19Z604 Embedded Systems

Dr.N.Arulanand,
Professor,
Dept of CSE,
PSG College of Technology

Resource Sharing

- Resource Sharing among real-time tasks
- CPU is a serially reusable resource
  - Can be used by one task at a time
  - The task can be preempted at any time without affecting correctness
- Other resources:
  - files, data structures, Memory, devices,
  - These resources need to be used in preemptable mode.

Resource Sharing

- Critical Section

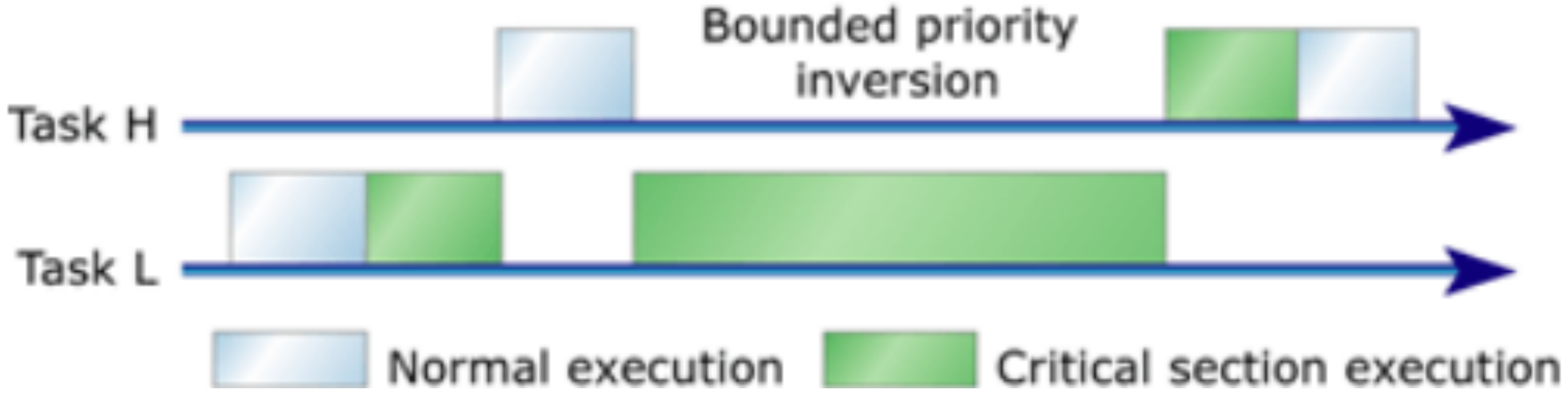
A part of the section in while a shared non-preemptable resource is accessed

- Traditional solution in Operating System to execute critical section
  - Semaphores
- In Real-time systems, the solution doesn’t work. It leads to
  - Priority Inversion
  - Unbounded Priority Inversion

Priority Inversion

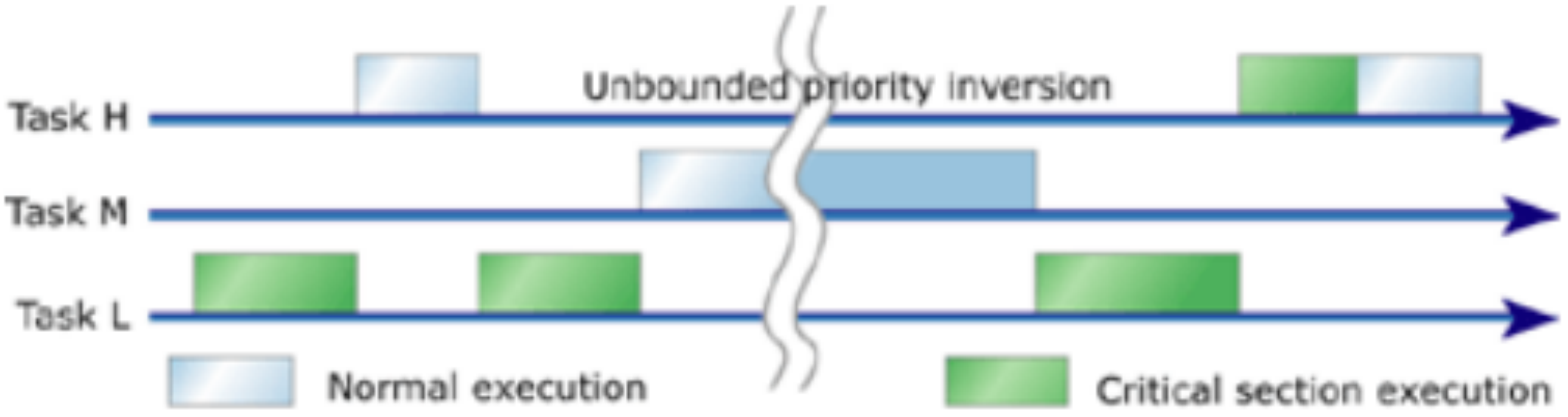
- Priority Inversion
  - **Priority inversion** is a situation in which a low-**priority** task executes while a higher **priority** task waits on it due to resource contentions
- Consequences in Real time system
  - If a task cannot be preempted (a low priority task) while computing a critical section
  - The higher priority task keeps waiting
  - Task Scheduling Algorithm: Low Priority task should yield to Higher priority task (ex: RMA) .
  - Basic assumption in Real Time Task Scheduling is Higher Priority will execute first....

