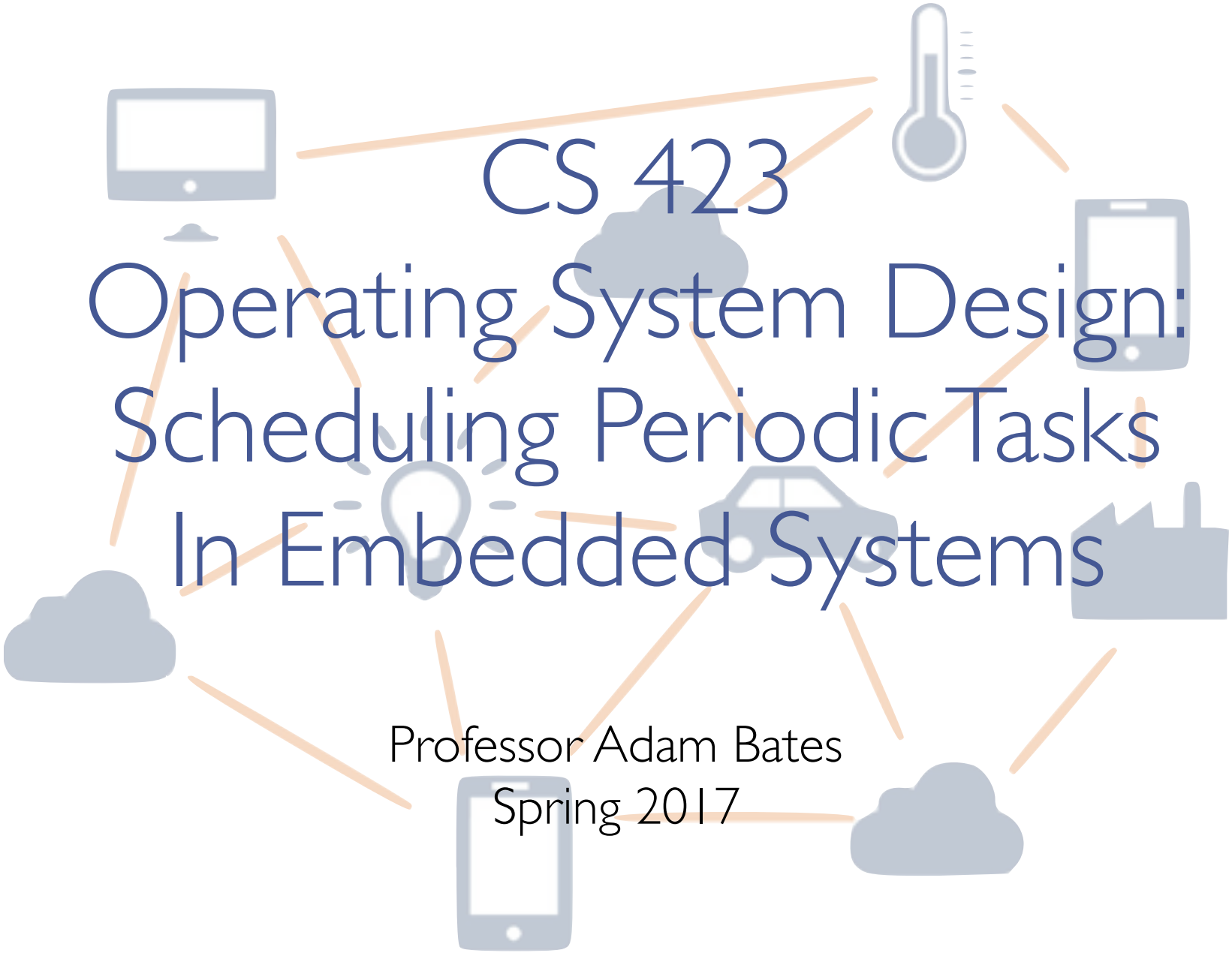


CS 423
Operating System Design: Scheduling Periodic Tasks In Embedded Systems

Professor Adam Bates

Spring 2017

Goals for Today

- Learning Objective:
  - Conclude discussion of scheduling by exploring real-time scheduling in embedded systems.
- Announcements, etc:
  - MP1 Due Feb 19
  - C4 Review Feedback — out this weekend

Reminder: Please put away devices at the start of class

Other scheduling policies

- What if you want to maximize throughput?
  - Shortest job first!
- What if you want to meet all deadlines?
  - Earliest deadline first!
  - Problem?
  - Works only if you are not "overloaded". If the total amount of work is more than capacity, a domino effect occurs as you always choose the task with the nearest deadline (that you have the least chance of finishing by the deadline), so you may miss a lot of deadlines!

