AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Architecture & Design of Embedded Real-Time Systems (TI-AREM)

Resource Patterns

B.D. Chapter 7. 301-351

Version: 23-2-2015

Agenda

1. Critical Section Pattern
2. Priority Inheritance Pattern
3. Highest Locker Pattern

---

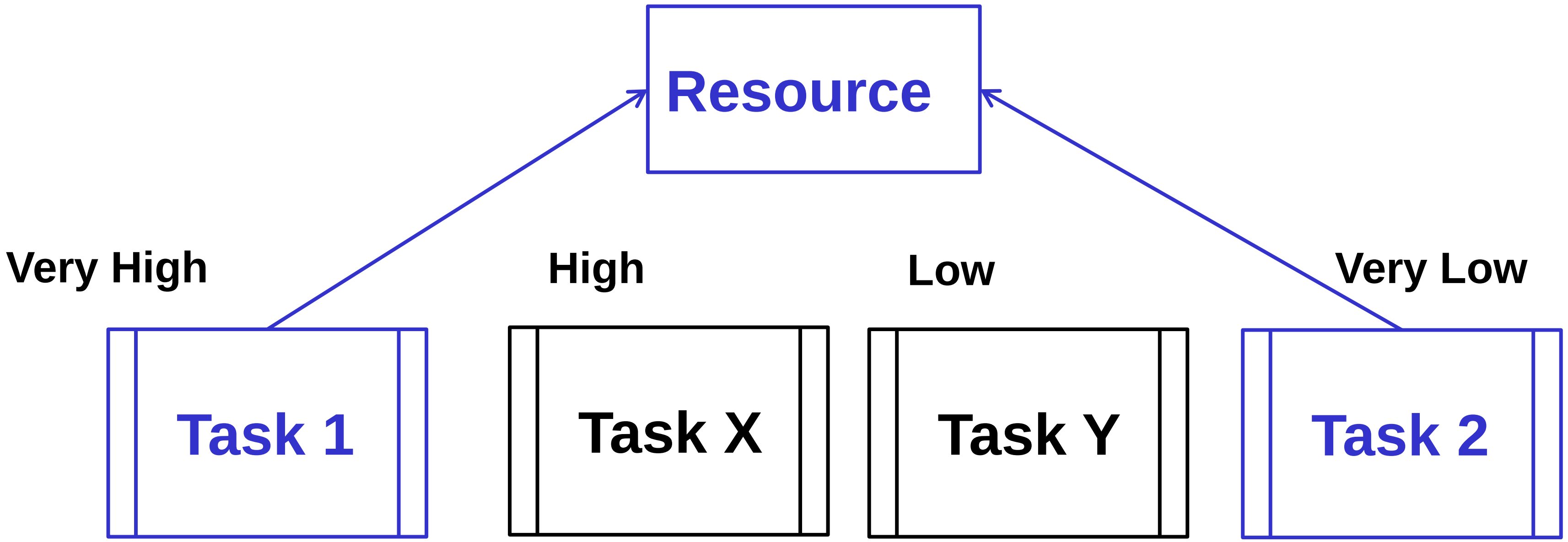
4. Priority Ceiling Pattern
5. Simultaneous Locking Pattern
6. Ordered Locking Pattern

Introduction to Resource Patterns

- All these patterns tackles problems using **preemptive scheduling**
- Three problems:
  - – Protection of a resource
  - – Priority inversion problem
  - – Deadlock problem

AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Unbounded Task Blocking (1)

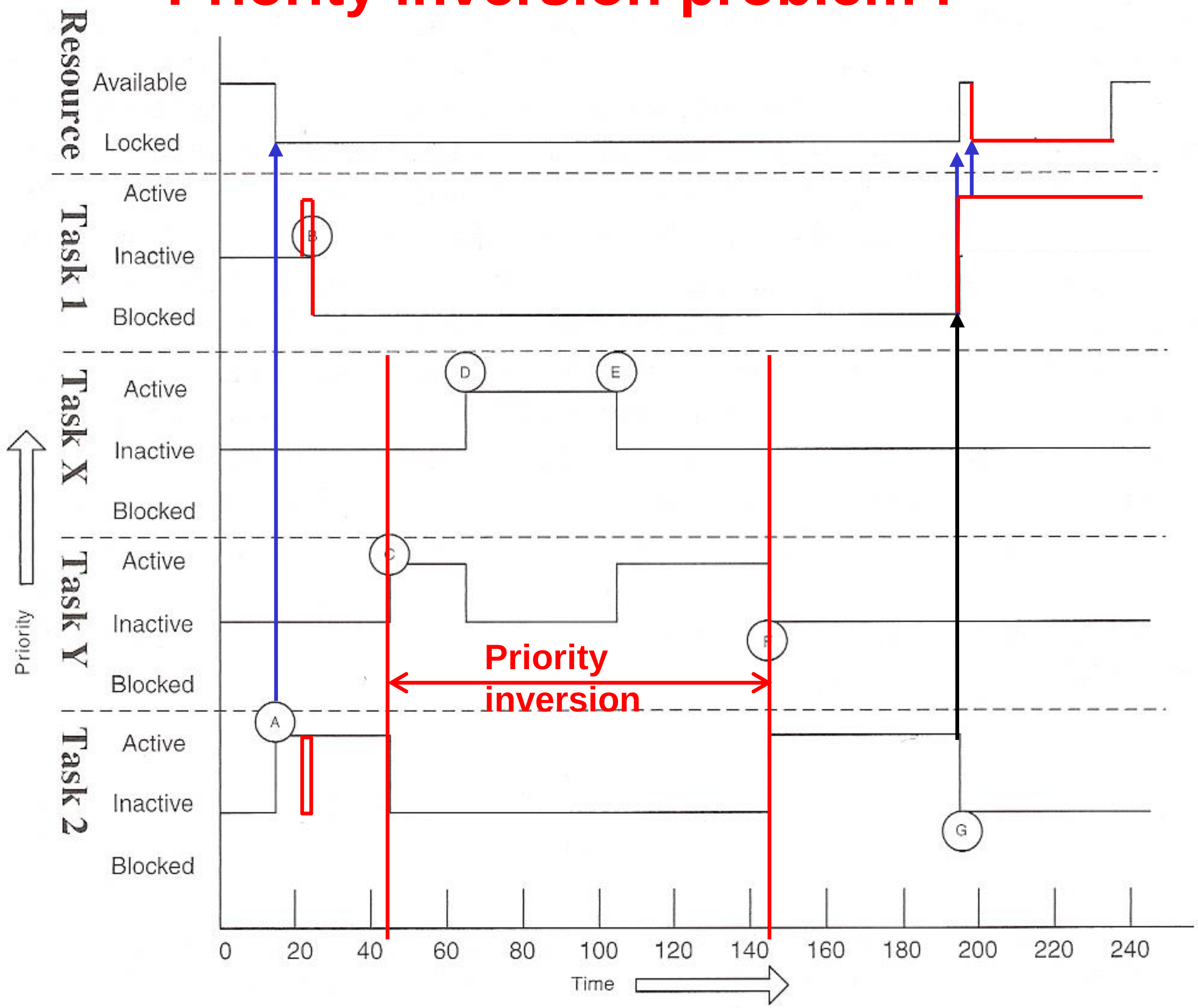


Using priority based preemptive scheduling strategy

AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Unbounded Task Blocking (2)

Priority inversion problem !



NB! Corrected figure from BPD book

Unbounded Task Blocking (3)

Legend:
Priorities: Task 1 > Task X > Task Y > Task 2
A: Task 2 is ready to run and starts.
B: Task 1 is ready to run but needs the Resource. It is blocked and must allow Task 2 to complete.
C: Task Y, which is a higher priority than Task 2, is ready to run. Since it doesn't need the resource, it preempts Task 2. Task 1 is now effectively blocked by both Task 2 and Task Y.
D: Task X, which is a higher priority than Task Y, is ready to run. Since it doesn't need the resource, it preempts Task Y. Task 1 is now effectively blocked by 3 tasks.
E: Task X completes, allowing Task Y to resume.
F: Task Y completes, allowing Task 2 to resume.
G: Task 2 (finally) completes and releases the resource, allowing Task 1 to access the resource.

This problem is called unbounded priority inversion

Solution – one of these 3 patterns:
1. Critical Section Pattern
2. Priority Inheritance Pattern
3. Highest Locker Pattern

Deadlock Prevention

A deadlock needs the following four conditions to occur:

1. Mutual exclusion (locking) of resources
2. Resources are held (locked) while others are waited for
3. Preemption while holding resources is permitted
4. A circular wait condition exists

   (P1 waits on P2, which waits on P3, which waits on P1)

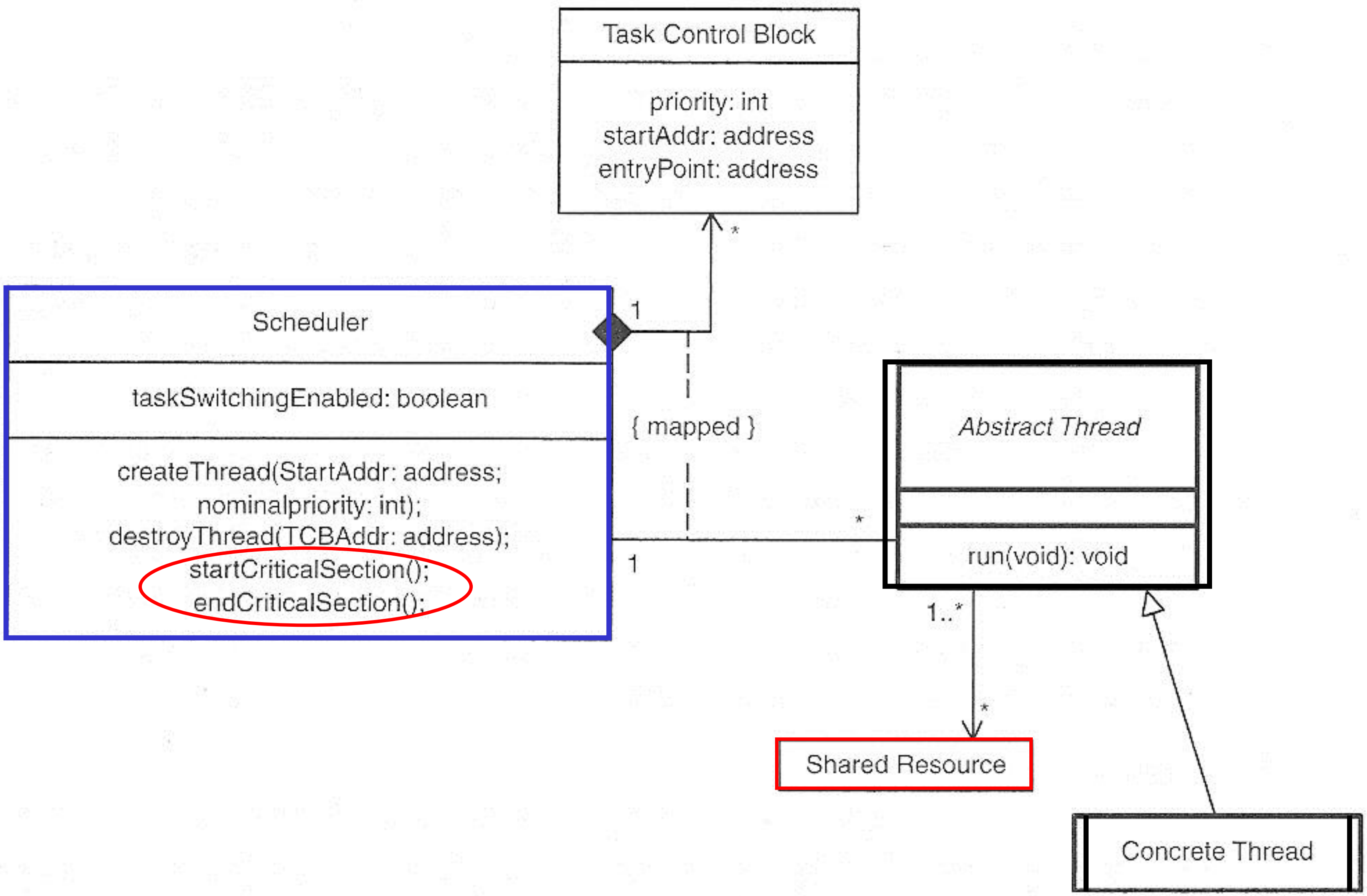
Solution - one of these 3 patterns:

4. Priority Ceiling Pattern breaks cond. 4.
5. Simultaneous Locking Pattern breaks cond. 2.
6. Ordered Locking Pattern breaks cond. 4.

