



Real-time Synchronization

(Semaphores, Resources and Blocking)

Priority Inheritance

Priority Ceiling

Slack Resource Policy



Reminder

- MP1 due soon.

Announcements

R&D

Wanted!!

- Undergraduates for R&D positions
- Masters/MCS: Full RAships
- Professional (when you graduate): Up to \$75K/yr

The collage features several overlapping elements:

- Chicago Tribune Article:** "University of Illinois leads \$25 million military technology initiative". The article includes a photo of military personnel in a meeting and a quote from Tarek Abdelzaher: "This award enables a true collaboration between researchers at ARL and researchers in academia and industry to change the status quo in smart battlefield services."
- The News-Gazette Article:** "UI leading \$25 million effort to... military". It includes a video thumbnail of a man speaking.
- Blue Sky Originals:** "The Army wants smarter tech gear. U. of I. is leading the charge." with a video thumbnail of a soldier in a futuristic environment.
- WCIA 3 News at Nine:** "INTERNET OF BATTLEFIELD THINGS" with a video thumbnail of a military vehicle in a field.
- CSL NEWS/MEDIA:** A header for the Coordinated Science Lab news page.

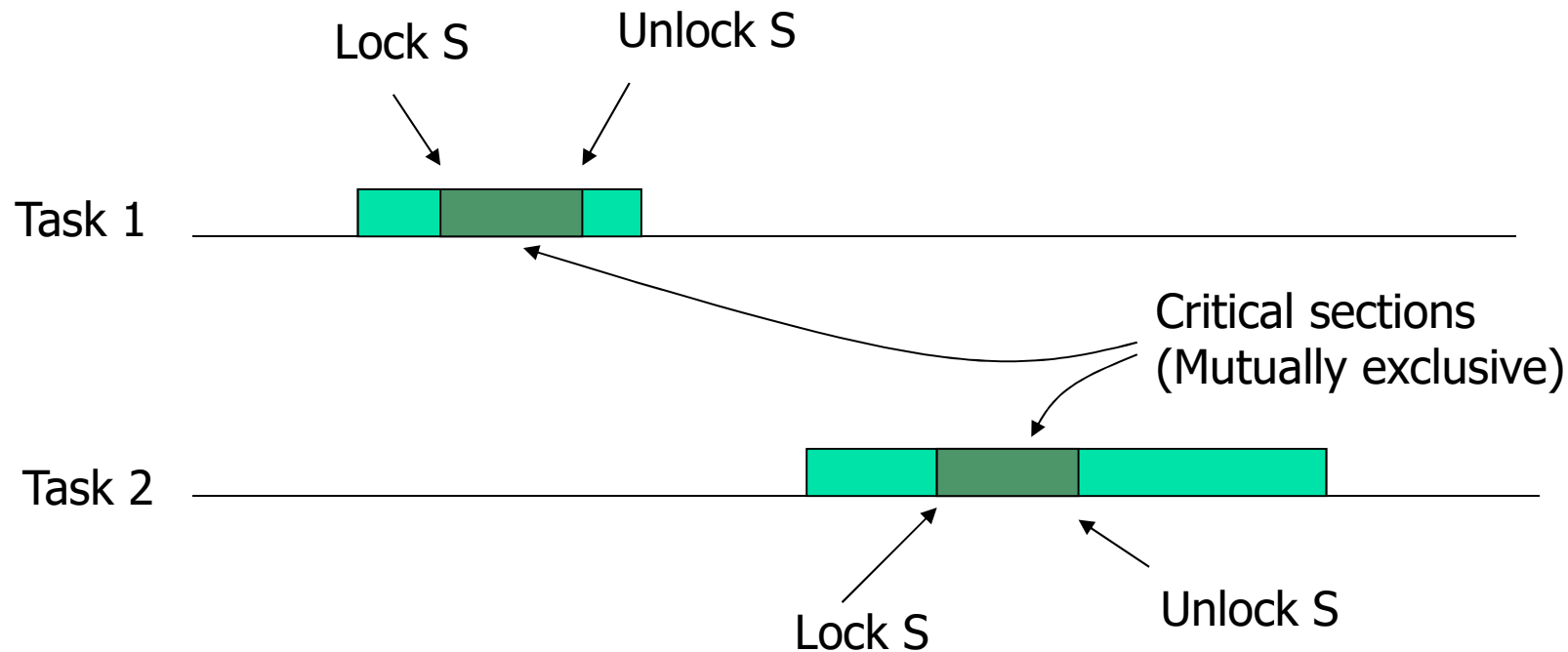


The Problem

- Tasks have synchronization constraints
 - Semaphores protect critical sections
- Blocking can cause a higher-priority task to wait on a **lower-priority** one to unlock a resource
 - Problem: In all previous derivations we assumed that a task can only wait for **higher-priority** tasks not **lower-priority** tasks
- Question
 - What is the maximum amount of time a higher-priority task can wait for a lower-priority task?
 - How to account for that time in schedulability analysis?