EDF Domino Effect

- Problem:
  - It is Monday. You have a homework due tomorrow (Tuesday), a homework due Wednesday, and a homework due Thursday
  - It takes on average 1.5 days to finish a homework.
- Question: What is your best (scheduling) policy?

EDF Domino Effect

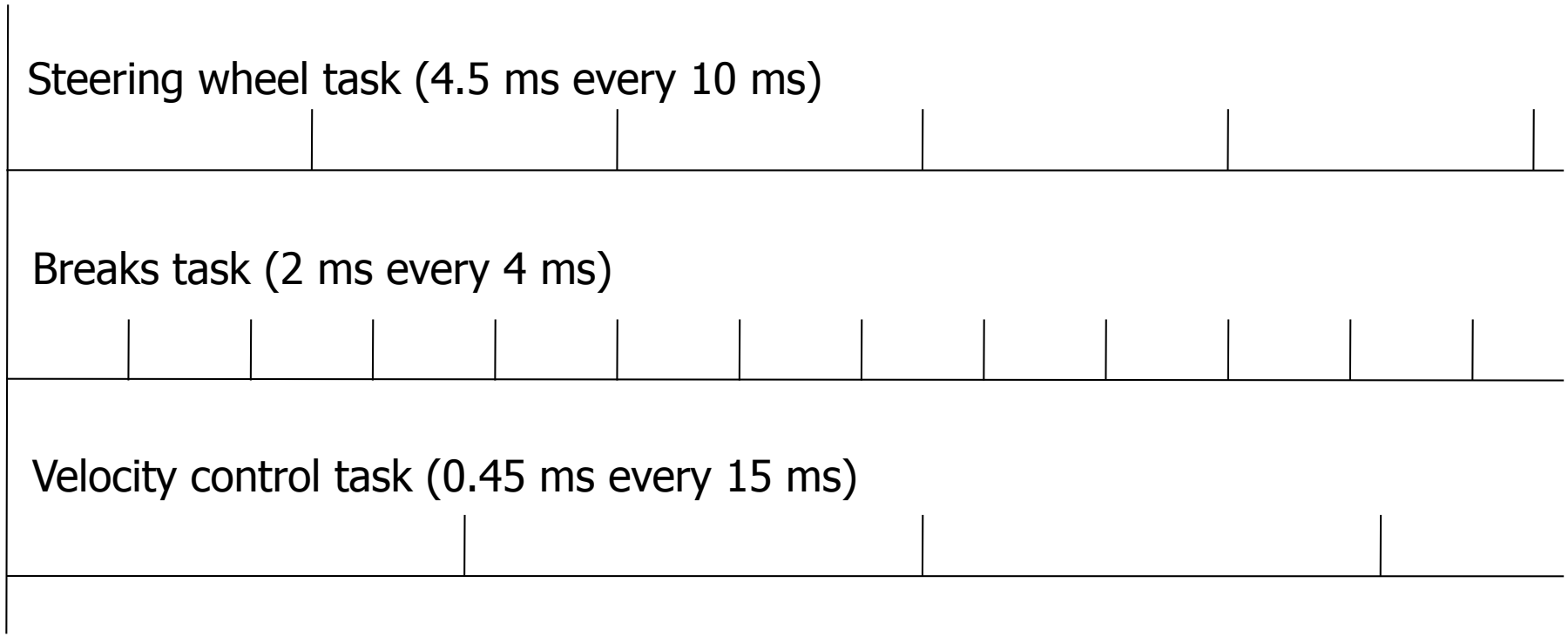
- Problem:
  - It is Monday. You have a homework due tomorrow (Tuesday), a homework due Wednesday, and a homework due Thursday
  - It takes on average 1.5 days to finish a homework.
- Question: What is your best (scheduling) policy?
  - You could instead skip tomorrow's homework and work on the next two, finishing them by their deadlines
  - Note that EDF is bad: It always forces you to work on the next deadline, but you have only one day between deadlines which is not enough to finish a 1.5 day homework – you might not complete any of the three homeworks!