1. Critical Section Pattern (~ Monitor Pattern)

The Critical Section Pattern
locks the Scheduler whenever a **resource** is accessed to prevent another task from simultaneously accessing it.

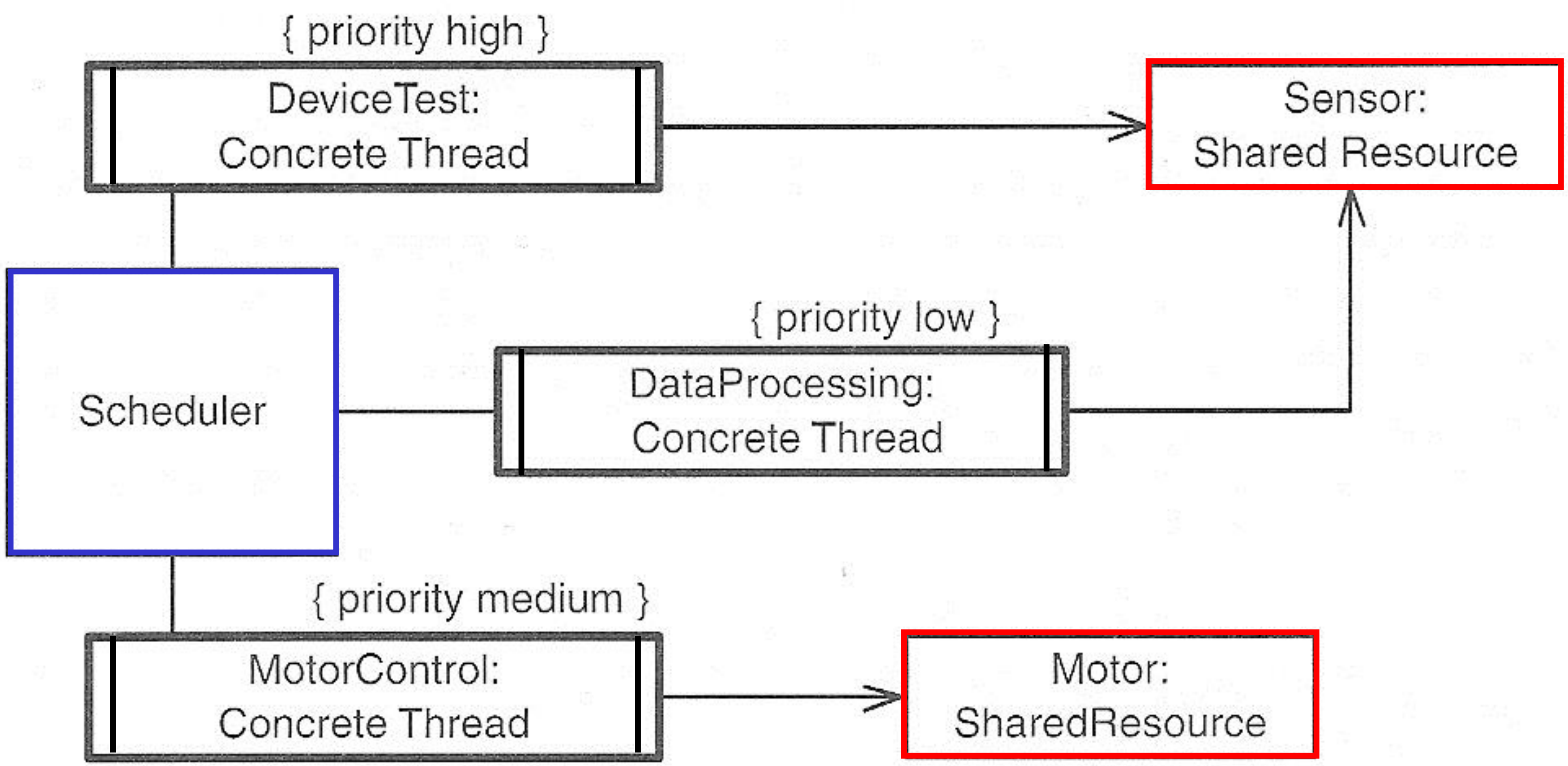
Critical Section Pattern Structure



Critical Section Pattern Strategy

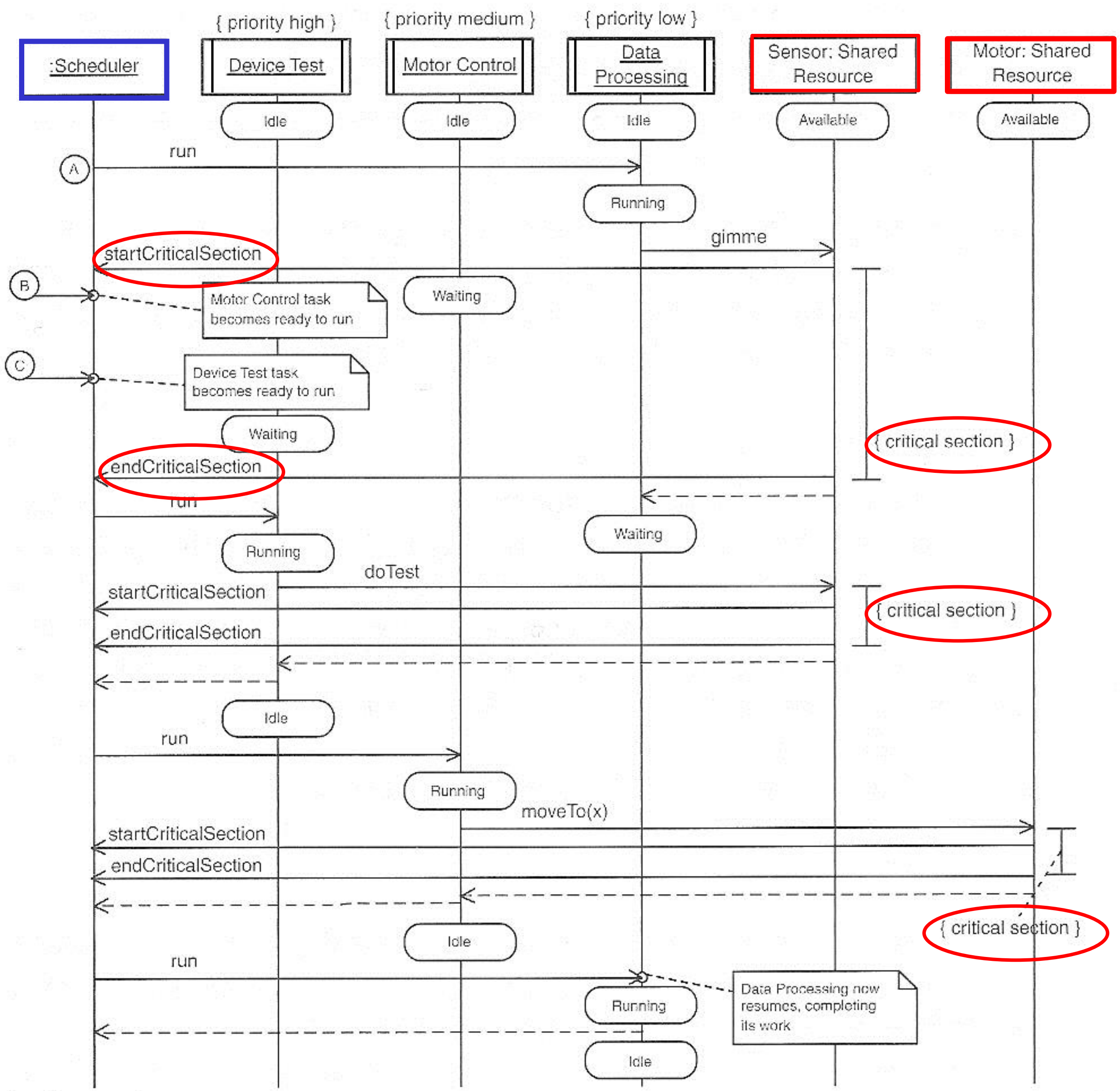
- Each operation in a shared resource starts and ends with:
  - scheduler->startCriticalSection()
  - scheduler->endCriticalSection()
- Effect:
  - Makes the current thread the highest priority as no other threads can preempt the scheduler during a critical section.

Critical Section Pattern Example (1)



a. Structure

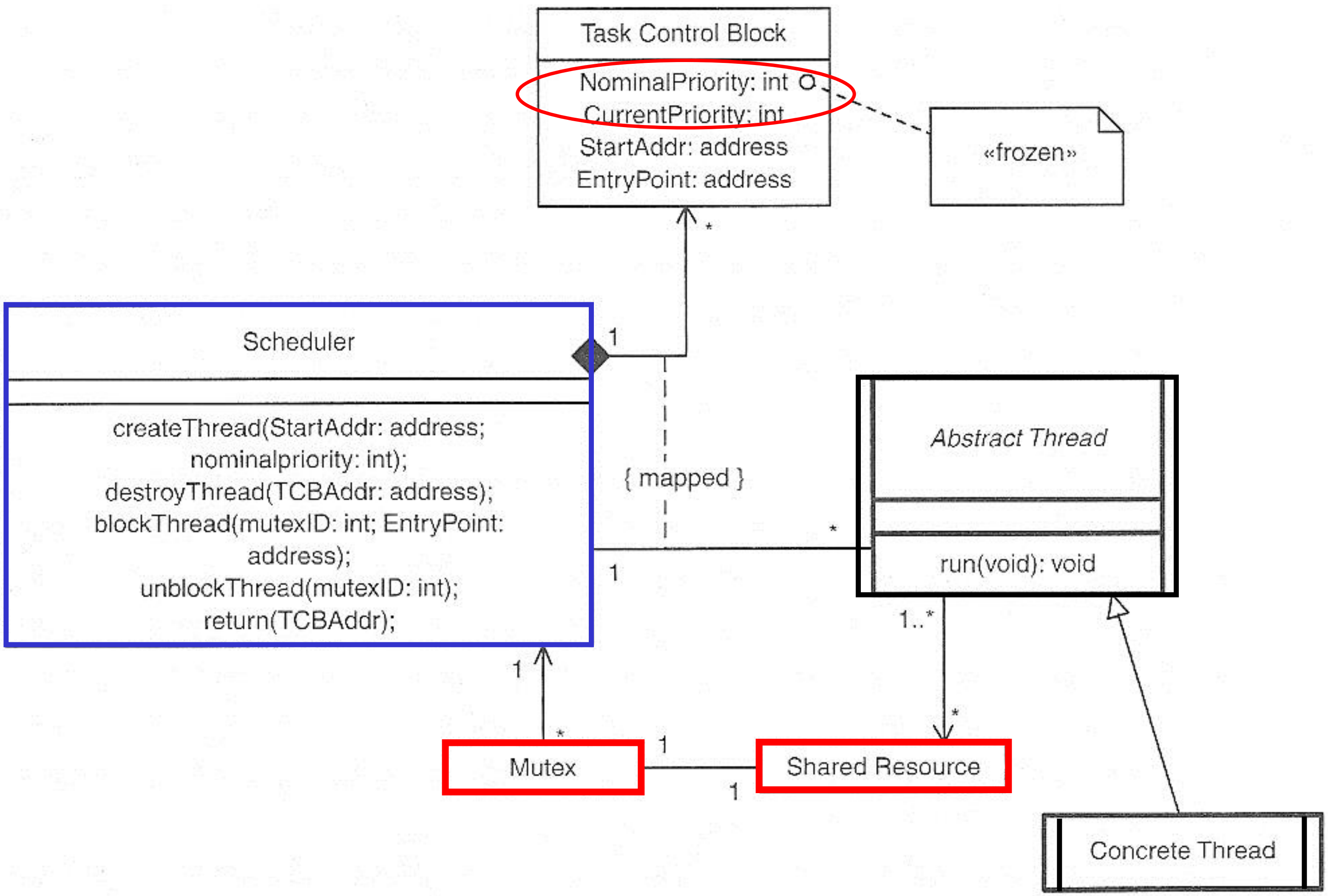
Critical Section Pattern Example (2)



2. Priority Inheritance Pattern

The Priority Inheritance Pattern **reduces priority inversion** by manipulating the executing priorities of tasks that locks shared resources.