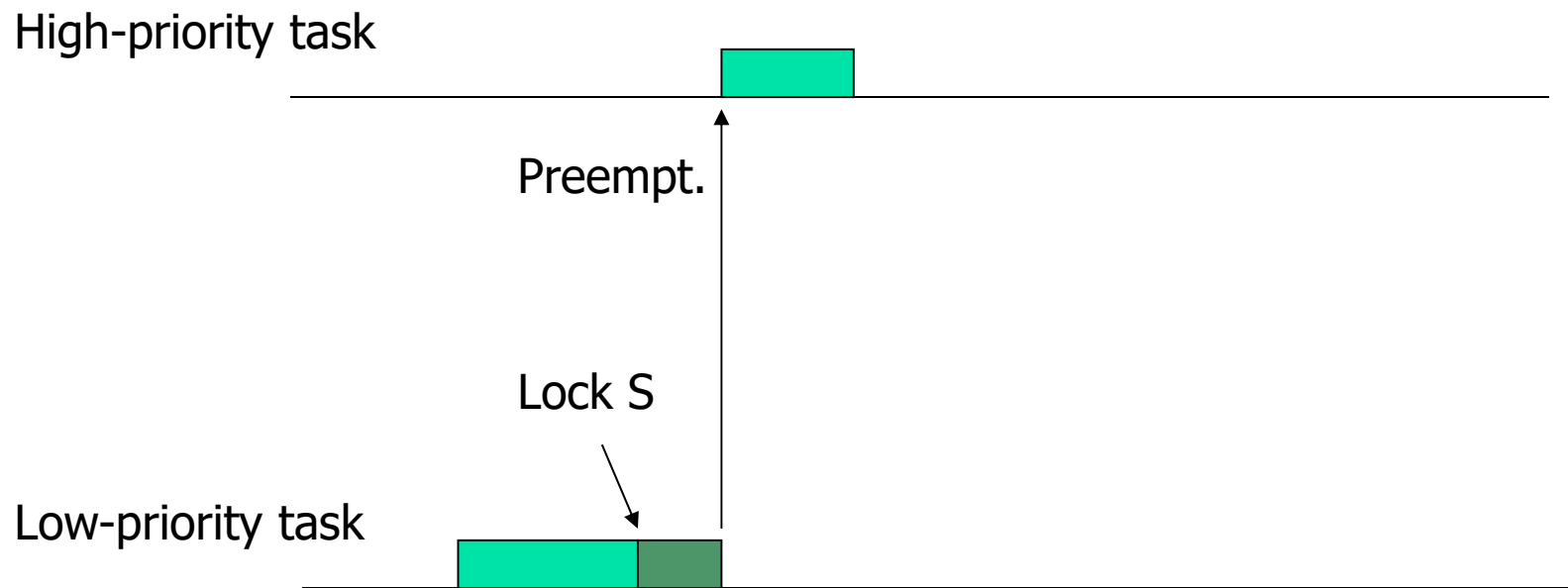
Mutual Exclusion Constraints

- Tasks that lock/unlock the same semaphore are said to have a mutual exclusion constraint