Bounded Priority Inversion



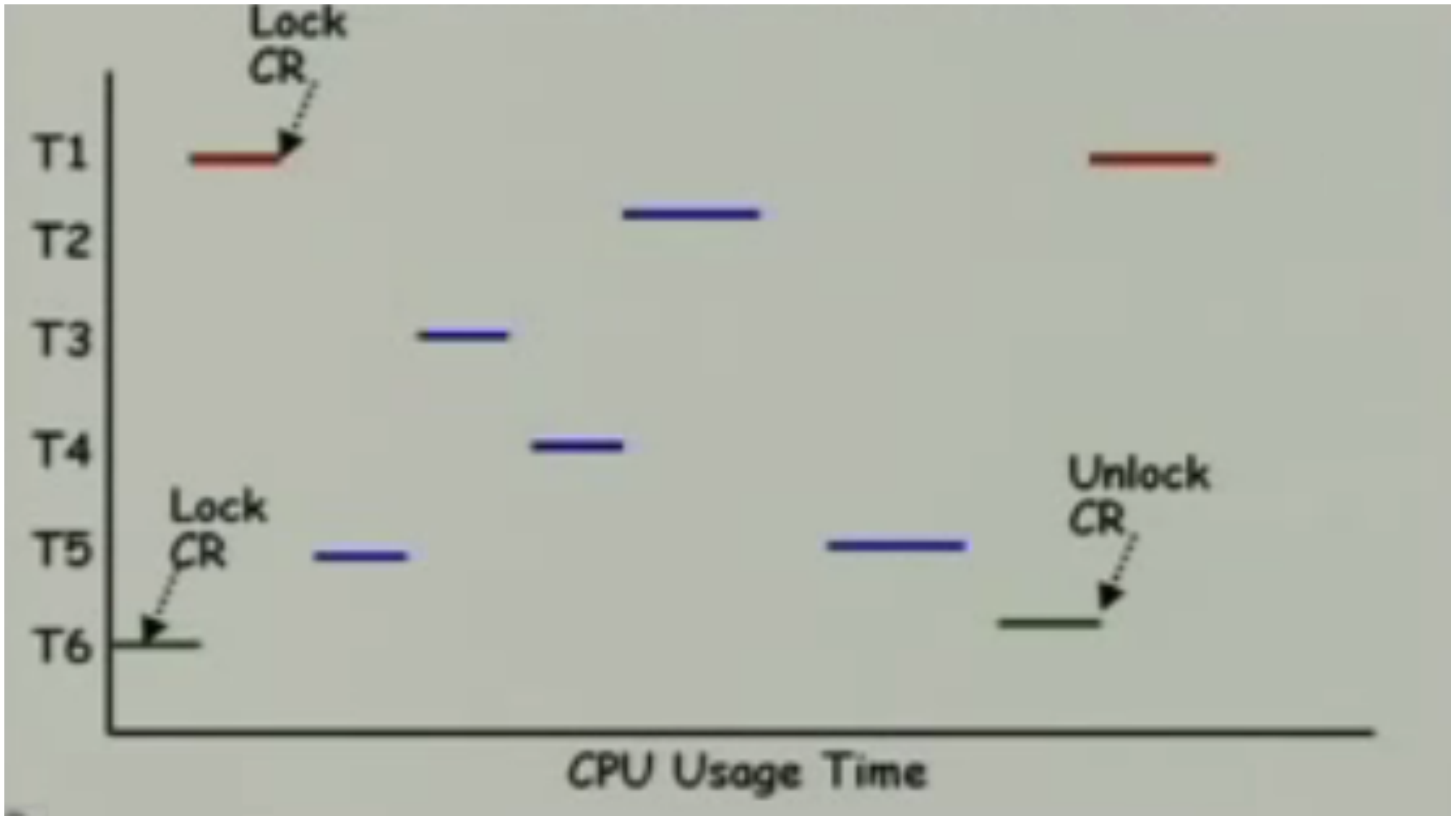
Unbounded Priority Inversion

- A complex problem than bounded priority inversion
  - Situation: Low priority task is holding a resource.
  - After some time, High priority task executes and waiting for T*L* to complete
  - Intermediate tasks, which doesn’t need to resource preempts to low priority task.



Unbounded Priority Inversion Example

- T1 has highest priority; T6 has lowest priority
- T1 Missing deadline....