Drive-By-Wire Example

- Consider a control system in a autonomous vehicle
  - Steering wheel sampled every 10 ms – wheel positions adjusted accordingly (computing the adjustment takes 4.5 ms of CPU time)
  - Breaks sampled every 4 ms – break pads adjusted accordingly (computing the adjustment takes 2ms of CPU time)
  - Velocity is sampled every 15 ms – acceleration is adjusted accordingly (computing the adjustment takes 0.45 ms)
  - For safe operation, adjustments must always be computed before the next sample is taken
- How to assign priorities?

Drive-By-Wire Example

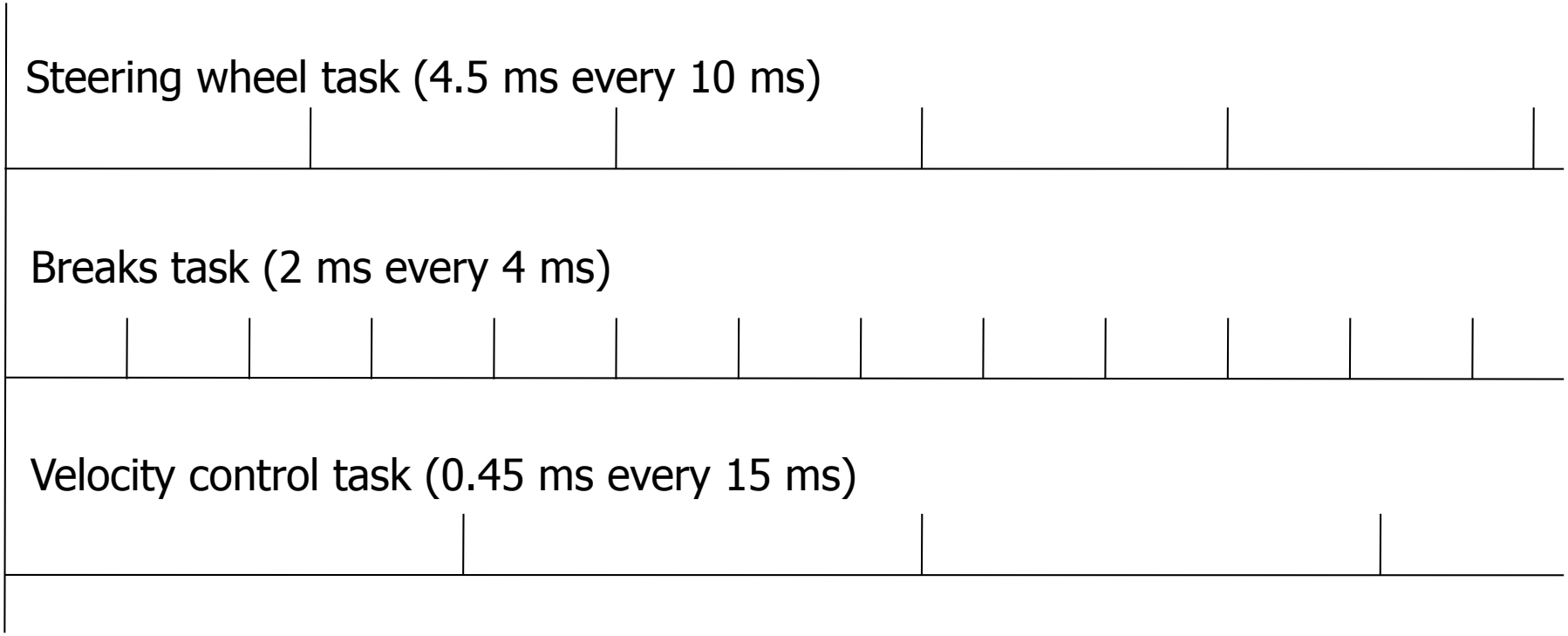
Find a schedule that makes sure all task invocations meet their deadlines



Drive-By-Wire Example

Sanity check: Is the processor over-utilized?