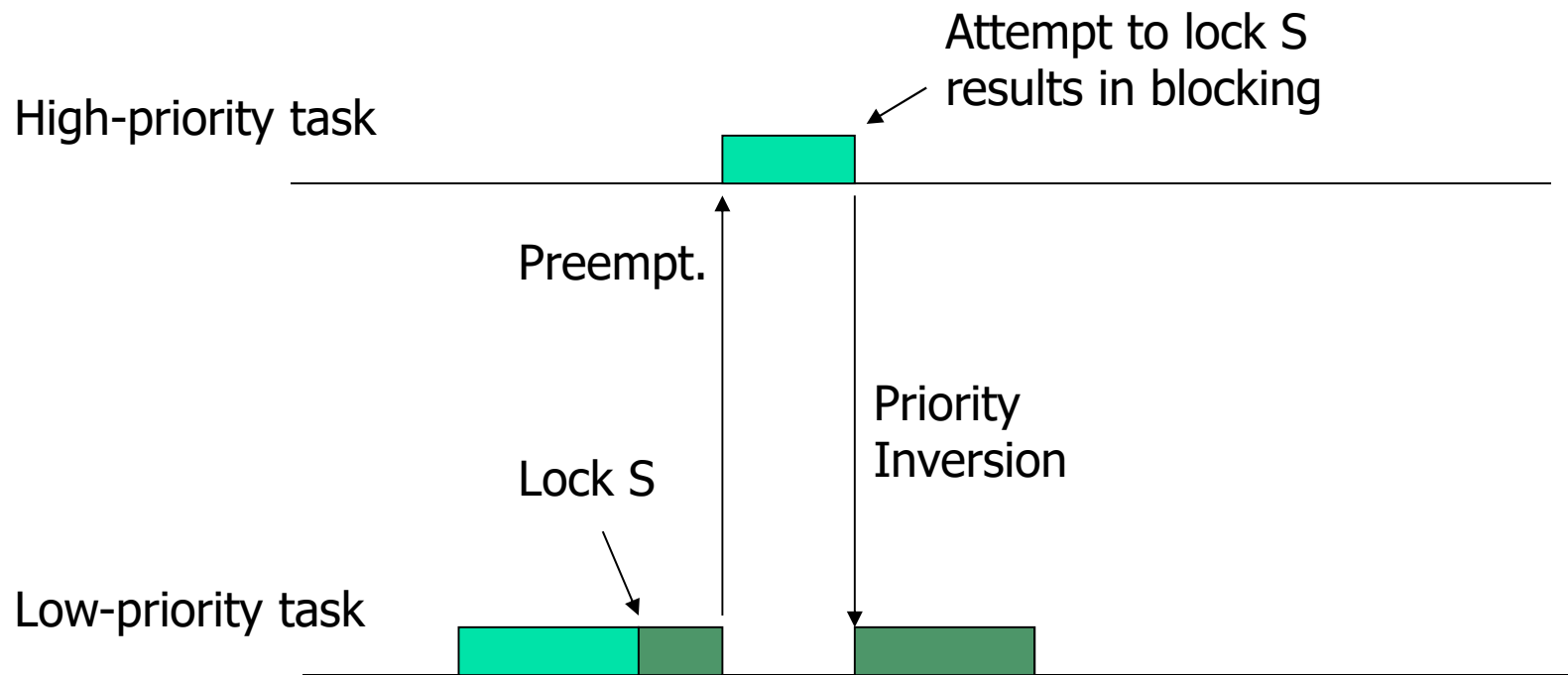
Priority Inversion

- Locks and priorities may be at odds. Locking results in priority inversion



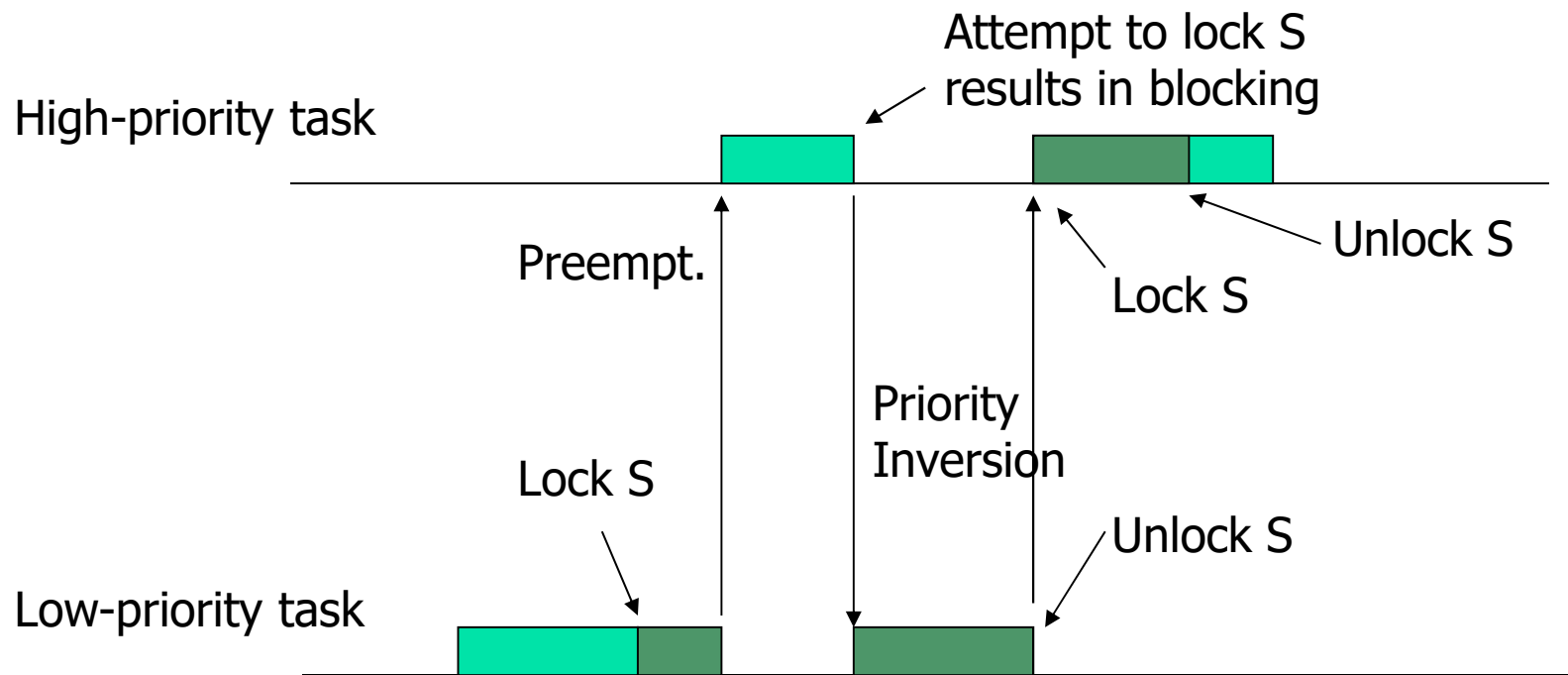
Priority Inversion

- Locks and priorities may be at odds. Locking results in priority inversion



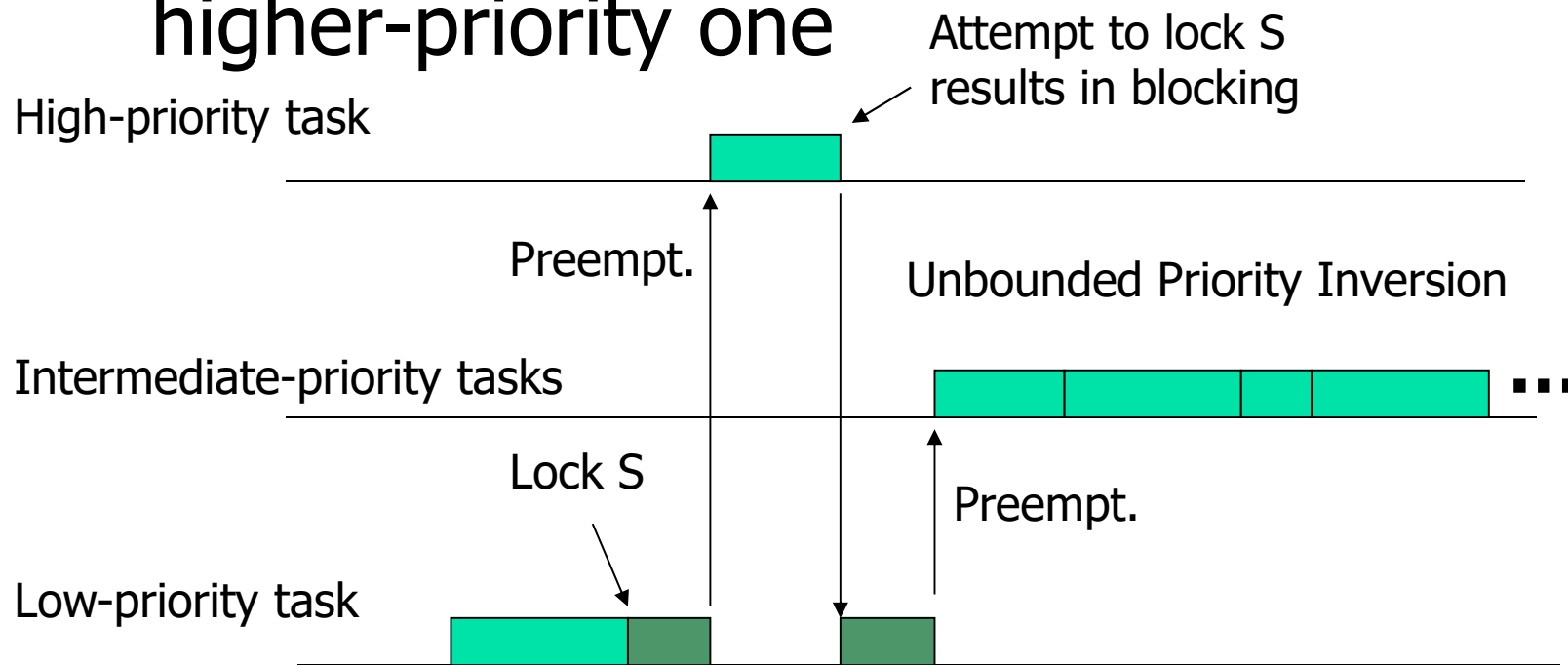
Priority Inversion

- How to account for priority inversion?