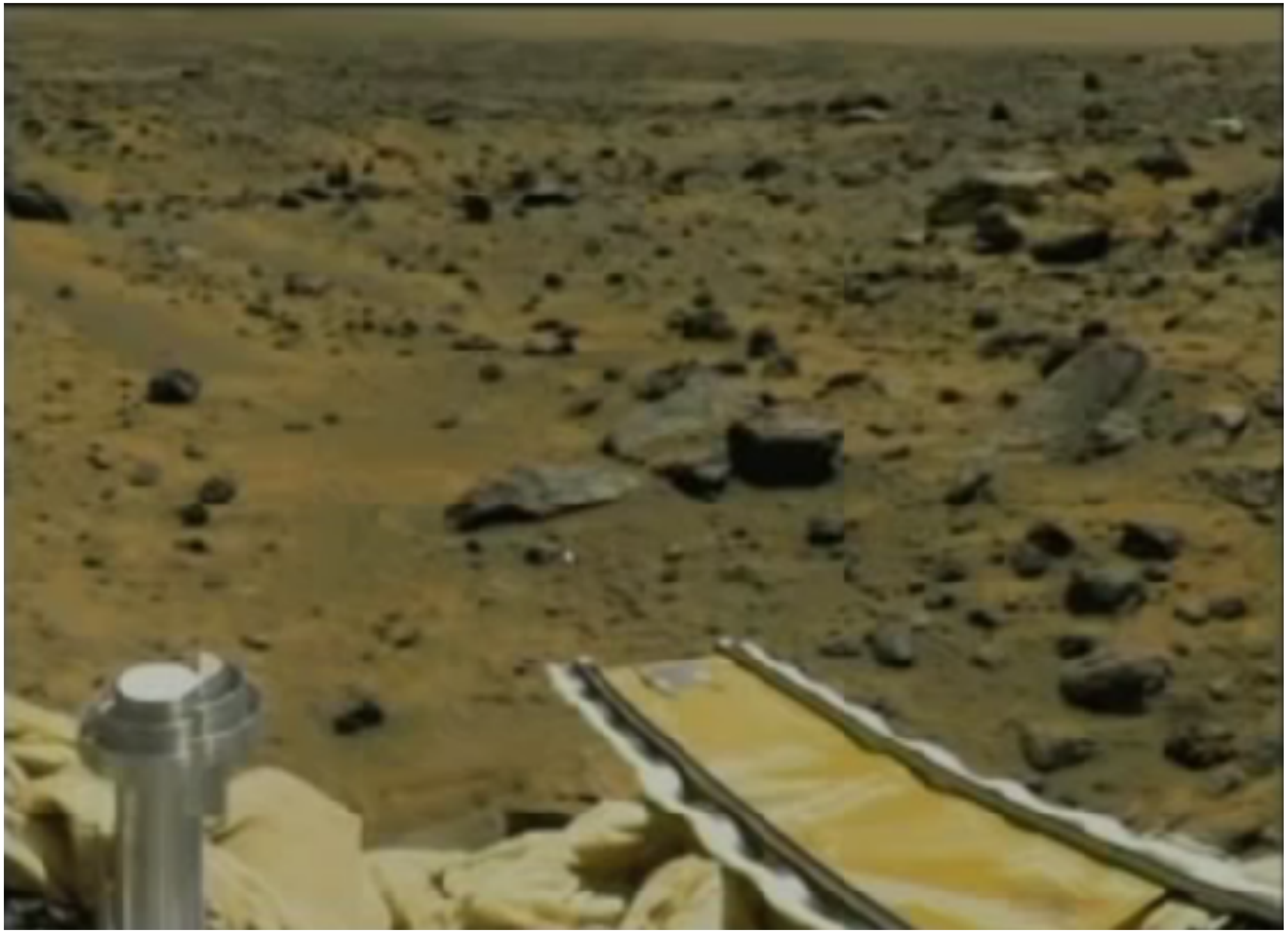


Mars PathFinder

- 4th July 1997, landed on the Mars surface
- Bounced on the martian surface, surrounded by airbags
- Deployed the Sojourner Rover
- Gathered and transmitted enormous data to the earth