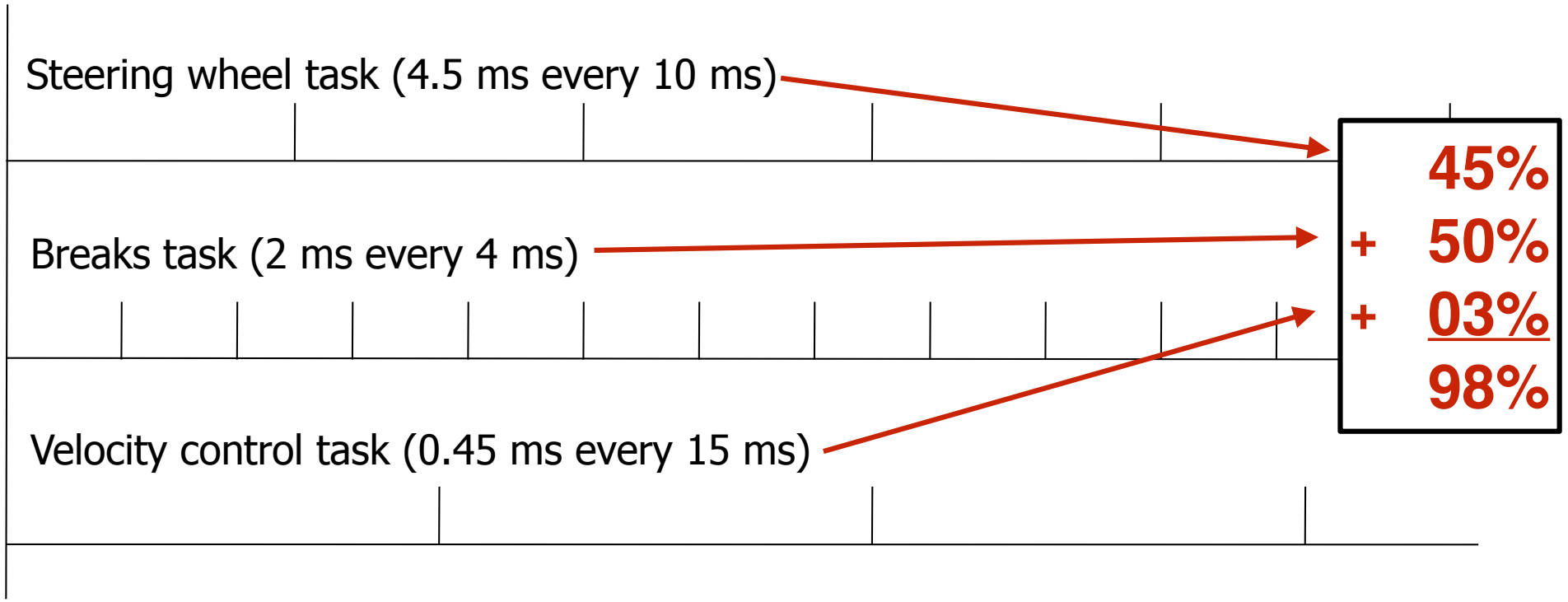
- E.G.: If you have 5 homeworks due this time tomorrow, each takes 6 hours, then you are over utilized (5x6 = 30 > 24).