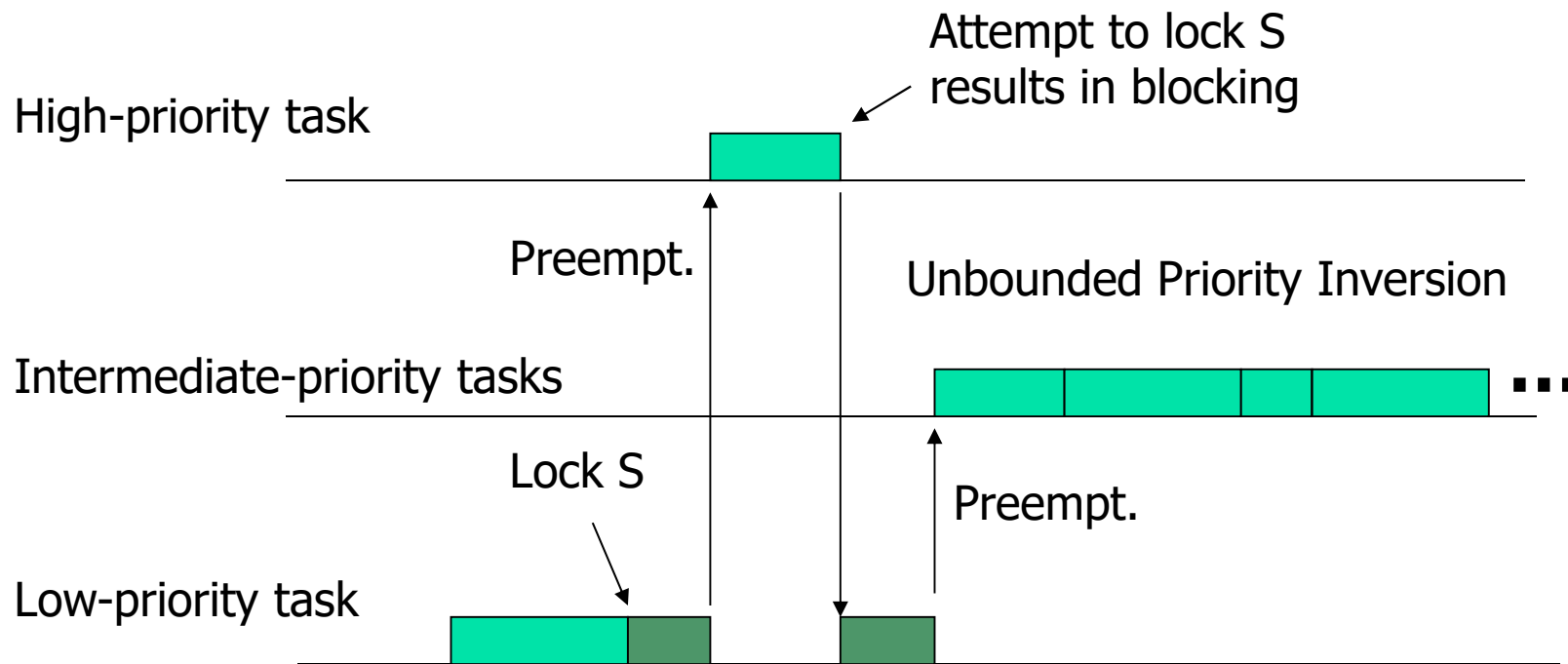
Unbounded Priority Inversion

- Consider the case below: a series of intermediate priority tasks is delaying a higher-priority one



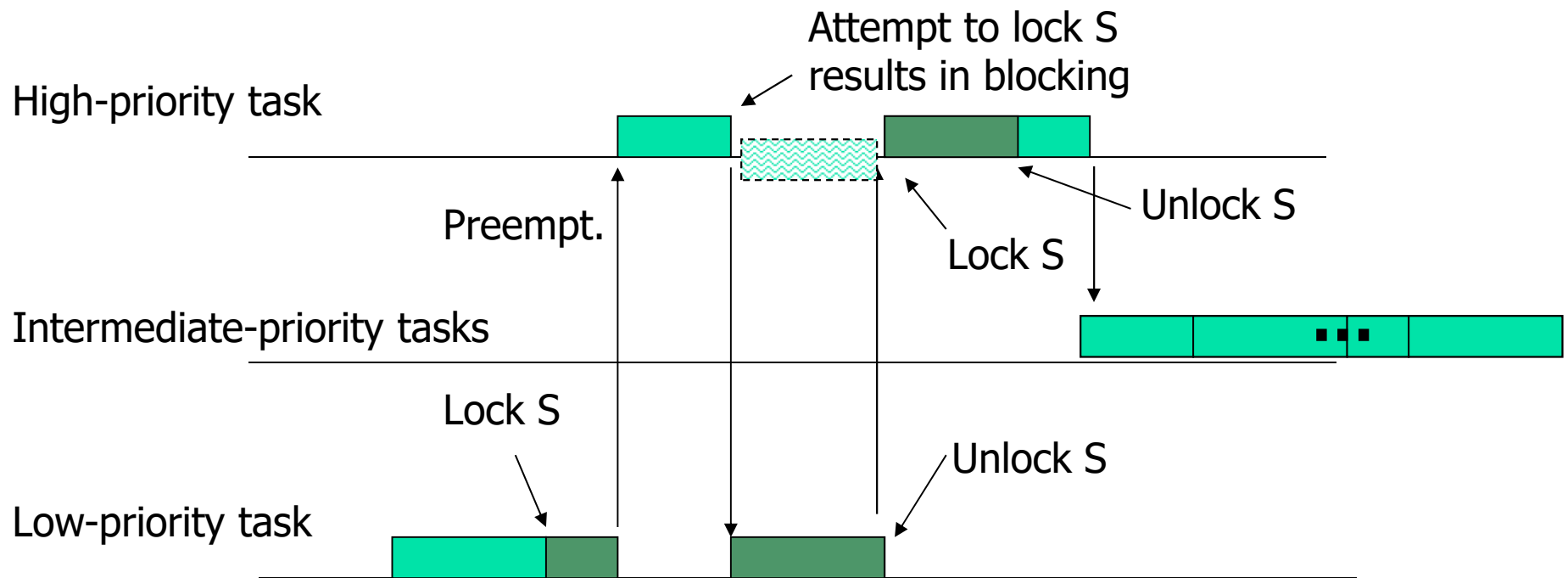
Unbounded Priority Inversion

- How to prevent unbounded priority inversion?



Priority Inheritance Protocol

- Let a task inherit the priority of any higher-priority task it is blocking



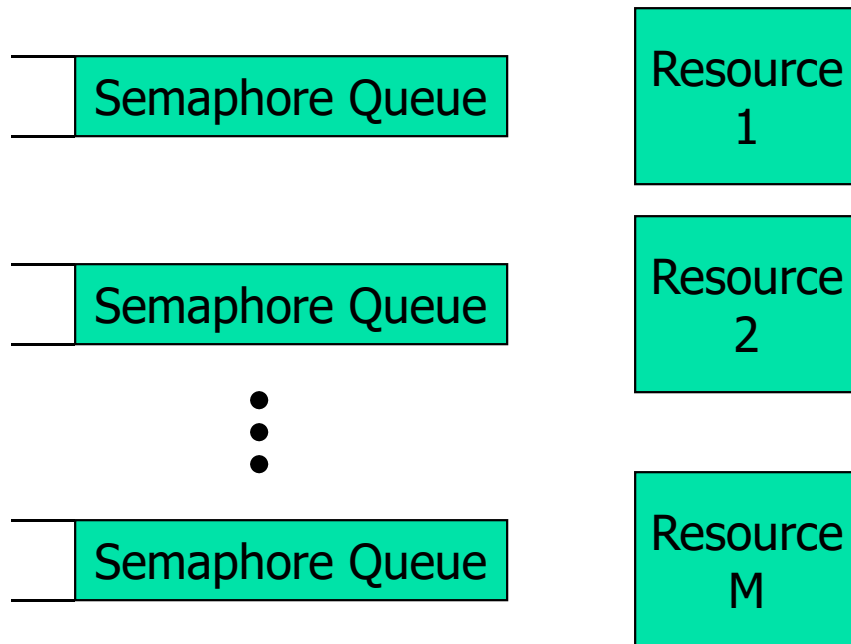


Priority Inheritance Protocol

- Question: What is the longest time a task can wait for lower-priority tasks?
 - Let there be N tasks and M semaphores
 - Let the largest critical section of task i be of length B_i
- Answer: ?