Mars PathFinder – Sojourner Rover

Pictures from Mars PathFinder



Mars PathFinder Gitch

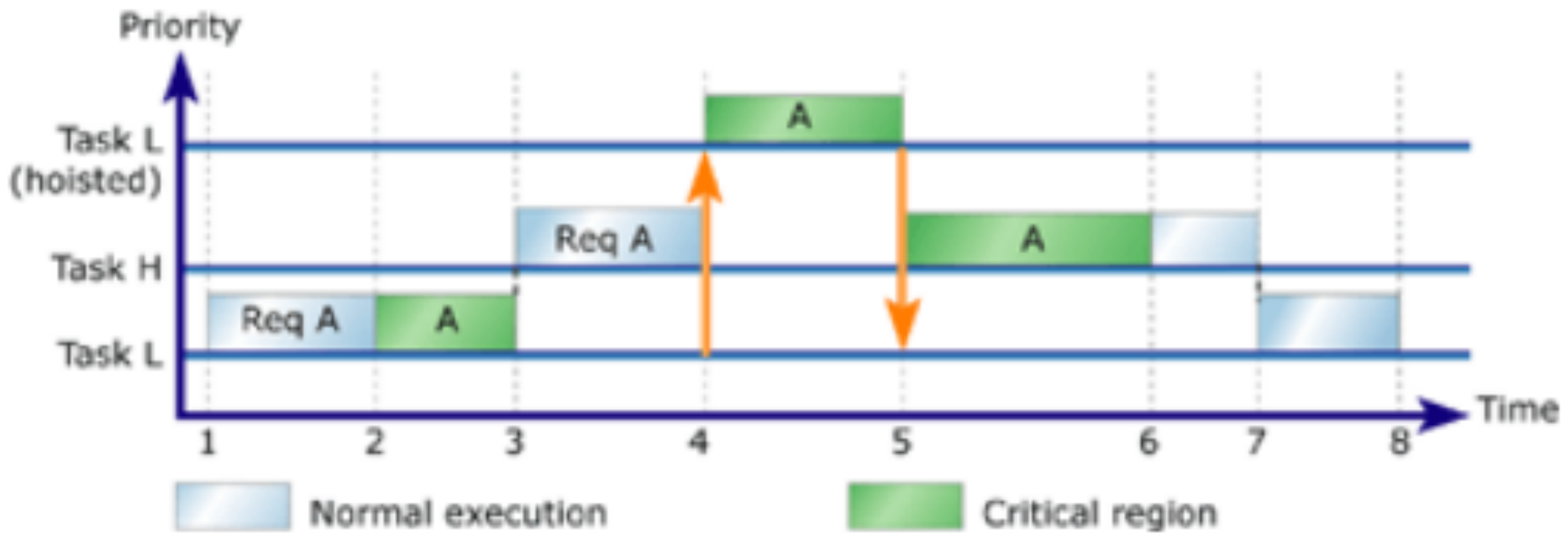
- Often started Resetting
- They were using real-time kernel from Wind River Systems Ltd – VxWorks RTOS
- VxWorks can run in Trace mode:
  - Events like context switching, use of synchronisation objects and interrupts are recorded
  - Replicate the precise condition for reset
  - Unbounded priority inversion was occurring
  - Found global variable to initialize the Priority Inheritance was turned OFF

Solution to Priority Inversion

- How to make a task complete as early as possible ?

- Priority Inheritance Protocol
- Priority Ceiling Protocol

Priority Inheritance Protocol



Raise the priority of Low Priority Task (if it’s holding the resource of higher priority)
The priority is raised to the level of the task that is holding the resource.

Priority Inheritance Protocol

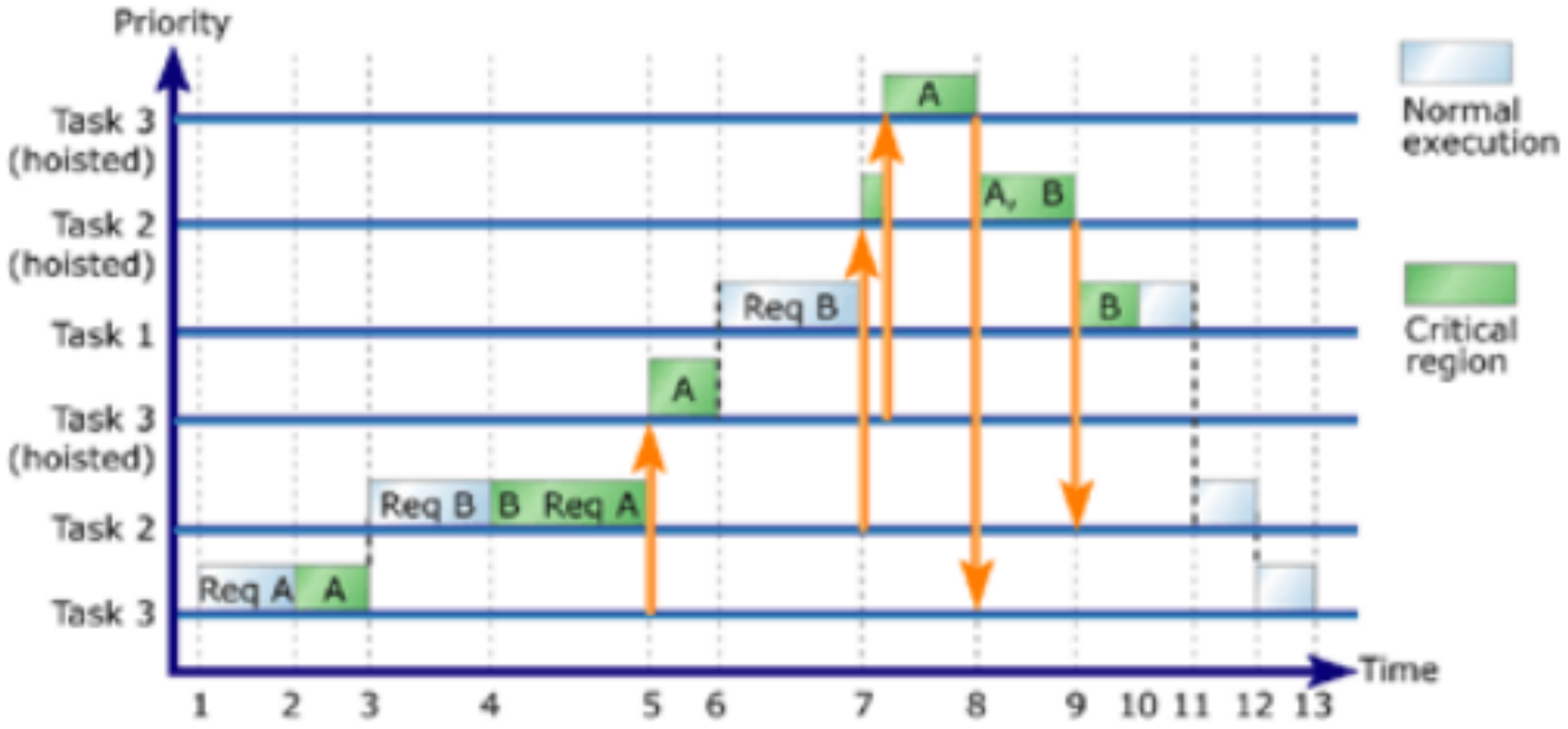
- As soon as the task release its resource,

