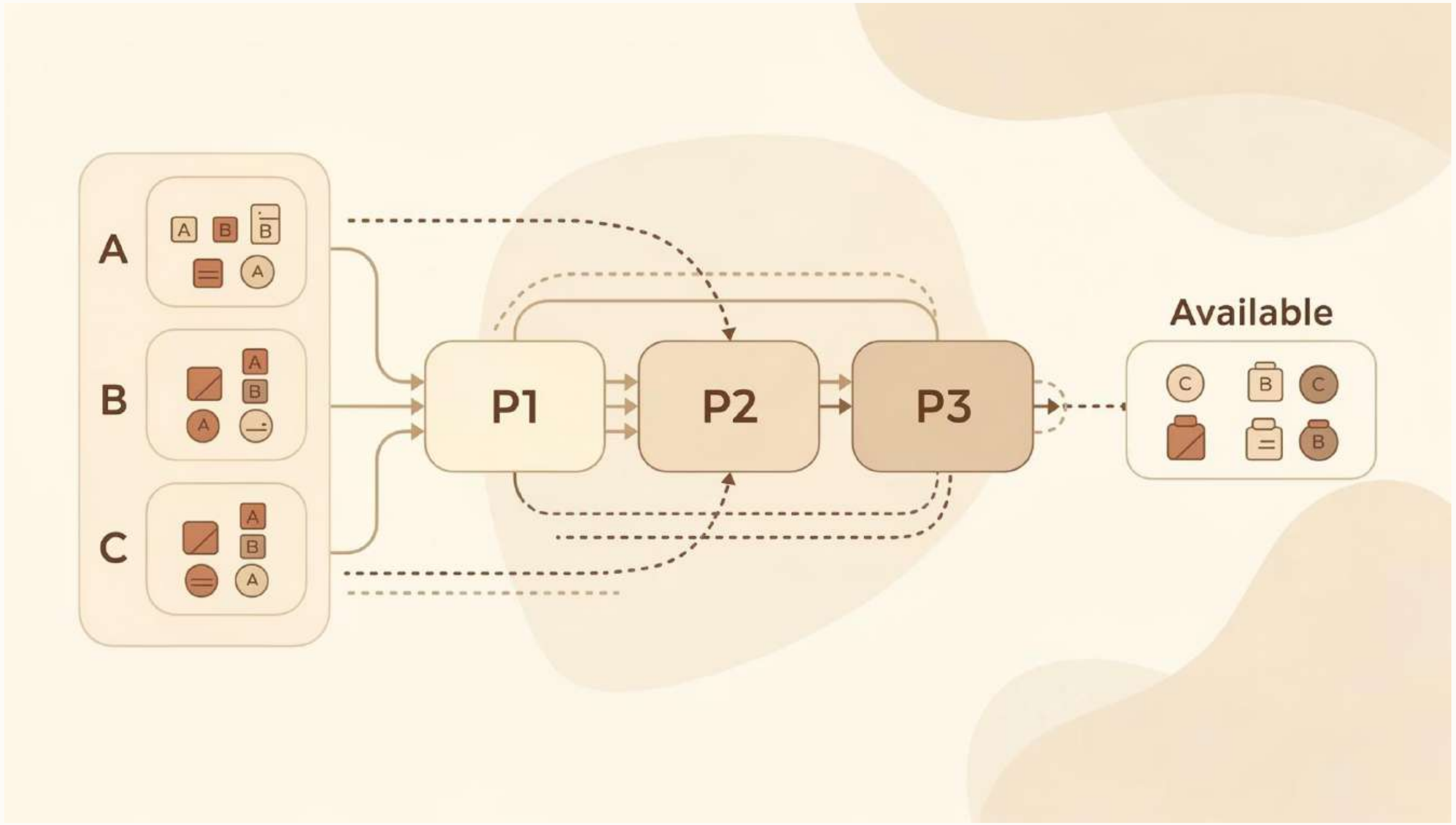
Prevention and Avoid·ance

Deadlock

CHALLENGE

The Resource Allocation Puzzle

A system has 3 processes (P1, P2, P3) and 3 resource types (A=10 instances, B=5 instances, C=7 instances).