Computing the Maximum Priority Inversion Time

- Consider the instant when a high-priority task that arrives.
 - What is the most it can wait for lower priority ones?



If I am a task, priority inversion occurs when

- Lower priority task holds a resource I need (direct blocking)
- Lower priority task inherits a higher priority than me because it holds a resource the higher-priority task needs (push-through blocking)



Maximum Blocking Time

- If all critical sections are equal (of length B):
 - Blocking time = $B \min(N, M)$
(Why?)
- If they are not equal?



Maximum Blocking Time

- If all critical sections are equal (of length B):
 - Blocking time = $B \min(N, M)$
(Why?)
- If they are not equal
 - Find the worst (maximum length) critical section for each resource
 - Add up the top $\min(N, M)$ sections in size
- The total priority inversion time for task i is called B_i



Schedulability Test

$$\forall i, 1 \leq i \leq n,$$

$$\frac{B_i}{P_i} + \sum_{k=1}^i \frac{C_k}{P_k} \leq i(2^{1/i} - 1)$$



Schedulability Test

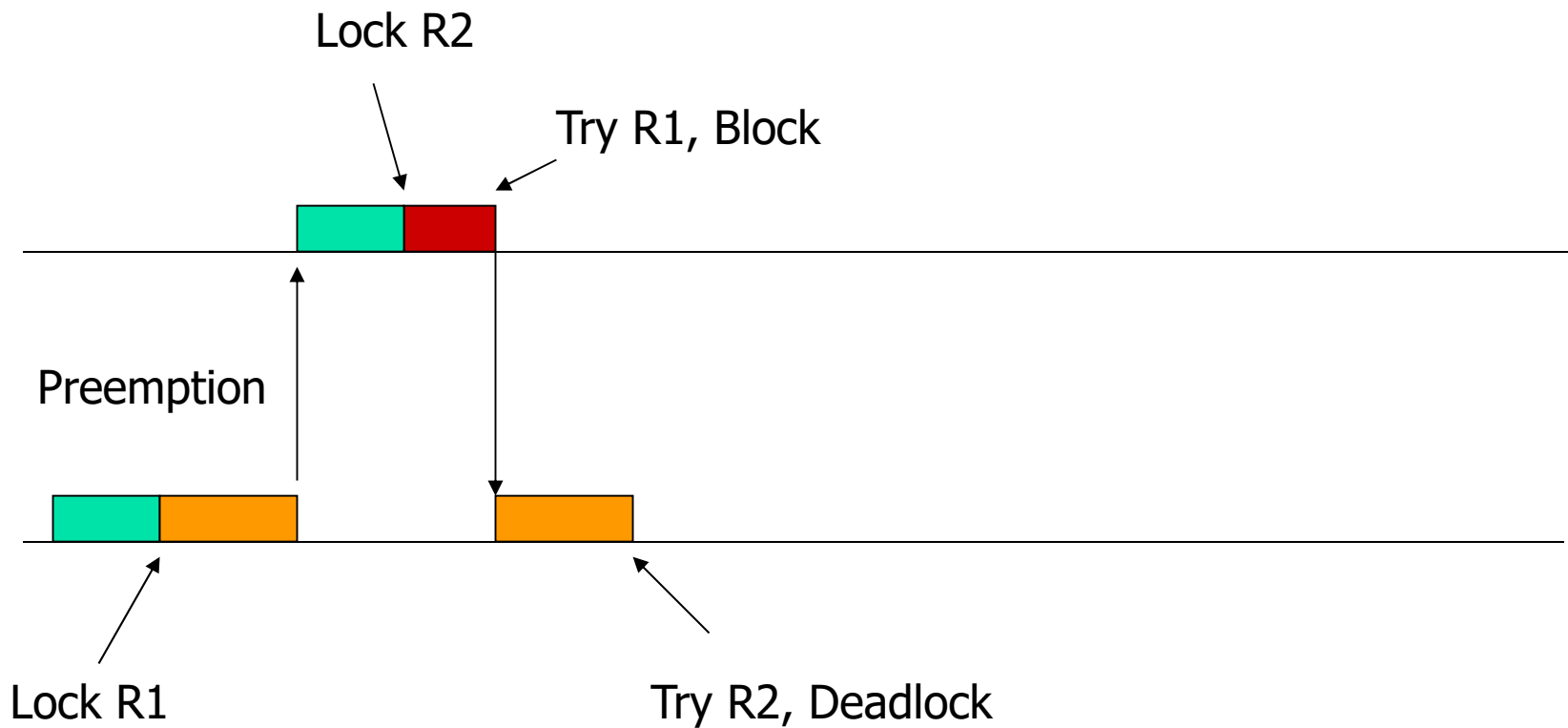
$$\forall i, 1 \leq i \leq n,$$

$$\frac{B_i}{P_i} + \sum_{k=1}^i \frac{C_k}{P_k} \leq i(2^{1/i} - 1)$$

Why do we have to test each task separately? Why not just one utilization-based test like it used to?