It gets back to its original priority value if it holding no other critical resource

- If in case, its holding other critical resources:

It inherits the priority of the highest priority task waiting for resource.

Three Tasks and 2 resources



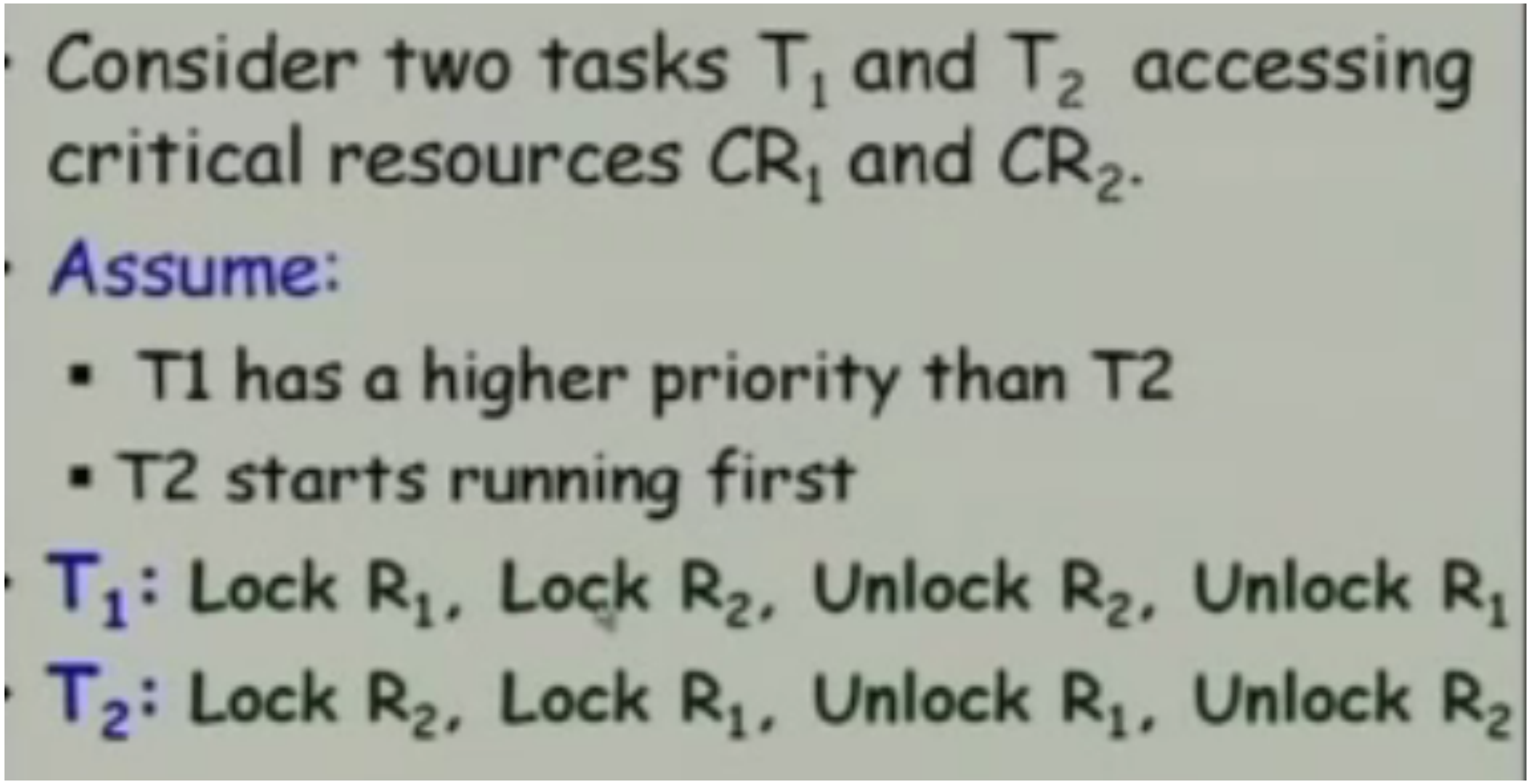
Priority Inheritance Protocol

- PIP resolves unbounded priority inversion issue.