Drive-By-Wire Example

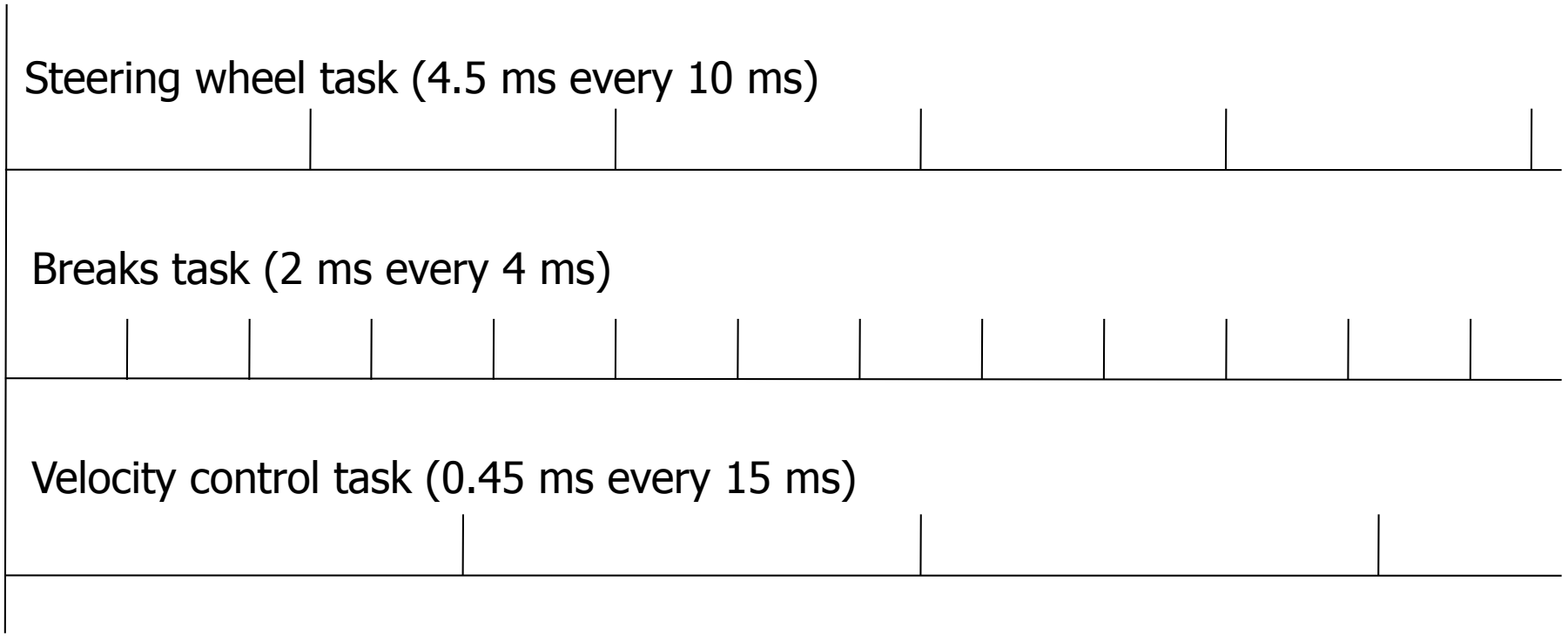
Sanity check: Is the processor over-utilized?

- E.G.: If you have 5 homeworks due this time tomorrow, each takes 6 hours, then you are over utilized (5x6 = 30 > 24).