Questions:

1. Is the system currently in a safe state?
2. If P2 requests (1, 0, 2), should the request be granted?
3. Explain your reasoning using the Banker's Algorithm

✓ SOLUTION

Puzzle Solution

01

Calculate Need Matrix

Need = Maximum - Allocated

P1: (7,4,3) | P2: (1,2,2) | P3: (6,0,0)

02

Find Safe Sequence

Available: (3,3,2) → P2 can finish (need ≤ available)

After P2: (5,3,2) → P1 can finish

After P1: (5,4,2) → P3 can finish

Safe sequence exists: P2 → P1 → P3

03

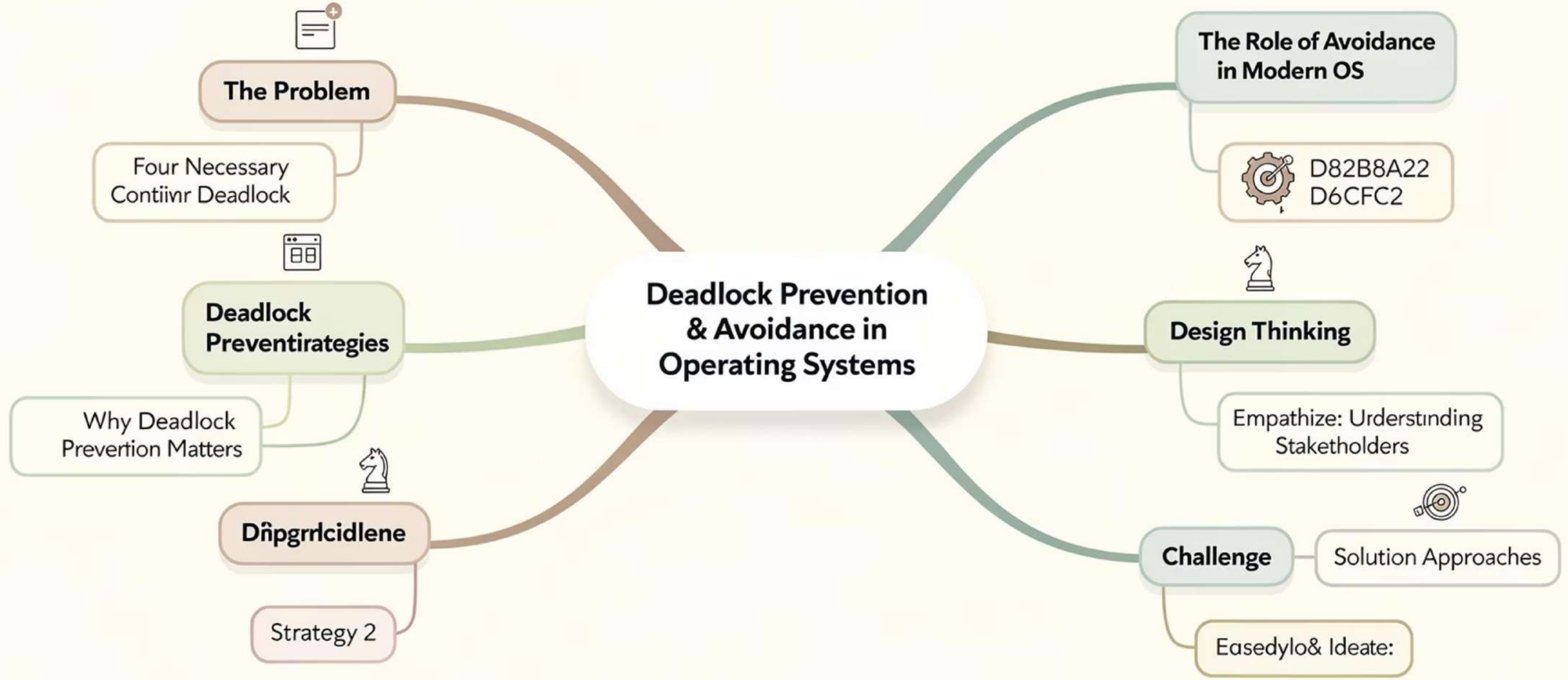
Evaluate P2's Request

Request (1,0,2). New available: (2,3,0)

P2's new need: (0,2,0). Check if any process can complete...

No safe sequence found → DENY request

Answer: Yes, currently safe. No, don't grant P2's request—it would create an unsafe state where no process can complete


Deadlock Prevention & Avoidance in Operating Systems
The Problem
Four Necessary Contiivr Deadlock
Deadlock Preventirategies
Why Deadlock Prevertion Matters
Dîpgrrlcidlene
Strategy 2
The Role of Avoidance in Modern OS
D82B8A22 D6CFC2
Design Thinking
Empathize: Urderstınding Stakeholders
Challenge
Solution Approaches
Easedylo& Ideate:

www.snsgroups.com

Thank You