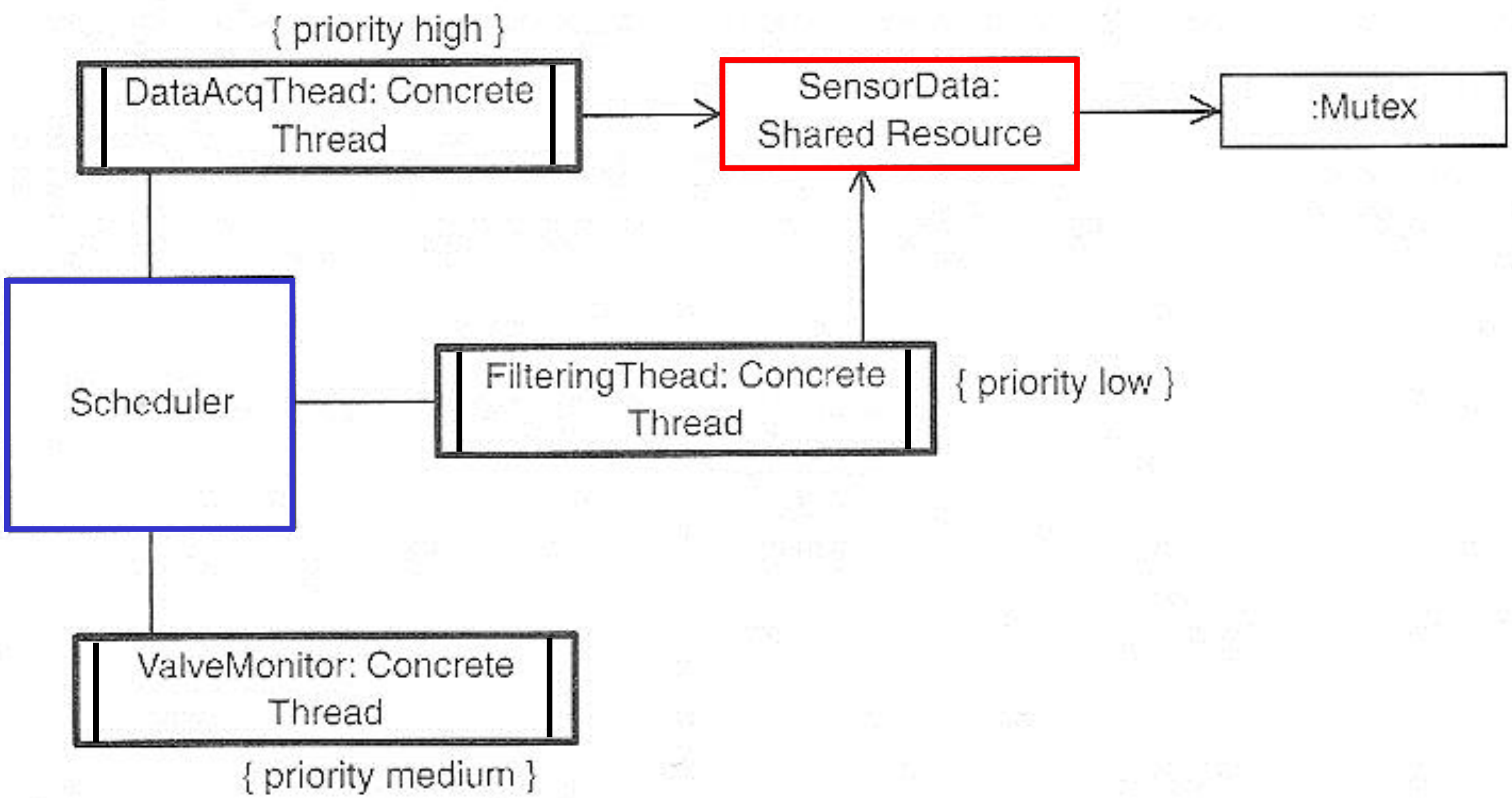
Priority Inheritance Pattern Structure



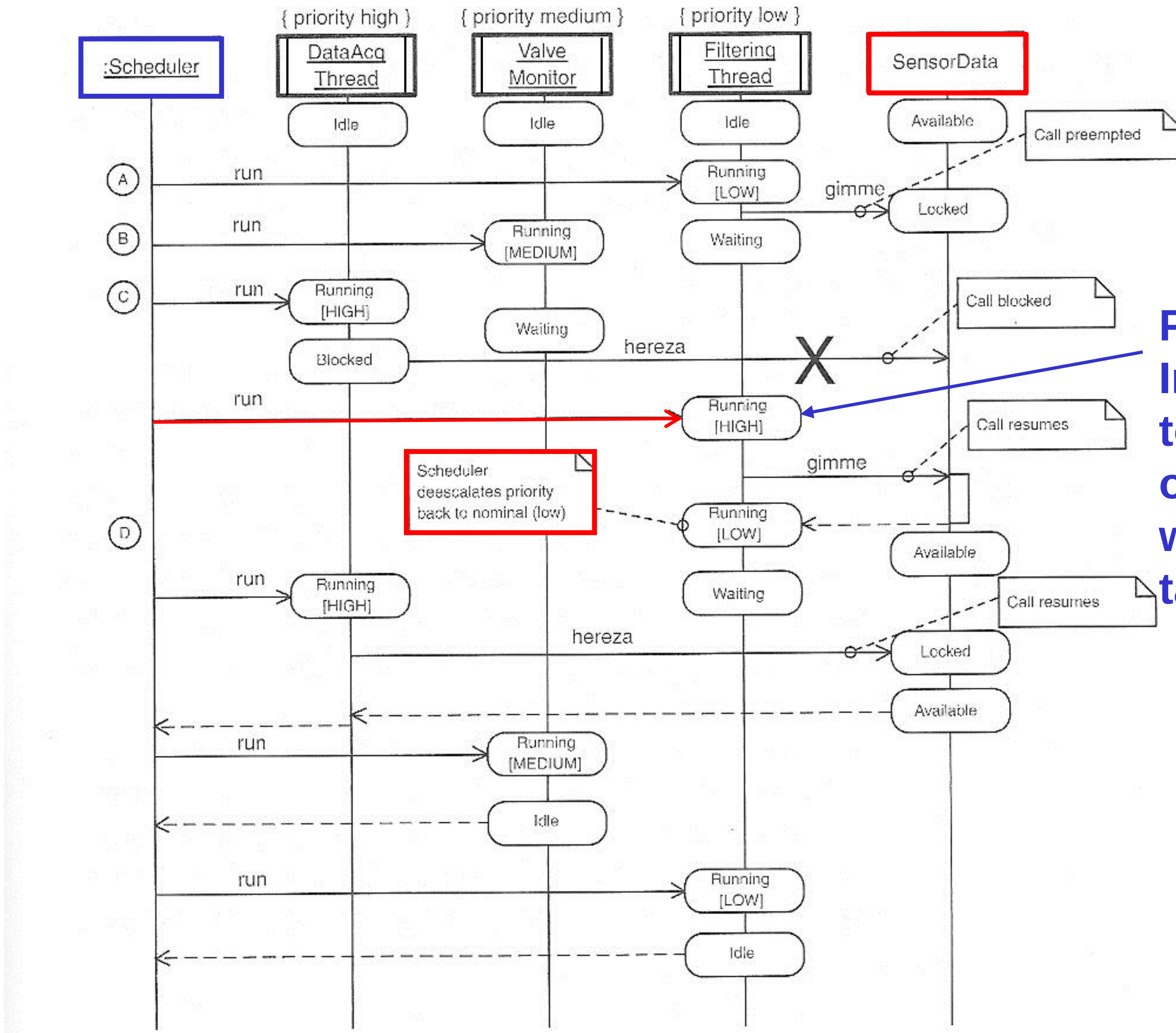
Priority Inheritance Pattern Strategy

- **Elevates the priority** of thread, who locks a shared resource, to the priority of a higher priority thread, when it is being blocked, trying to access the same resource.
- **Lowers the priority** of the thread, back to normal, when it unlocks the resource

Priority Inheritance Pattern Example (1)



Priority Inheritance Pattern Example (2)