Task Scheduling

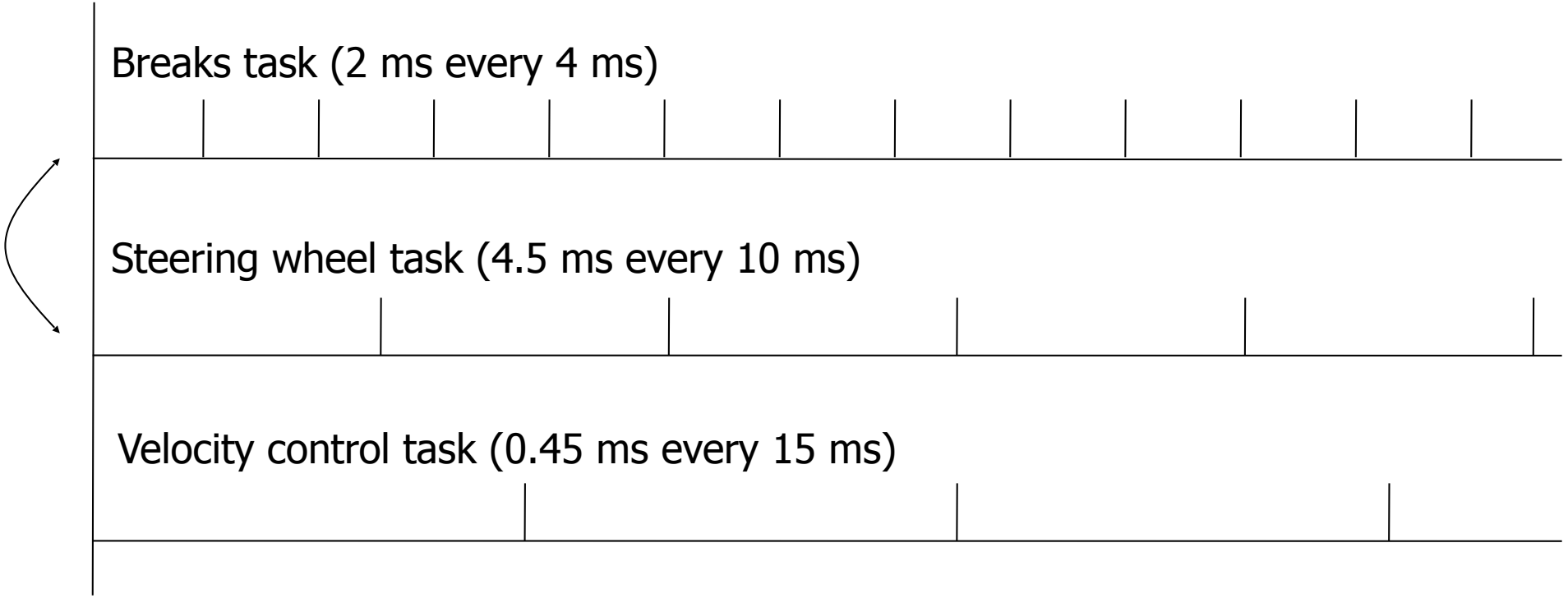
How do we assign task priorities? (SCHED_FIFO)



Task Scheduling

How do we assign task priorities? (SCHED_FIFO)

- ***Rate Monotonic*** (large rate = higher priority)