These two issues still persist:

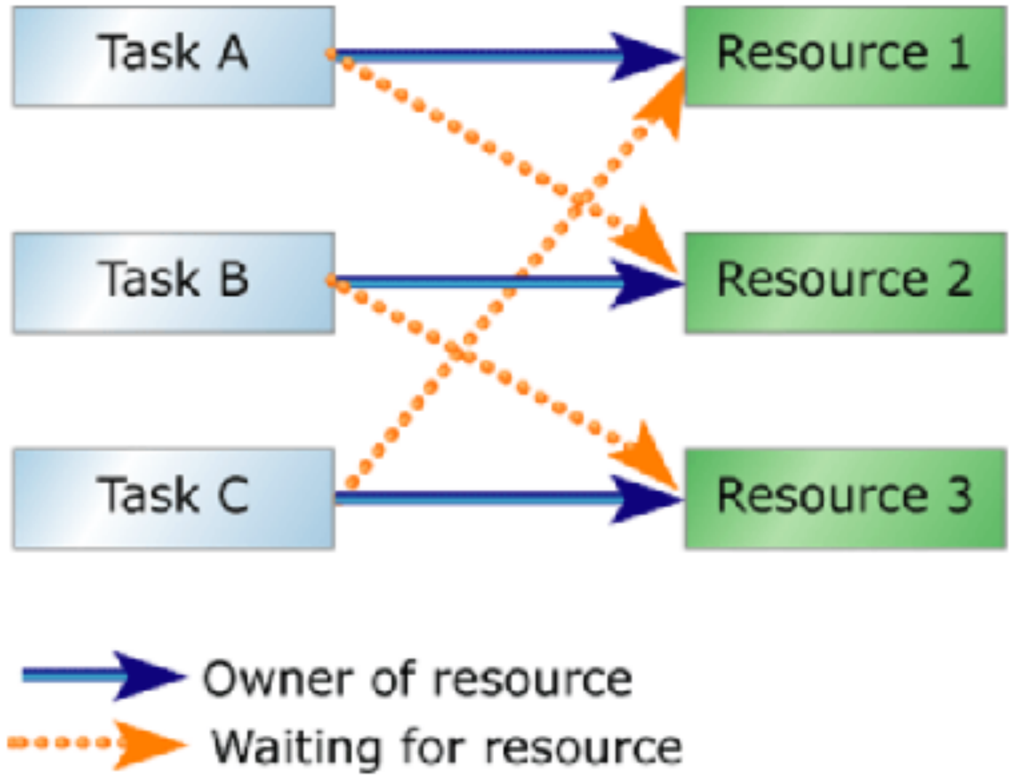
- Deadlock
- Chain Blocking

Deadlocks



- Consider two tasks T_1 and T_2 accessing critical resources CR_1 and CR_2.
- Assume:
  - T1 has a higher priority than T2
  - T2 starts running first
- T_1: Lock R_1, Lock R_2, Unlock R_2, Unlock R_1
- T_2: Lock R_2, Lock R_1, Unlock R_1, Unlock R_2