Priority Inheritance to priority of highest waiting task

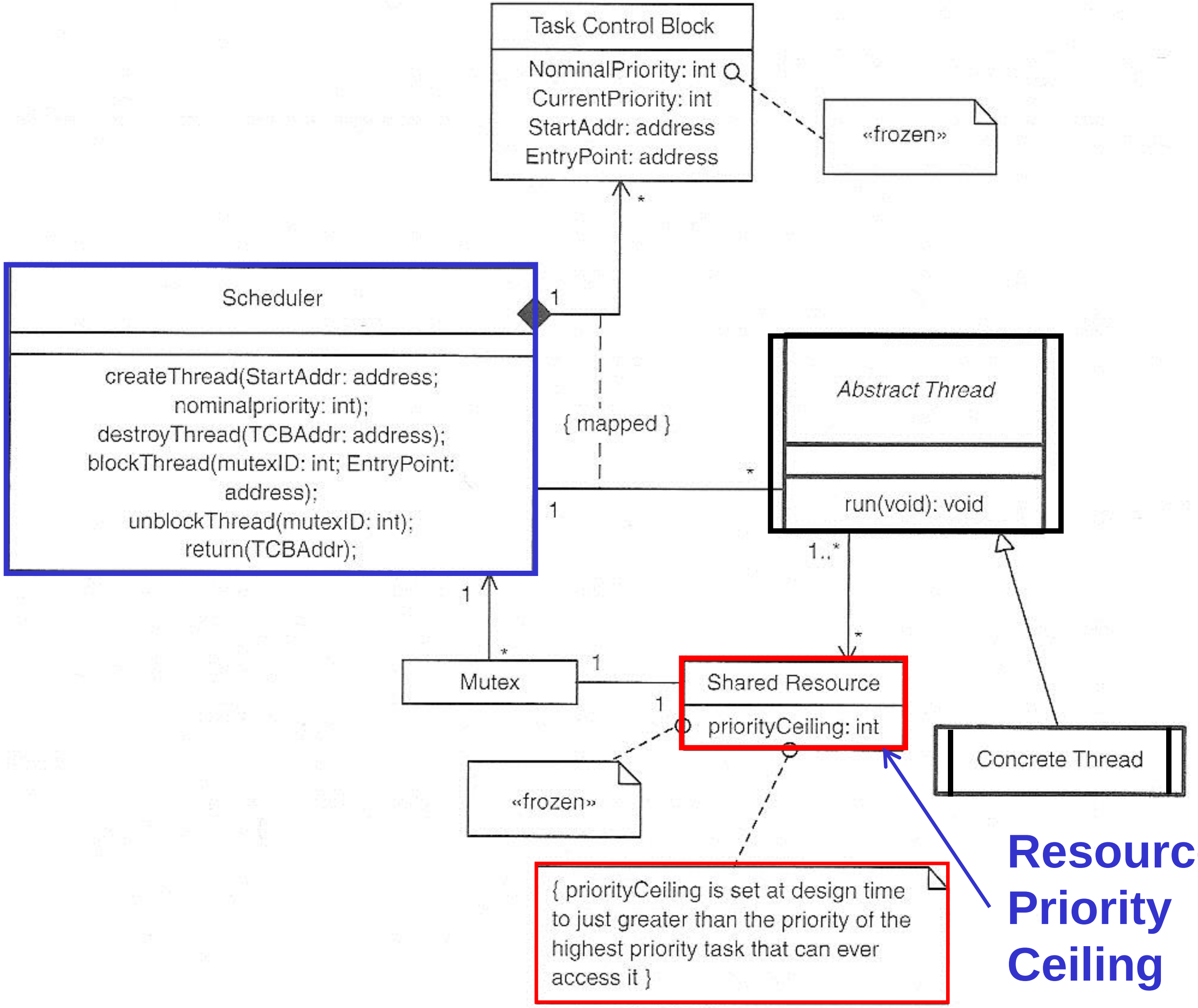
3. Highest Locker Pattern

The Highest Locker Pattern reduces priority inversion by manipulating the executing priorities of tasks that locks shared resources.

The Highest Locker Pattern defines a ***priority ceiling*** with each resource.

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

Highest Locker Pattern Structure

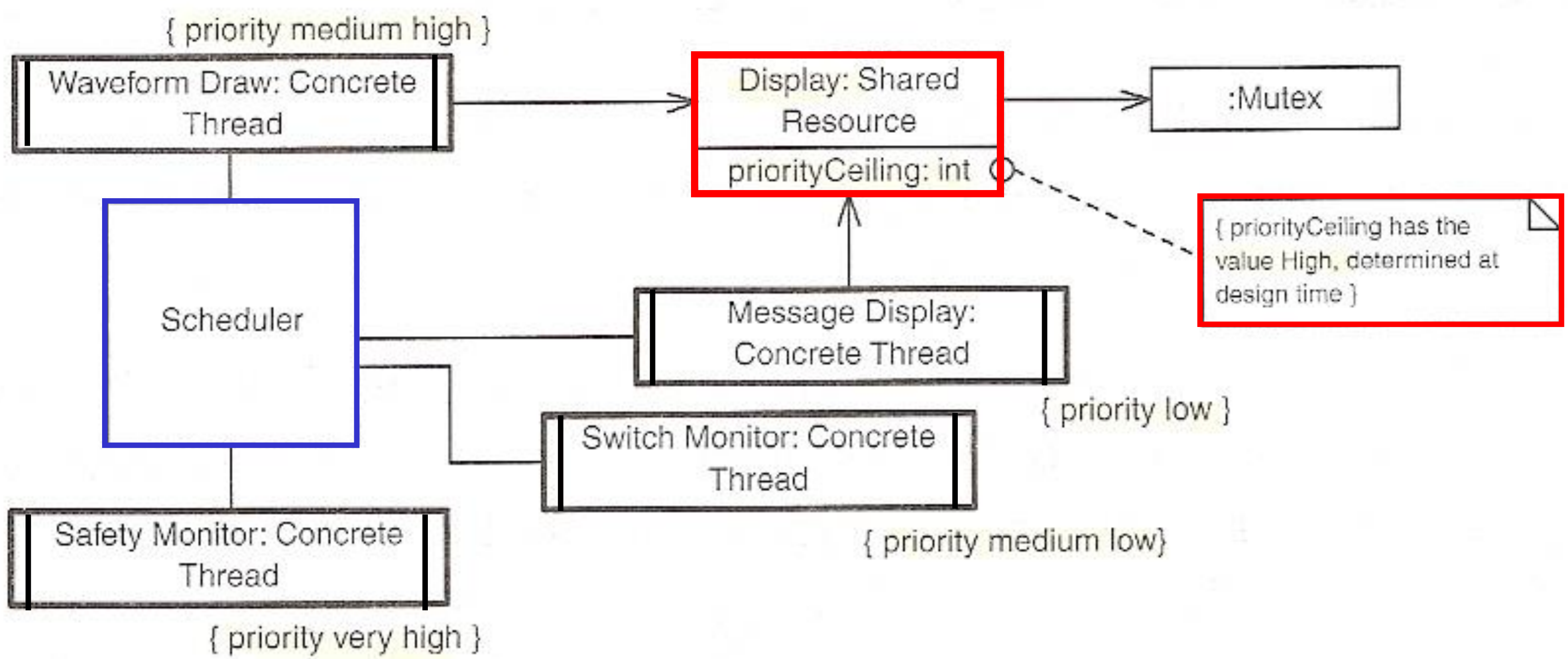


Resource Priority Ceiling

Highest Locker Pattern Strategy

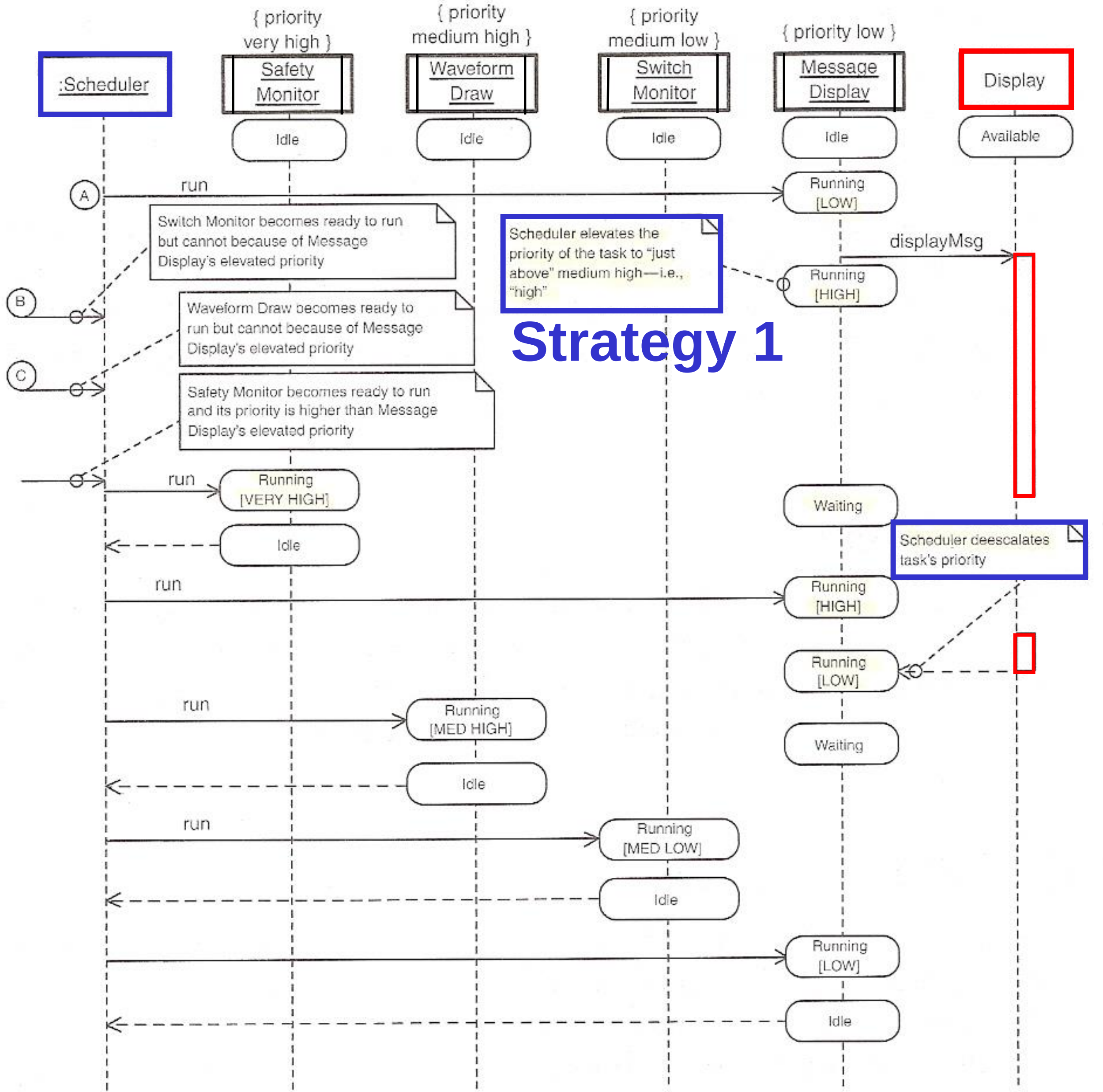
- Each shared resource has a **priority ceiling**= one greater than the priority of the highest-priority client thread using the resource
- **Strategy 1:**
  - When the resource is locked, the priority of the locking thread is immediately elevated to the priority ceiling
- **Strategy 2:**
  - Delays the priority elevation until another higher priority thread tries to lock the resource
- **Lowers the priority** of the thread, back to normal, when it unlocks the resource

Highest Locker Pattern Example (1)



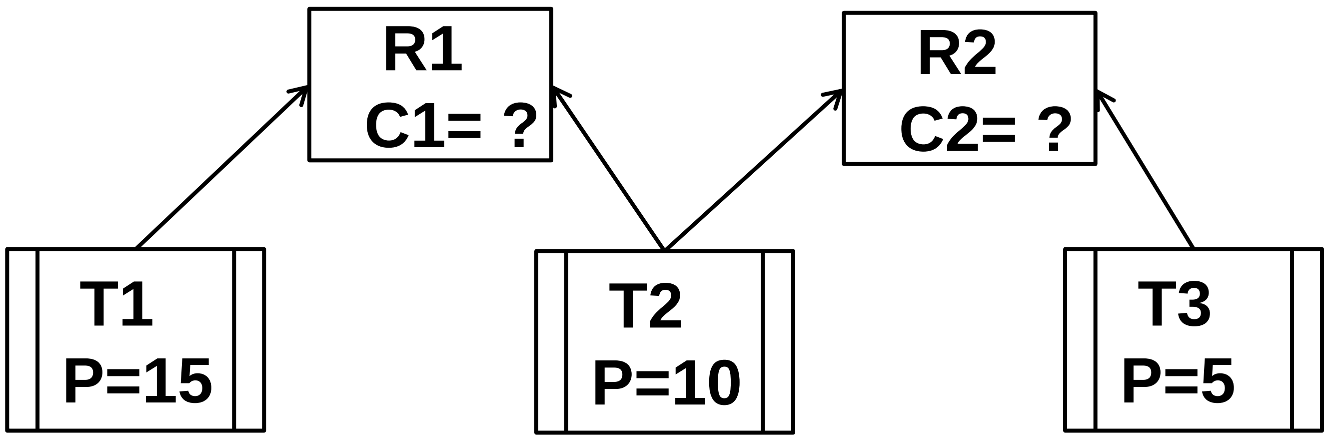
AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