Problem: Deadlock

Deadlock occurs if two tasks locked two semaphores in opposite order



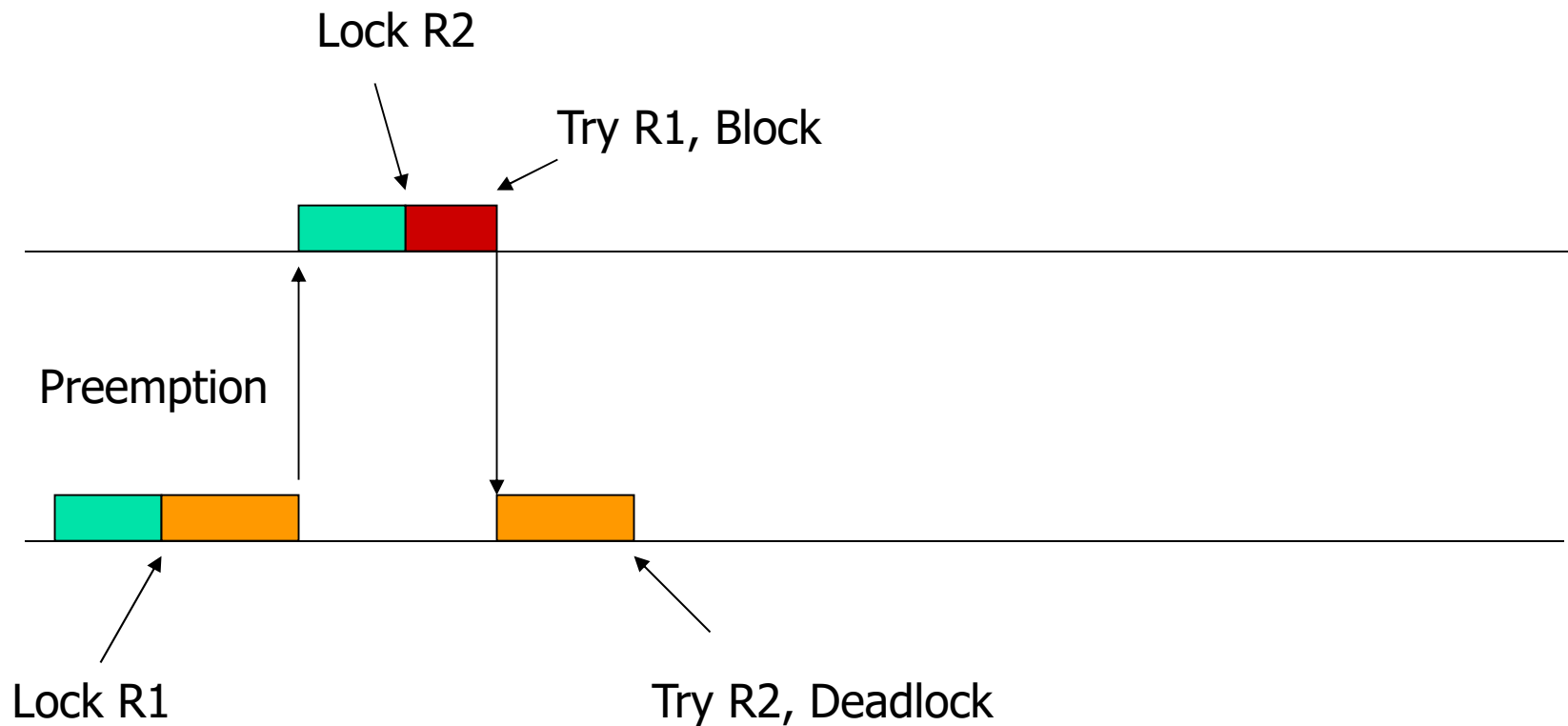


Priority Ceiling Protocol

- Definition: The priority ceiling of a semaphore is the highest priority of any task that can lock it
- A task that requests a lock R_k is denied if its priority is not higher than the highest priority ceiling of all currently locked semaphores (say it belongs to semaphore R_h)
 - The task is said to be blocked by the task holding lock R_h
- A task inherits the priority of the top higher-priority task it is blocking

Problem: Deadlock?

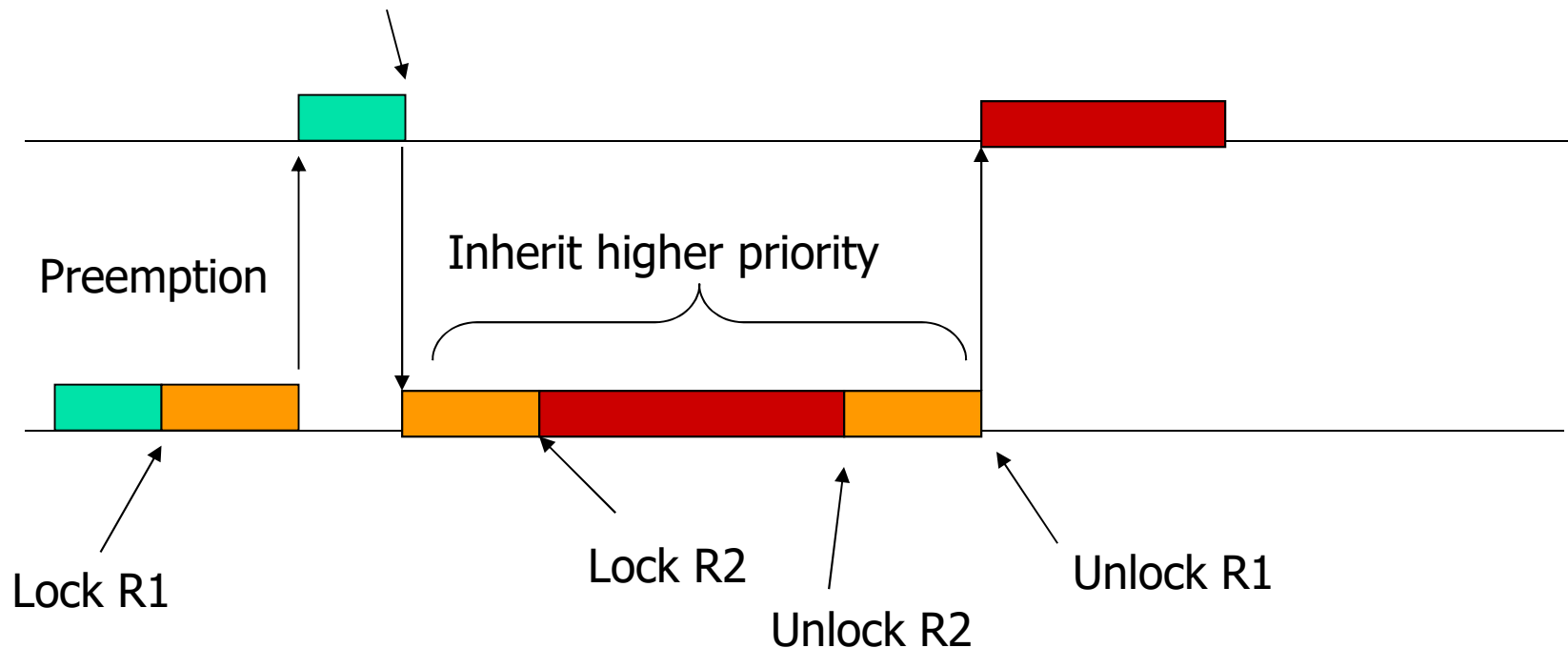
Deadlock used to occur if two tasks locked two semaphores in opposite order. Can it still occur in priority ceiling?