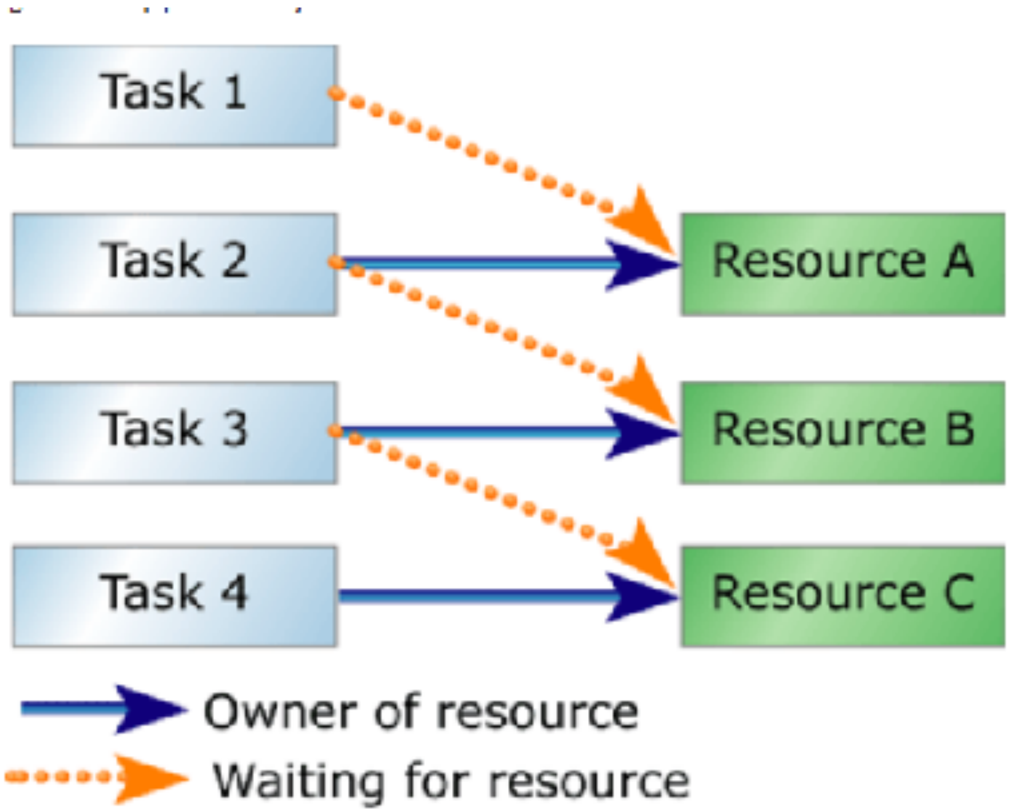
Deadlock – Example 2



Chain Blocking

- Each time a Task needs a resource, it needs to undergo priority inversion
- Example: A high priority task T1 needs several resources.....from other tasks...