Highest Locker Pattern Example (2)



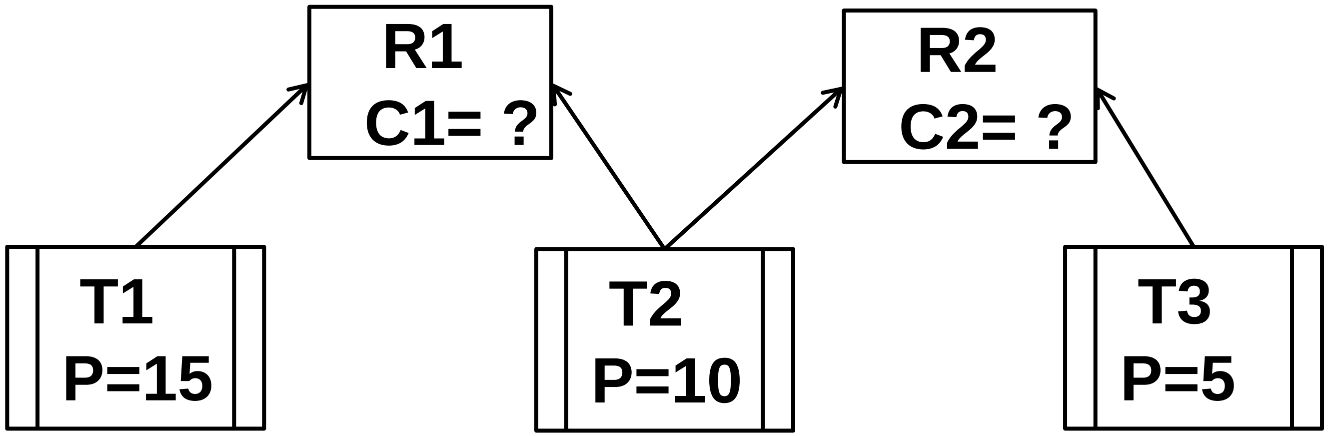
Strategy 1

Class Exercise 1.



1. Determine priority ceilings C1 and C2.
2. Draw a timeline as fig. 7.2 for the following situation assuming that task priority is changed immediately to the priority ceiling, when accessing a shared resource (strategy 1).
t0: T3 ready, t2: T3 locks R2 for 10 time units, t5: T2 ready locks R1 for 7 time units, t10: T1 ready, locks R1 for 3 time units.
3. Determine the blocking time for each task caused by the other tasks.
(NB! High number for P = high priority)

Class Exercise 2.



1. Determine priority ceilings C1 and C2.
2. Draw a timeline as fig. 7.2 for the following situation assuming that task priority change to the priority ceiling is delayed until another task is requesting the locked ressource (strategy 2).
t0: T3 ready, t2: T3 locks R2 for 10 time units, t5: T2 ready locks R1 for 7 time units, t10: T1 ready, locks R1 for 3 time units.
3. Determine the blocking time for each task caused by the other tasks.
4. Compare with class exercise 1.

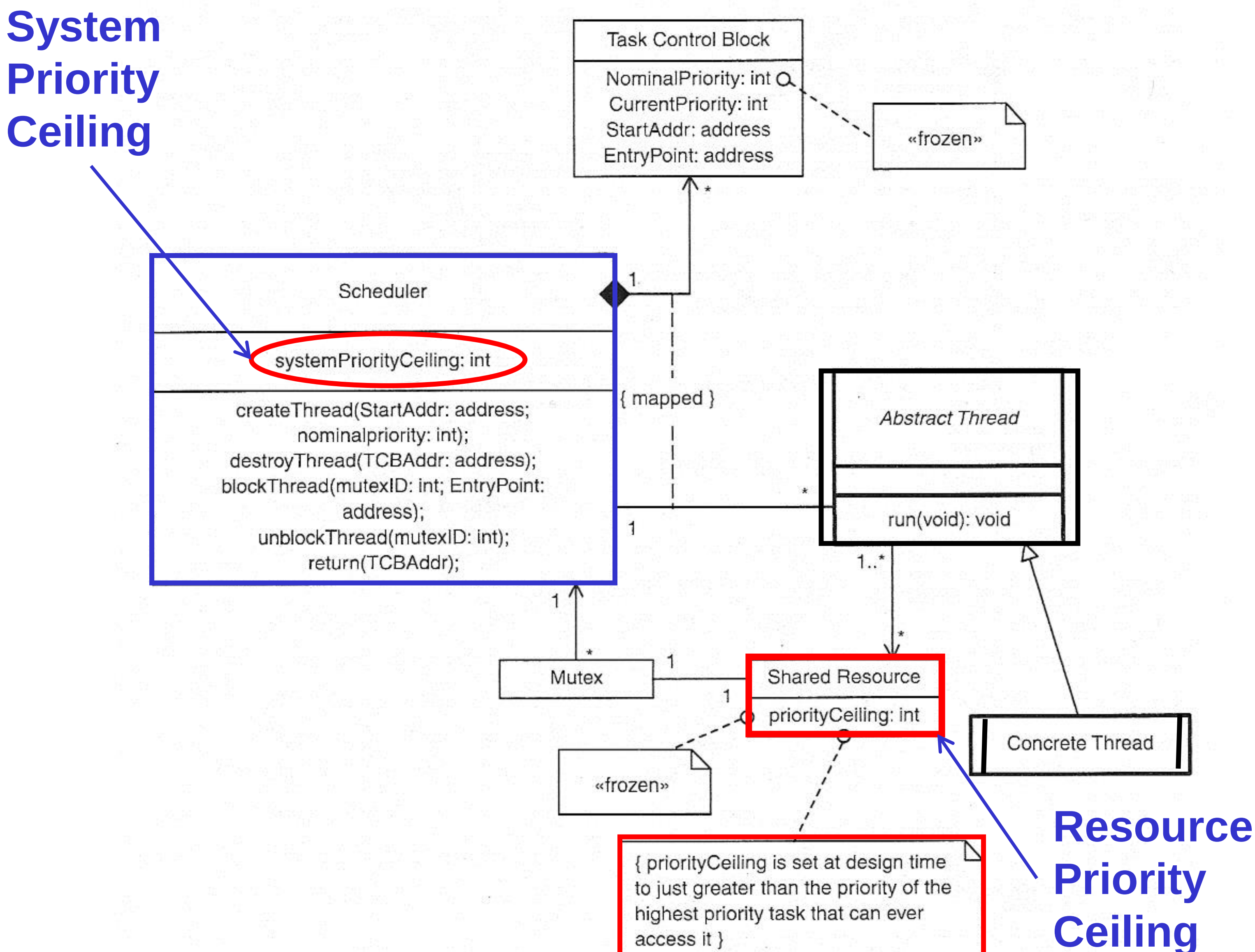
4. Priority Ceiling Pattern

The Priority Ceiling Pattern or Priority Ceiling Protocol (PCP) addresses **both bounding priority inversion** and **removal of deadlock**.