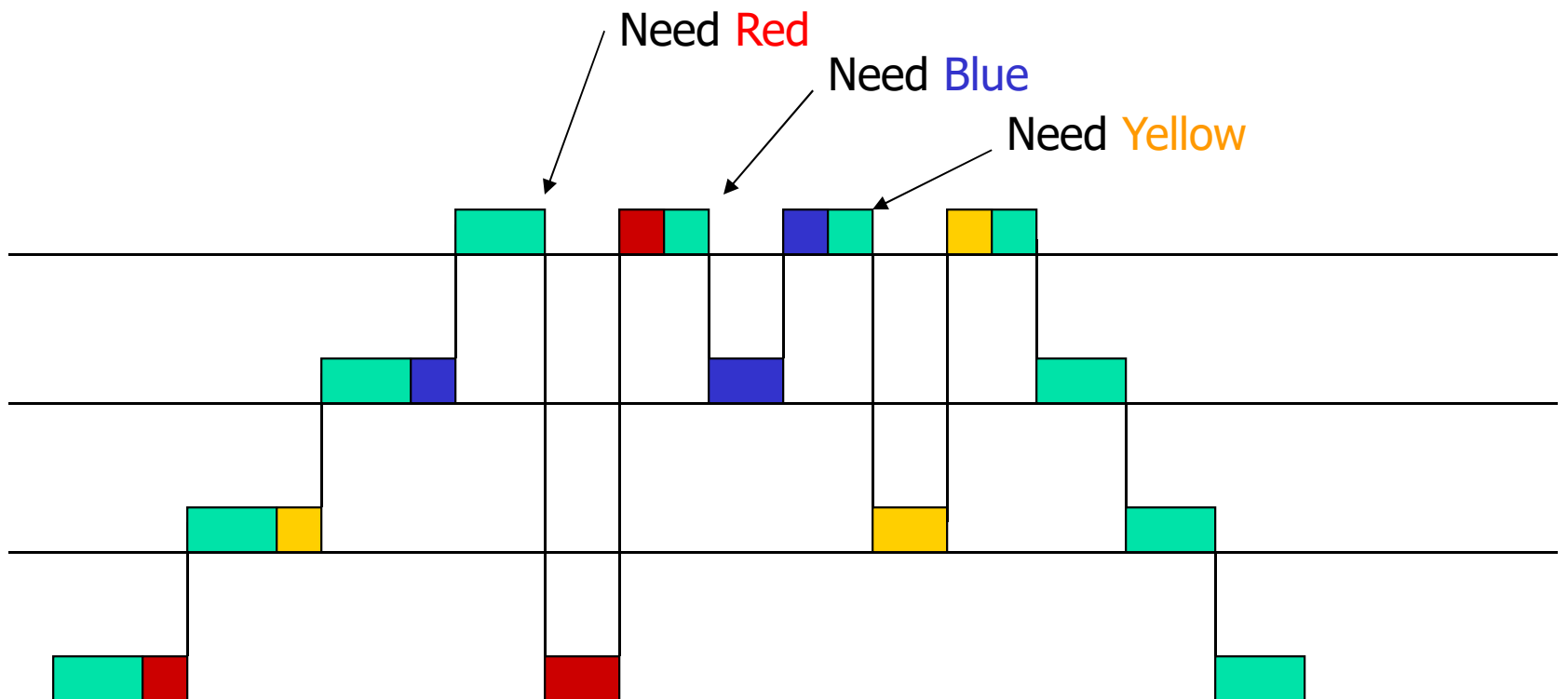
Problem: Deadlock?

Deadlock used to occur if two tasks locked two semaphores in opposite order. Can it still occur in priority ceiling?

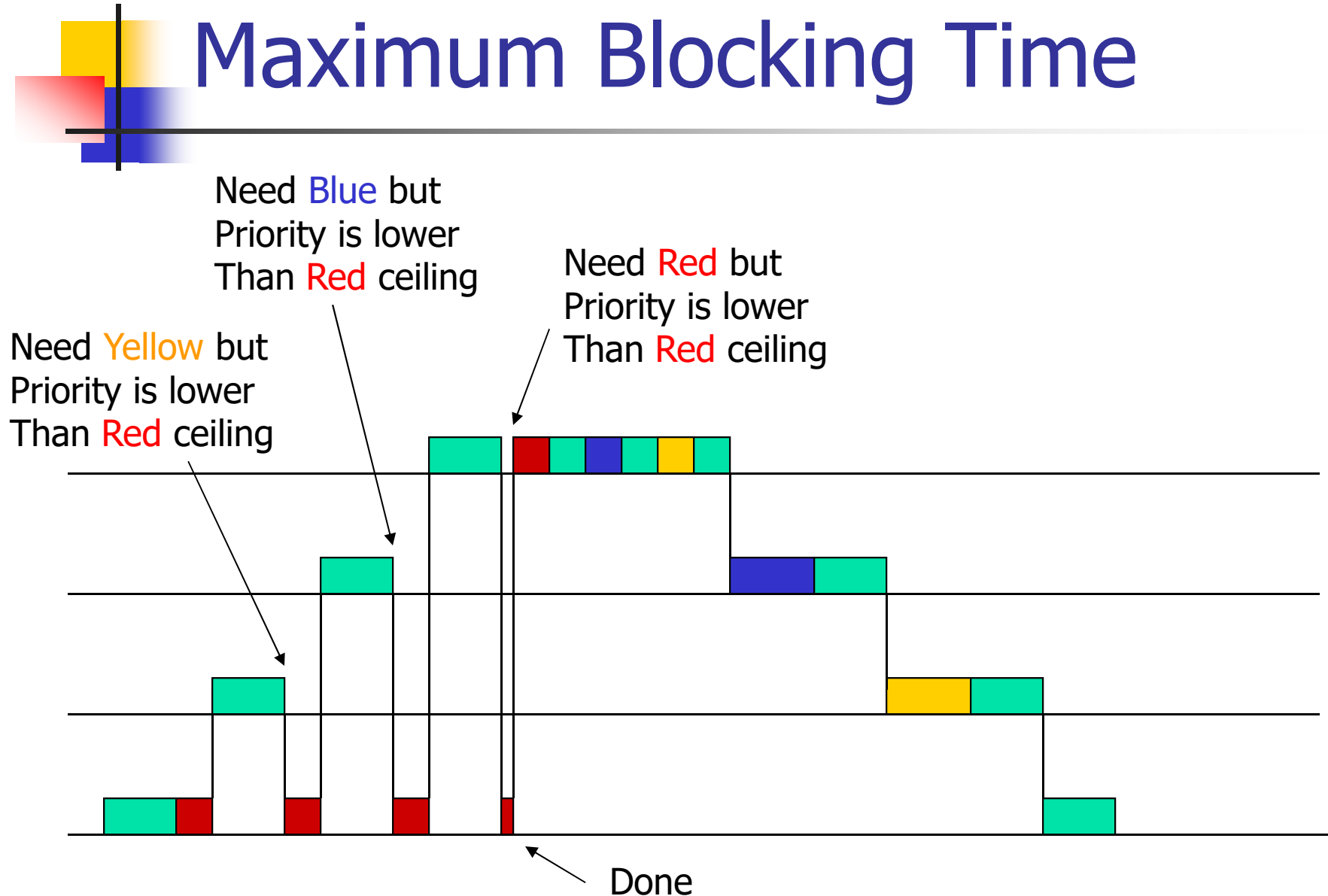
Lock R2: **Denied because its priority is not higher than ceiling of R1**