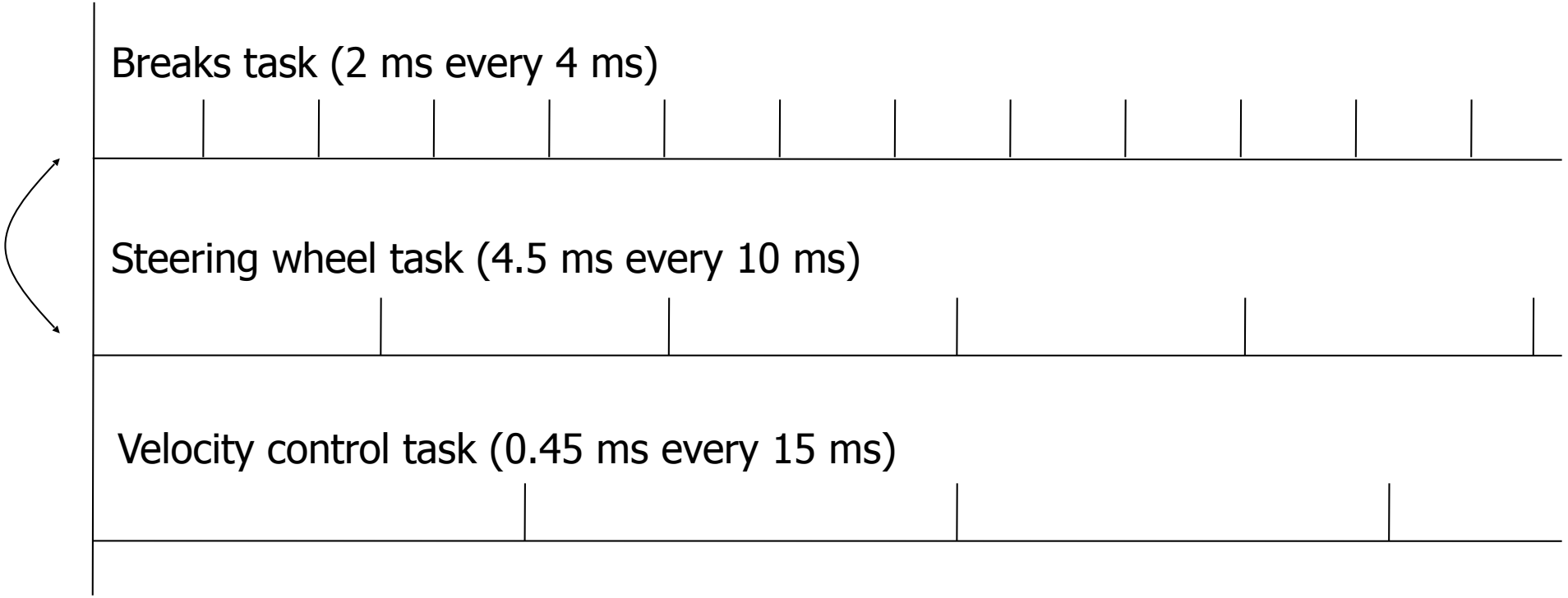


Intuition: Urgent tasks should be higher in priority

Task Scheduling

How do we assign task priorities? (SCHED_FIFO)

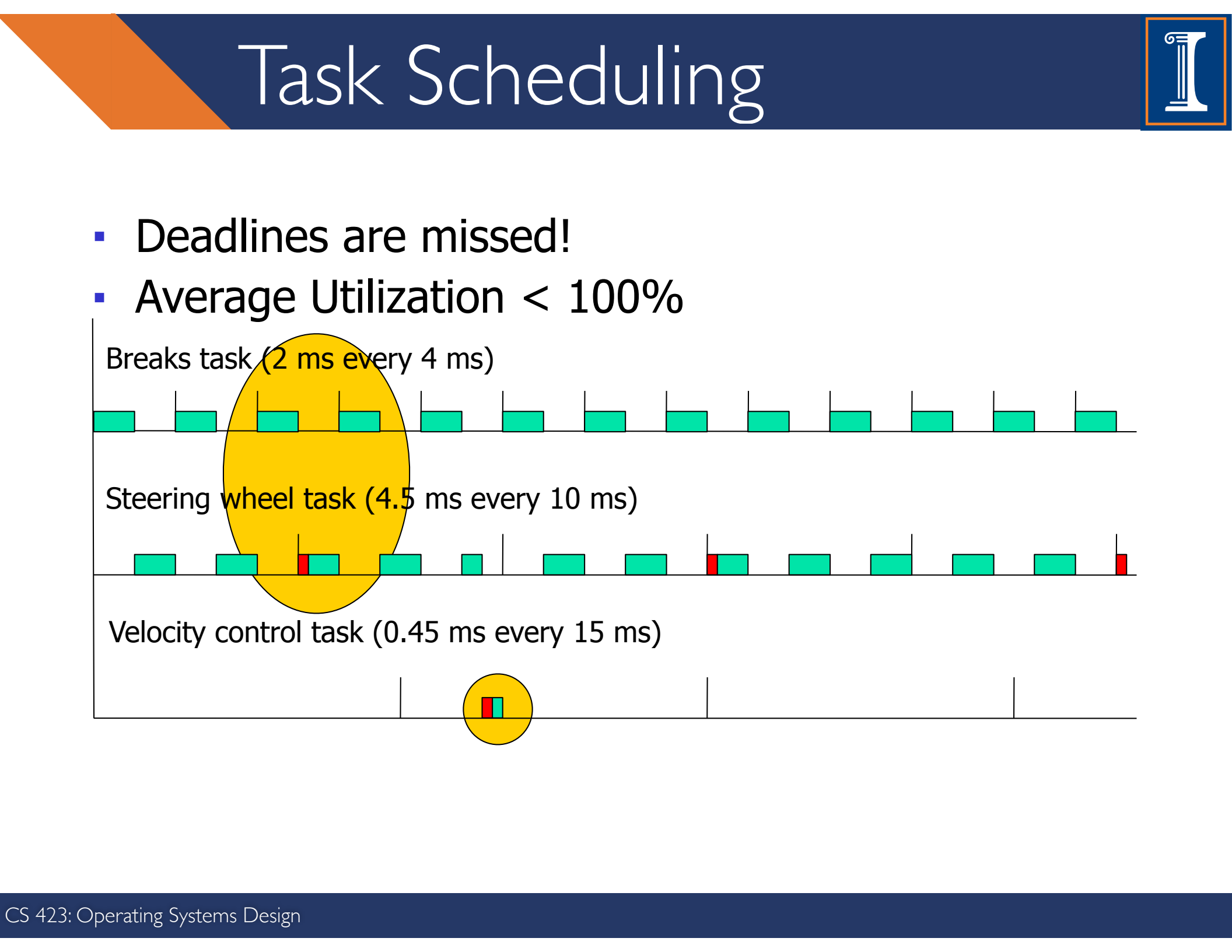
- ***Rate Monotonic*** (large rate = higher priority)