Operates both with a **System** and a **Resource Priority Ceiling**

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

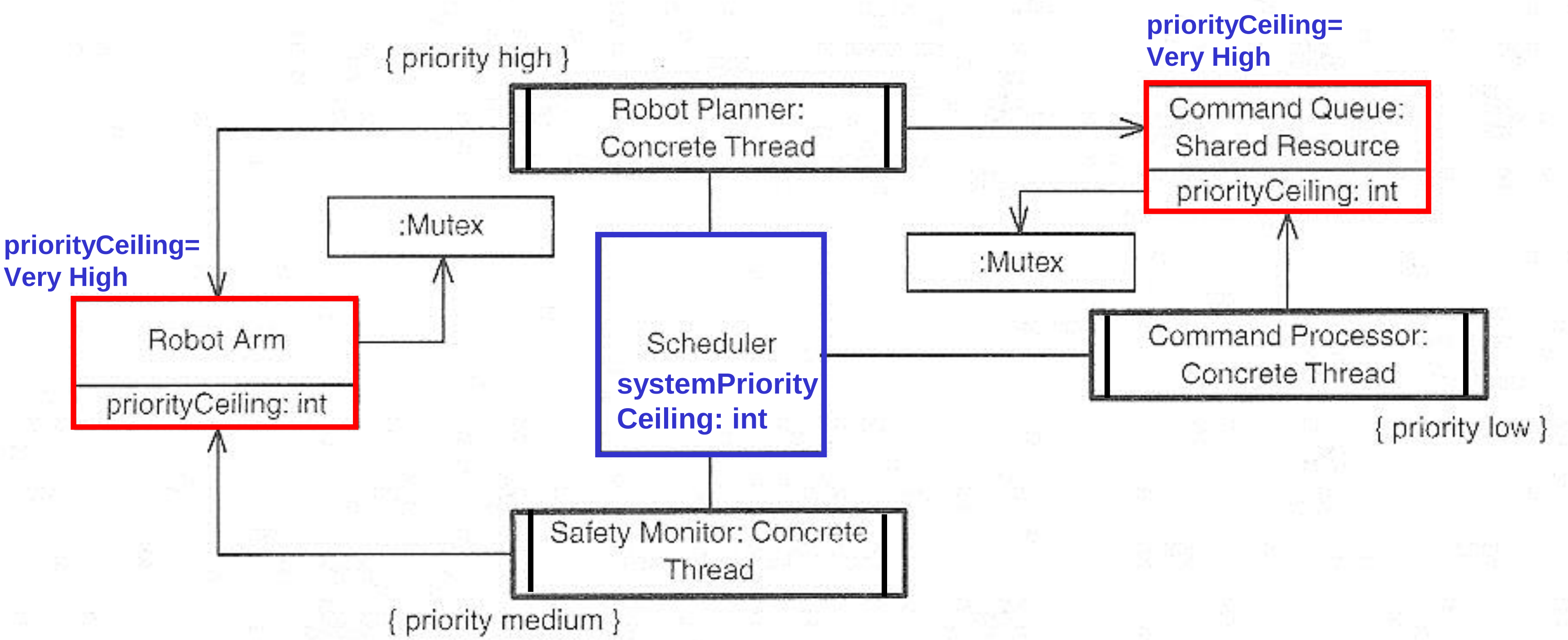
Priority Ceiling Pattern Structure



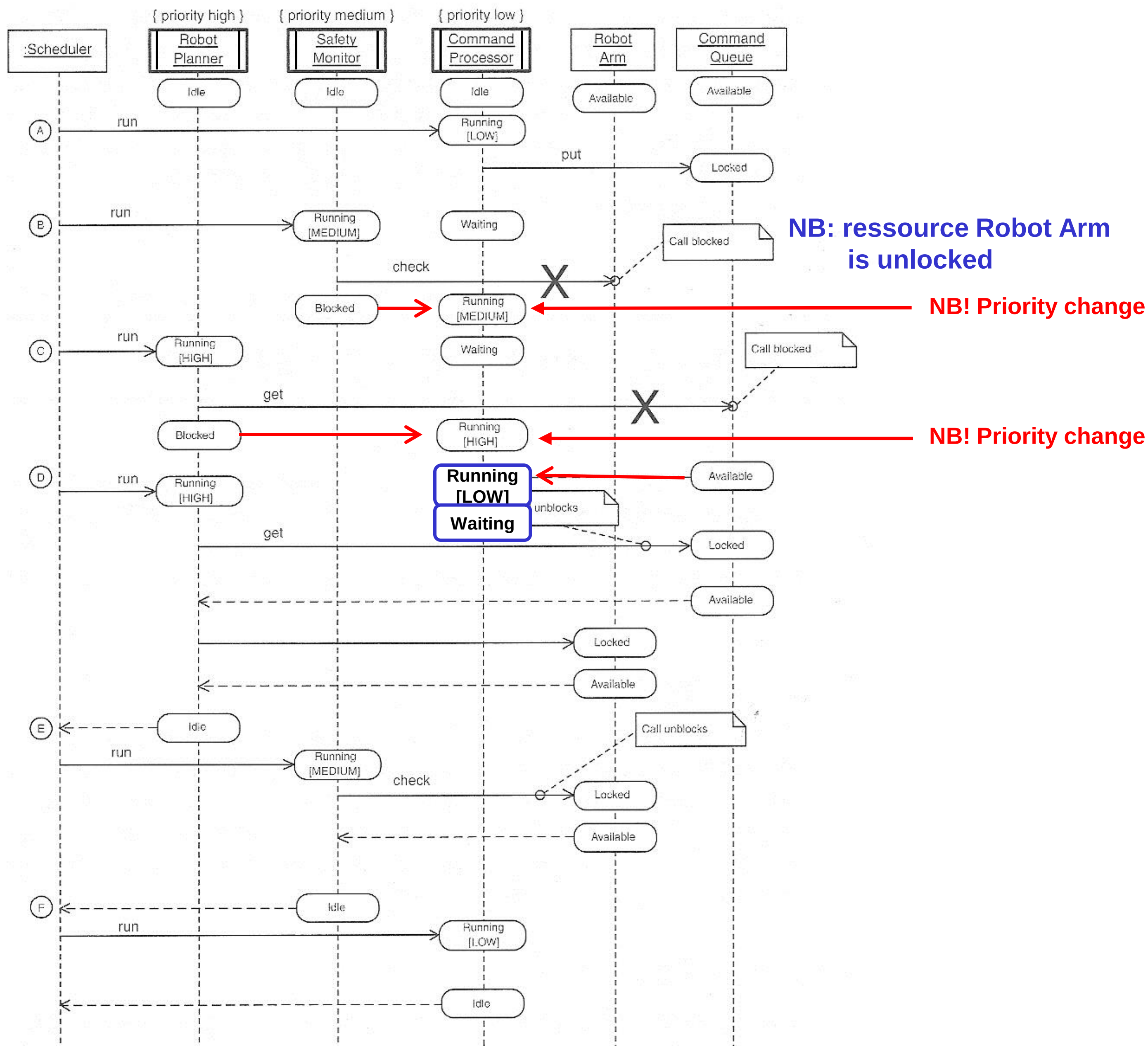
Priority Ceiling Pattern Strategy

1. **System Priority Ceiling**= priority ceiling of highest currently locked resource
2. A thread can only lock a resource if its **priority ceiling > current System Priority Ceiling**.
3. A thread having a locked resource is elevated to the priority of a blocked thread, when a higher priority thread runs and tries to lock the same or another shared resource.
4. Releasing a lock on a shared resource will reestablish the normal priority for the thread

Priority Ceiling Pattern Example (1)



Priority Ceiling Pattern Example (2)

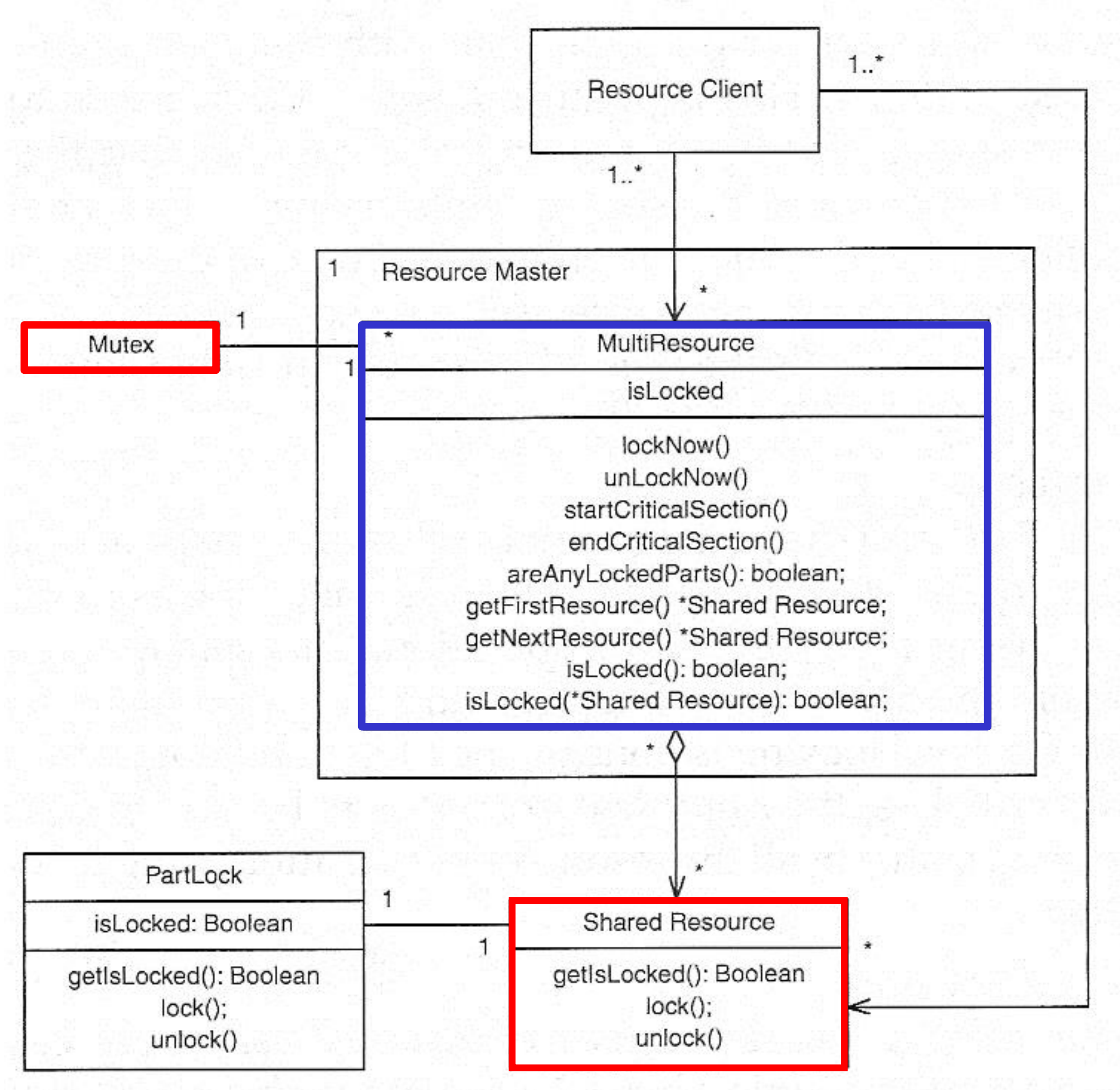


5. Simultaneous Locking Pattern

The Simultaneous Locking Pattern is a pattern solely concerned with **deadlock avoidance**. **Either all resources needed are locked at once or none at all.**

The Simultaneous Locking patterns breaks condition 2 – not allowing any task to lock resources while waiting for others to be free.