Priority Inheritance Protocol: Maximum Blocking Time



Priority Ceiling Protocol: Maximum Blocking Time





Schedulability

- A task can be preempted by only one critical section of a lower priority task (that is guarded by a semaphore of equal or higher priority ceiling). Let max length of such section be B_i

$$\forall i, 1 \leq i \leq n,$$

$$\frac{B_i}{P_i} + \sum_{k=1}^i \frac{C_k}{P_k} \leq i(2^{1/i} - 1)$$

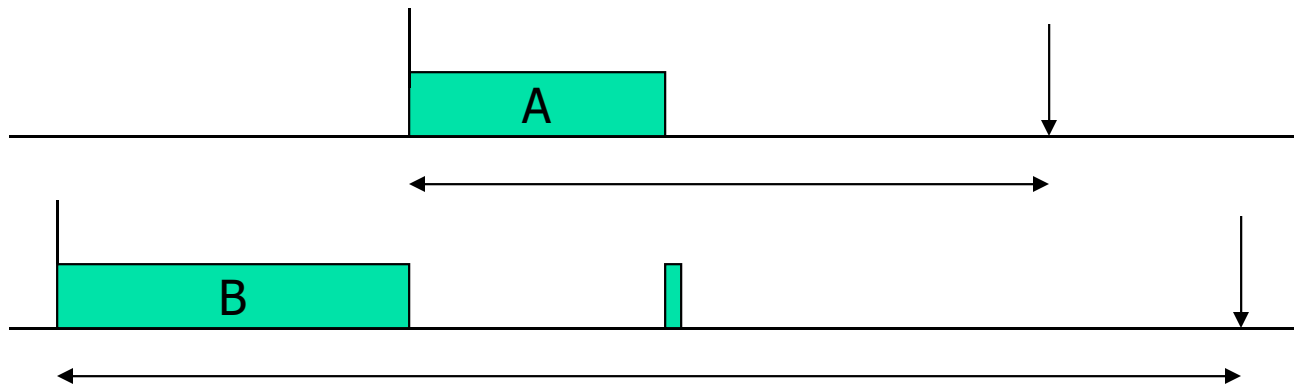


Slack Resource Policy

- Priority:
 - Any static or dynamic policy (e.g., EDF, RM, ...)
- Preemption Level
 - Any *fixed value* that satisfies: If A arrives after B and Priority (A) > Priority (B) then PreemptionLevel (A) > PreemptionLevel (B)
- Resource Ceiling
 - Highest preemption level of all tasks that might access the resource
- System Ceiling
 - Highest resource ceiling of all currently locked resources
- A task can preempt another if:
 - It has the highest priority
 - Its preemption level is higher than the system ceiling

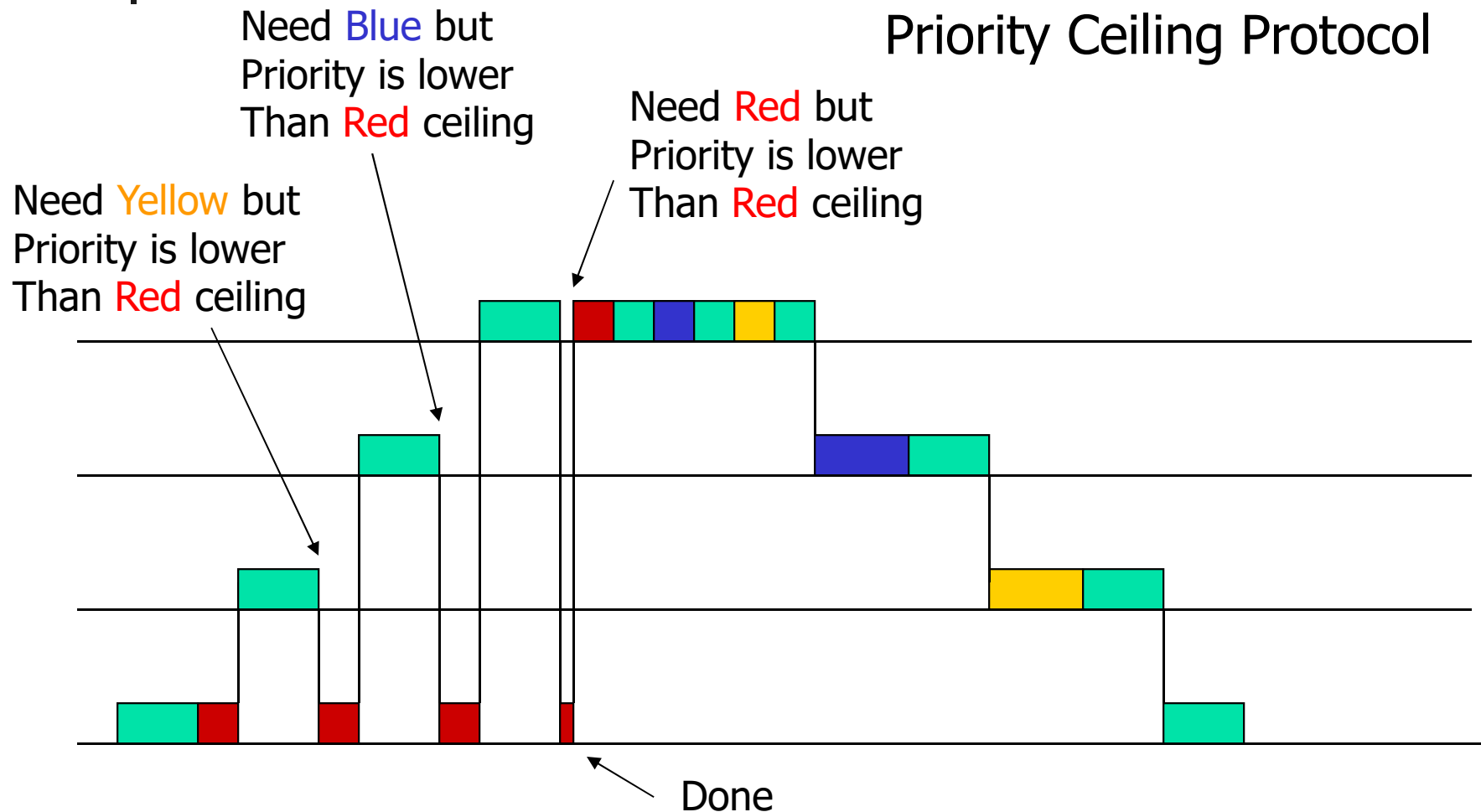
Example: EDF

- Priority is proportional to the absolute deadline
- Preemption level is proportional to the relative deadline (shorter \rightarrow higher priority).
- Observe that:
 - If A arrives after B and $\text{Priority}(A) > \text{Priority}(B)$ then $\text{PreemptionLevel}(A) > \text{PreemptionLevel}(B)$



Maximum Blocking Time

Priority Ceiling Protocol



Maximum Blocking Time

Slack Resource Policy

Can't preempt
Preemption level is not
higher than ceiling