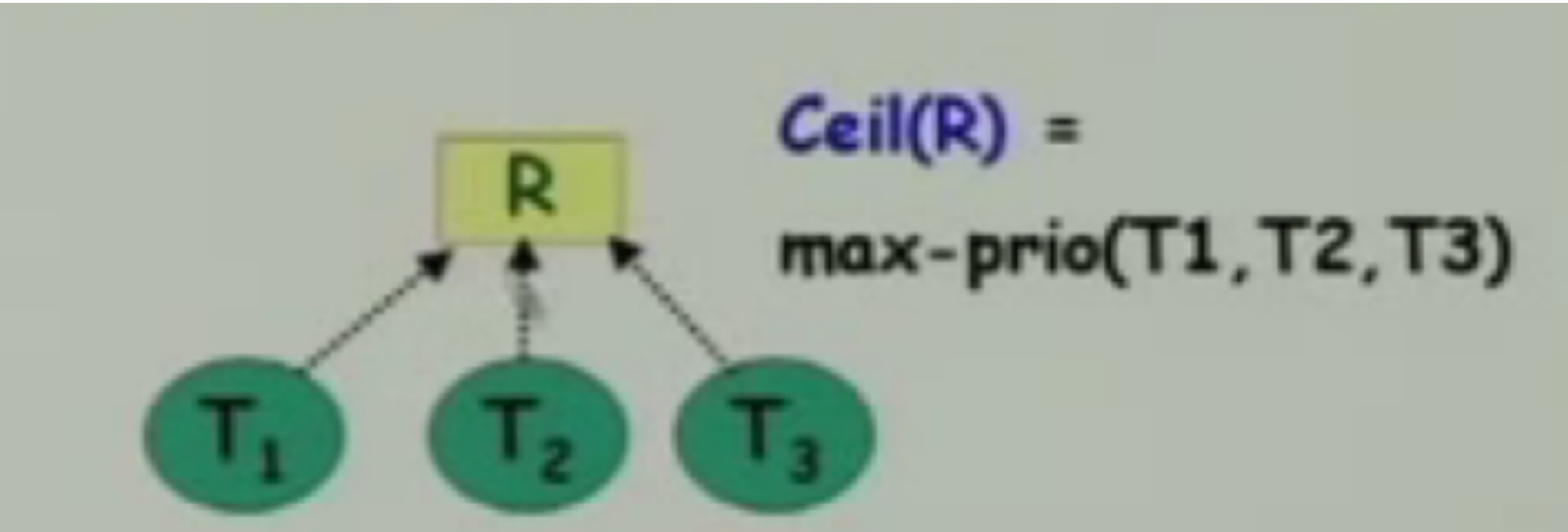
Chain of Nested Resource Locks



Priority Ceiling Protocol

Ceiling priority of a resource

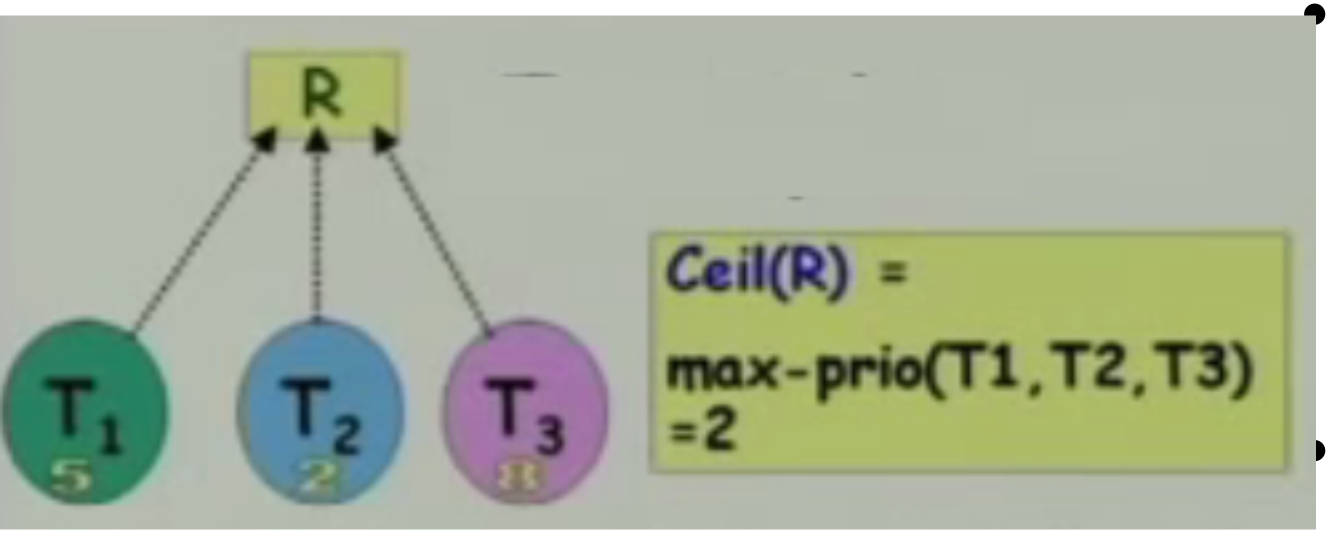
- As soon as a task acquires a Resource, it's priority value is raised to the ceiling priority of that resource



Priority Ceiling Protocol

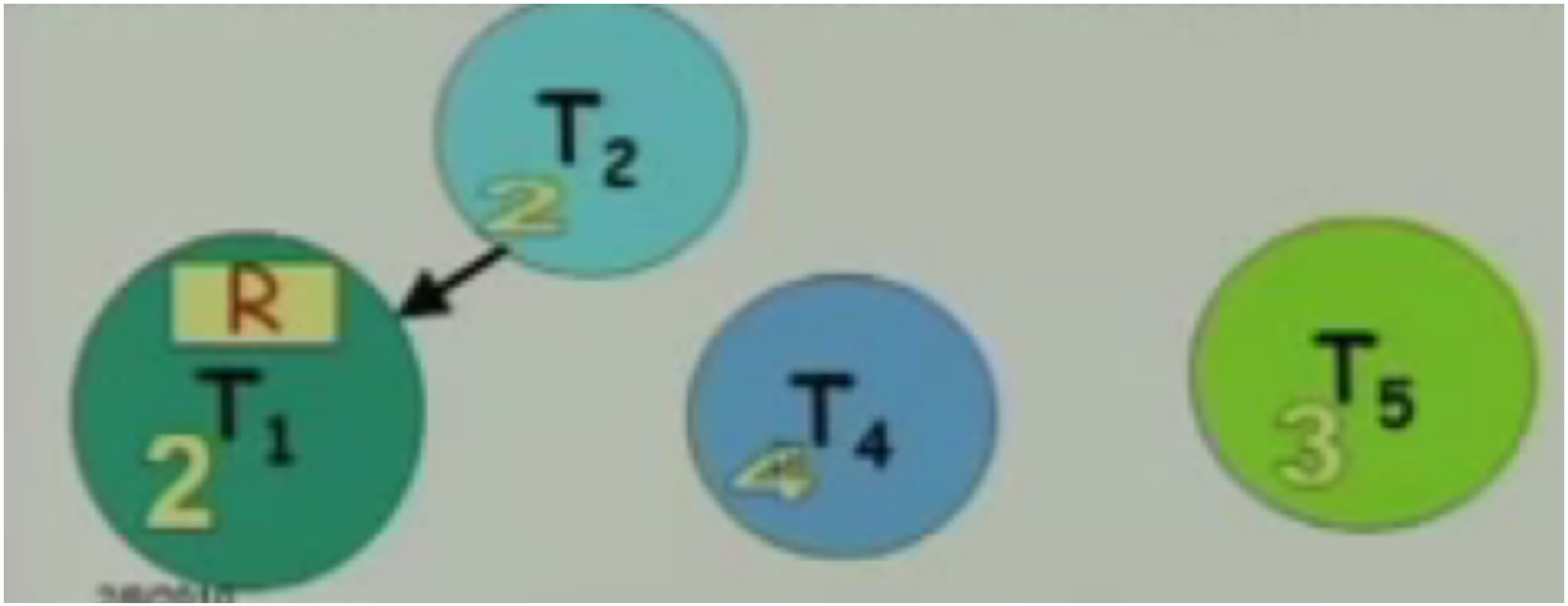
- Solves the problem of unbounded priority inversion, deadlock and chain blocking

Example



- In Unix, lower priority value indicates higher priority
- In Windows, lower value indicates lower priority

Priority Ceiling Protocol



- Unbounded priority inversion is resolved

Priority Ceiling Protocol - Issues

- Inheritance Blocking:
  - When a priority value of a low priority task holding a resource is raised to higher value
  - Intermediate priority tasks not needing the resource, cannot execute and hence undergo priority inversion.....
- Practically, it’s not possible to know when a resource is needed by a task...