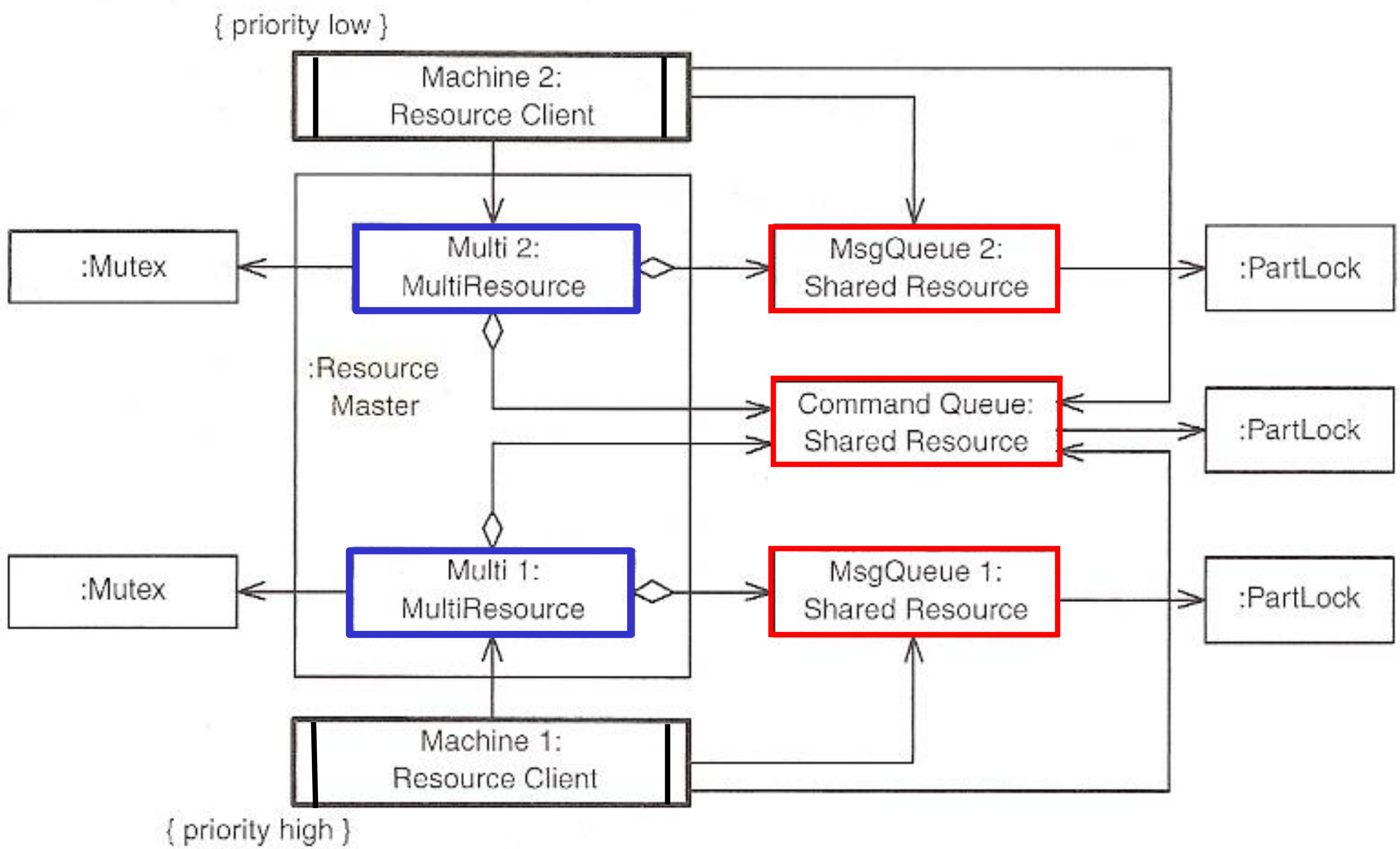
Simultaneous Locking Pattern Structure



Simultaneous Locking Pattern Strategy

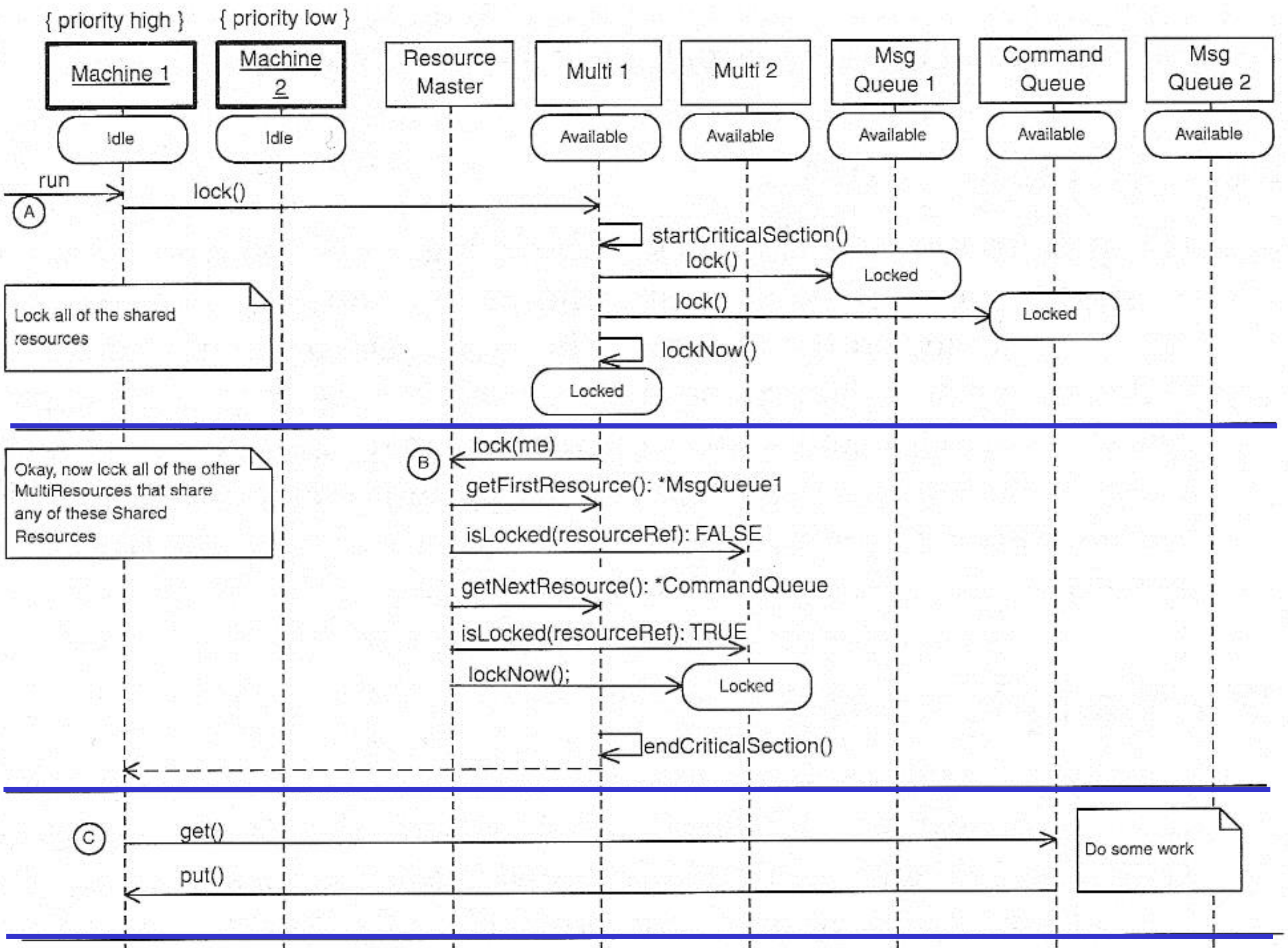
- Each MultiResource has a single Mutex that locks only when the entire aggregated Shared Resources are locked
- Locking and unlocking in MultiResource must be done as a critical section

Simultaneous Locking Pattern Example (1)

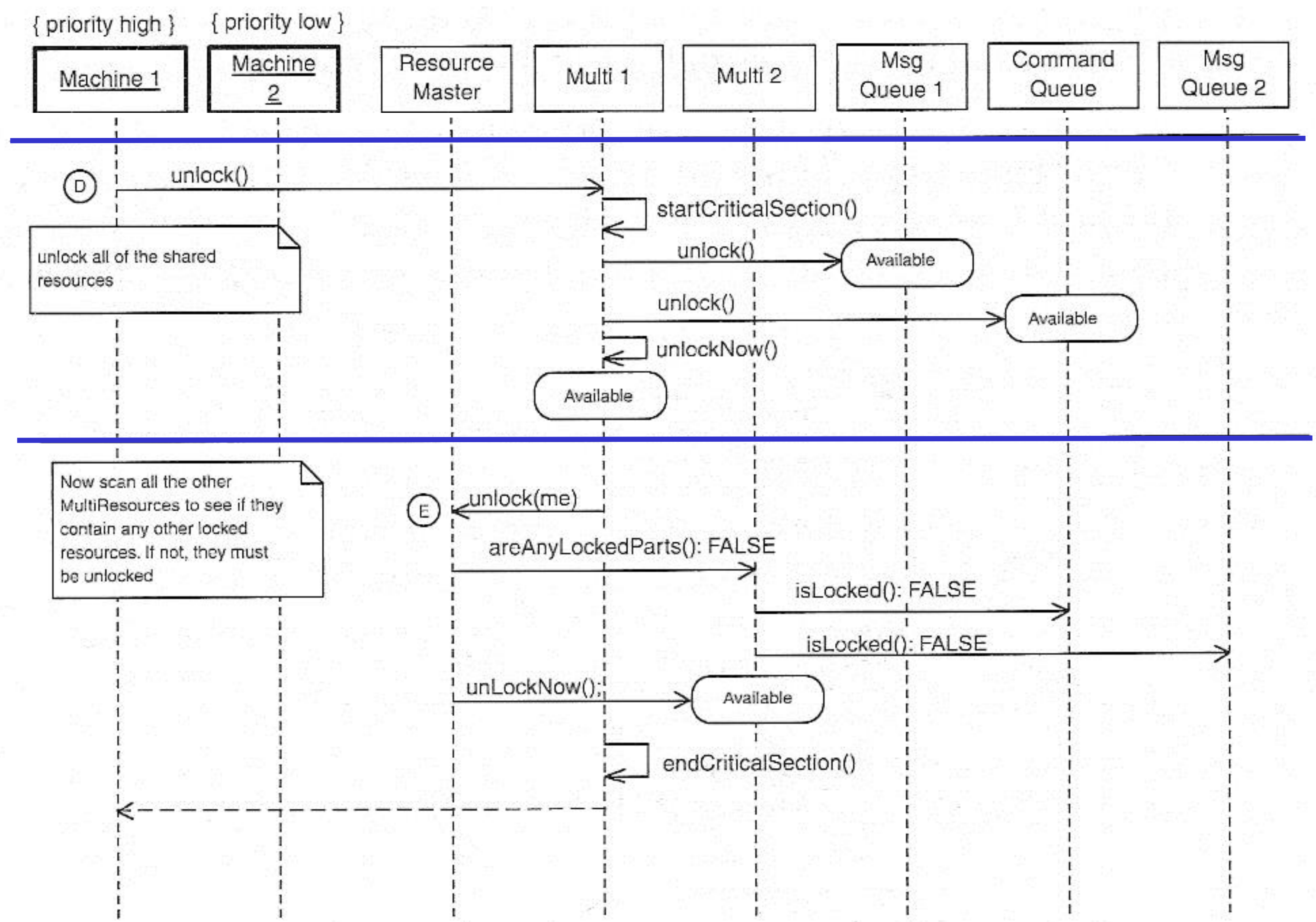


AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

Simultaneous Locking Pattern Example (2)



Simultaneous Locking Pattern Example (3)

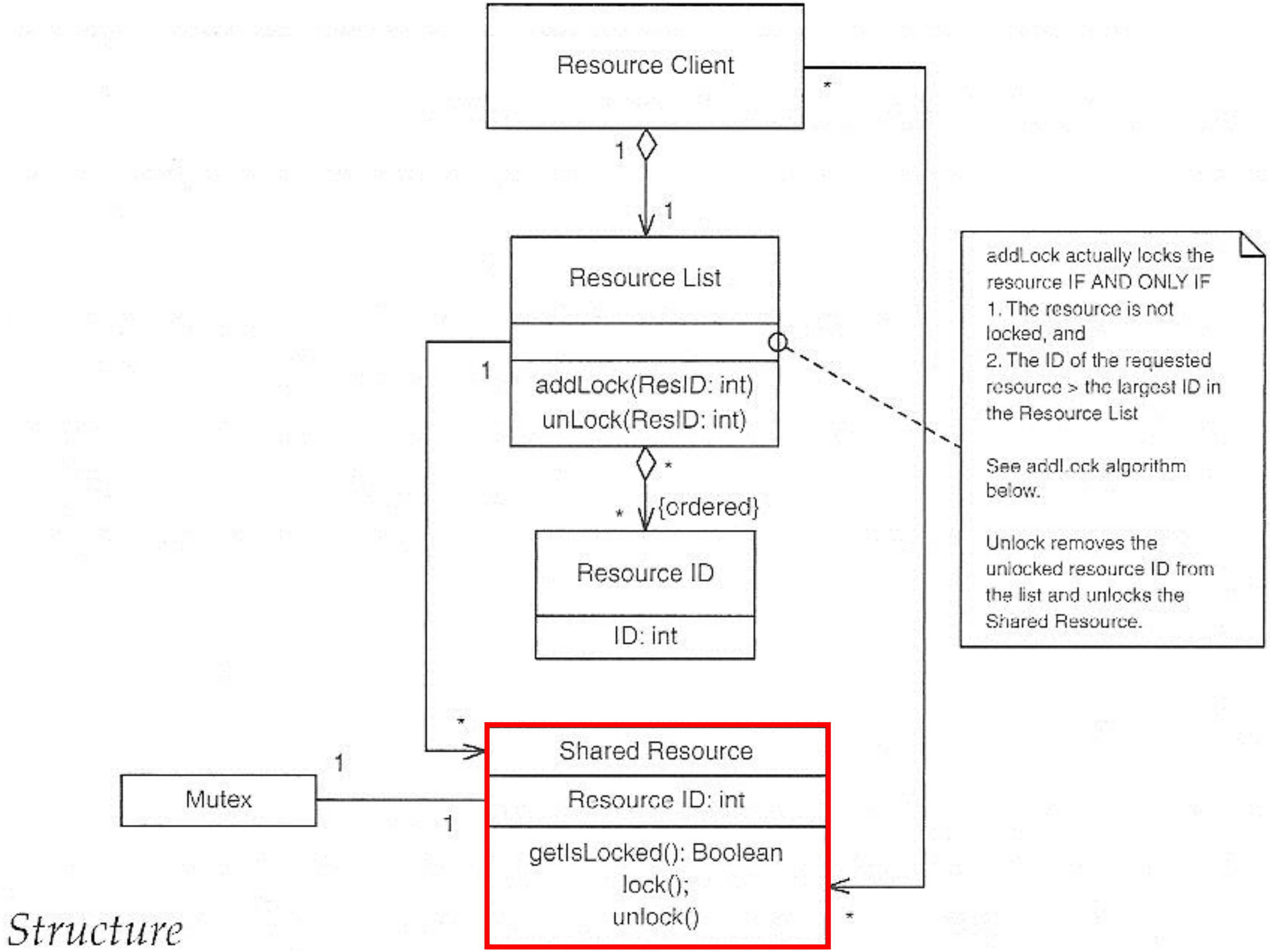


6. Ordered Locking Pattern

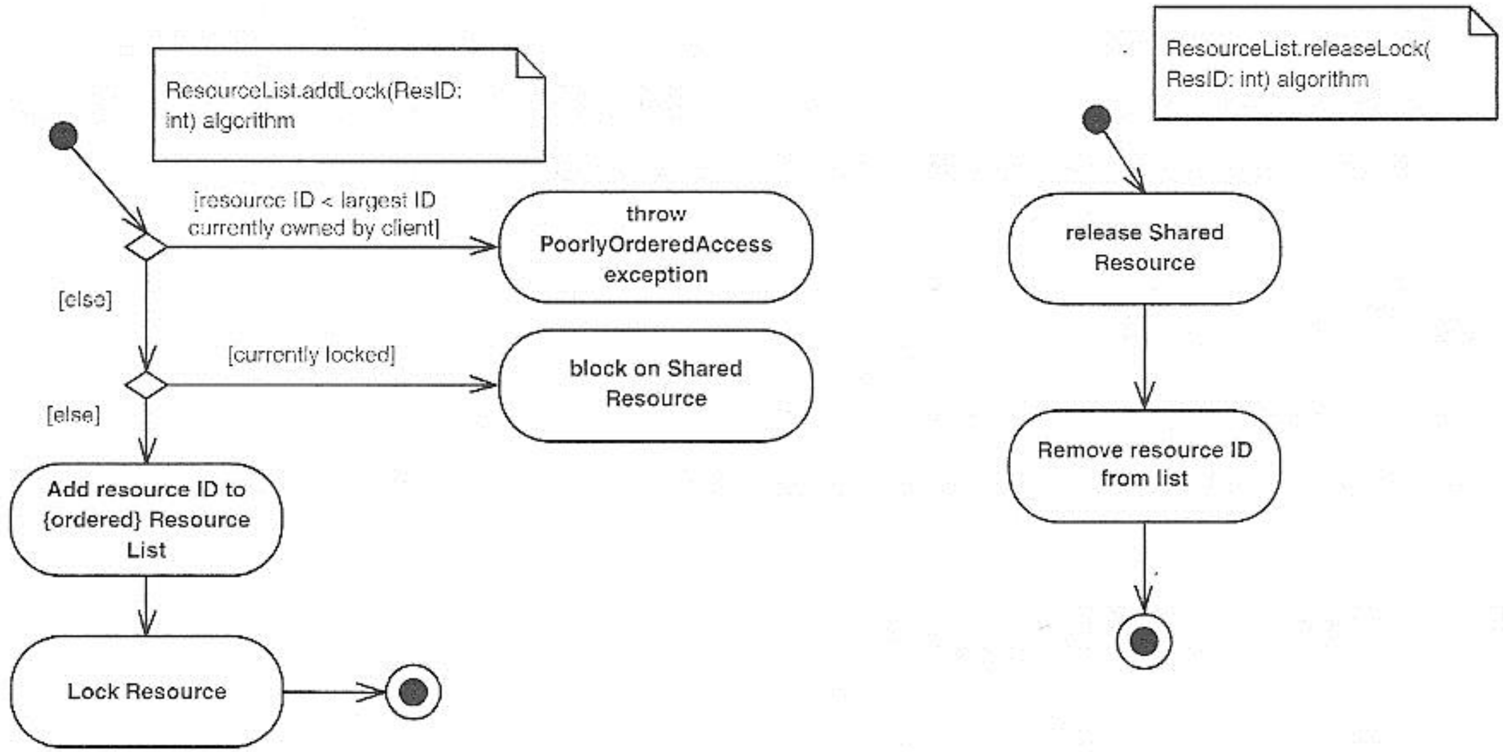
The Ordered Locking Pattern is another way to ensure that **deadlock cannot occur**.

It does this by ordering the resources and requiring that they always be accessed by any client in that specified order.

Ordered Locking Pattern Structure

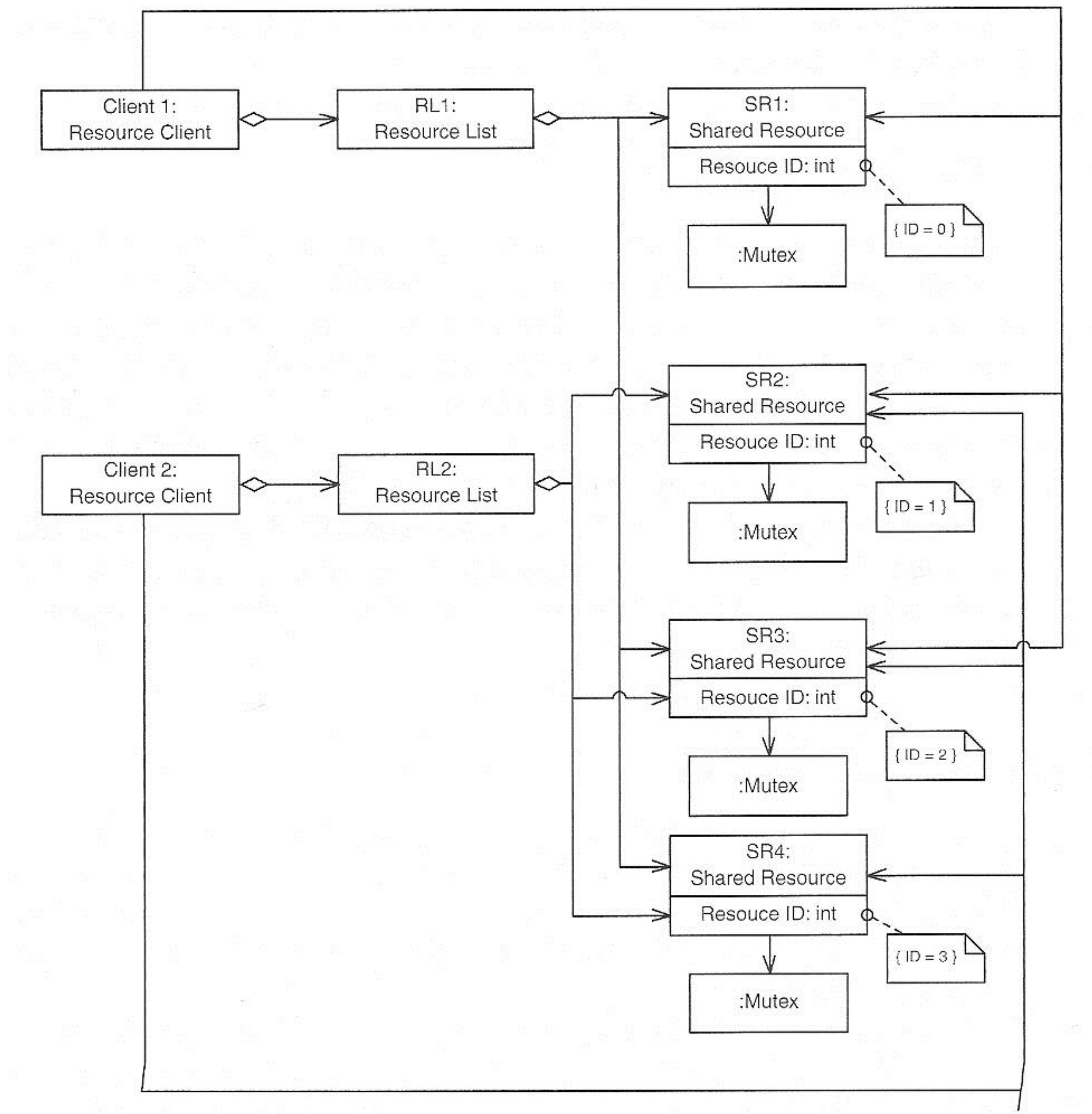


Ordered Locking Access Algorithms



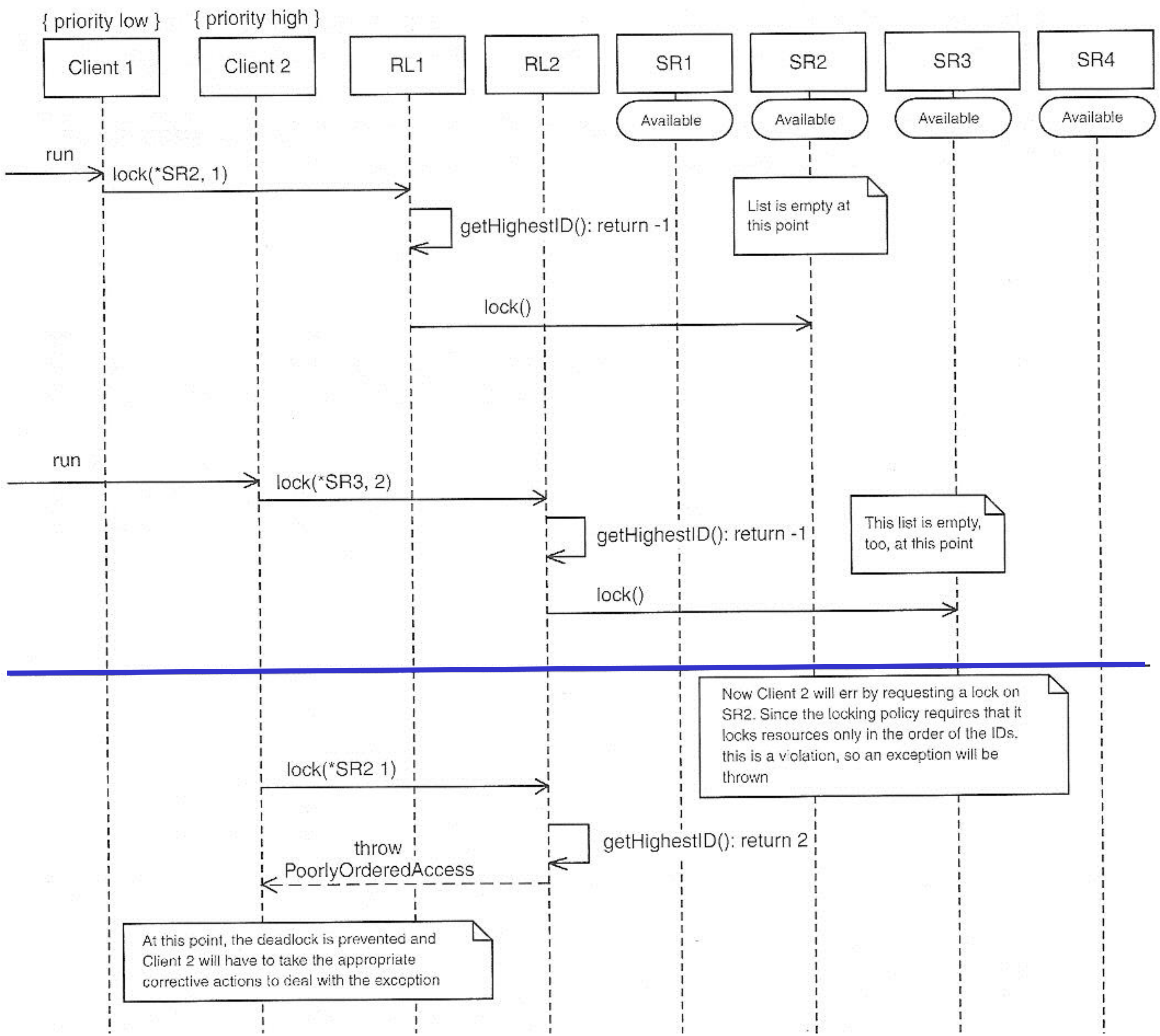
b. Access Algorithms

Ordered Locking Pattern Example (1)



AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

Ordered Locking Pattern Example (2)



Summary

1. Critical Section Pattern
2. Priority Inheritance Pattern
3. Highest Locker Pattern
4. Priority Ceiling Pattern
5. Simultaneous Locking Pattern
6. Ordered Locking Pattern

1..6: Solution to resource protection
1,2,3,4: Solution to unbounded priority inversion
4,5,6: Prevents deadlock