Intuition: Urgent tasks should be higher in priority

Is there a problem here??

Task Scheduling

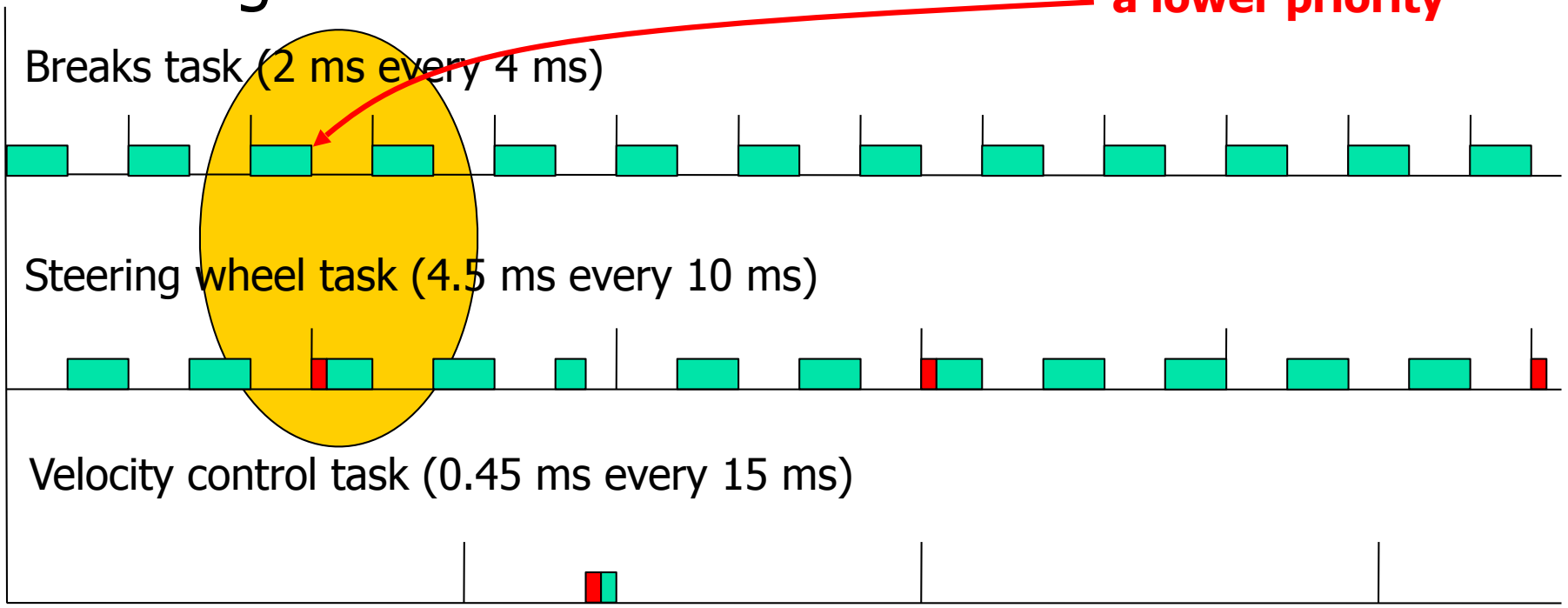


- Deadlines are missed!
- Average Utilization < 100%

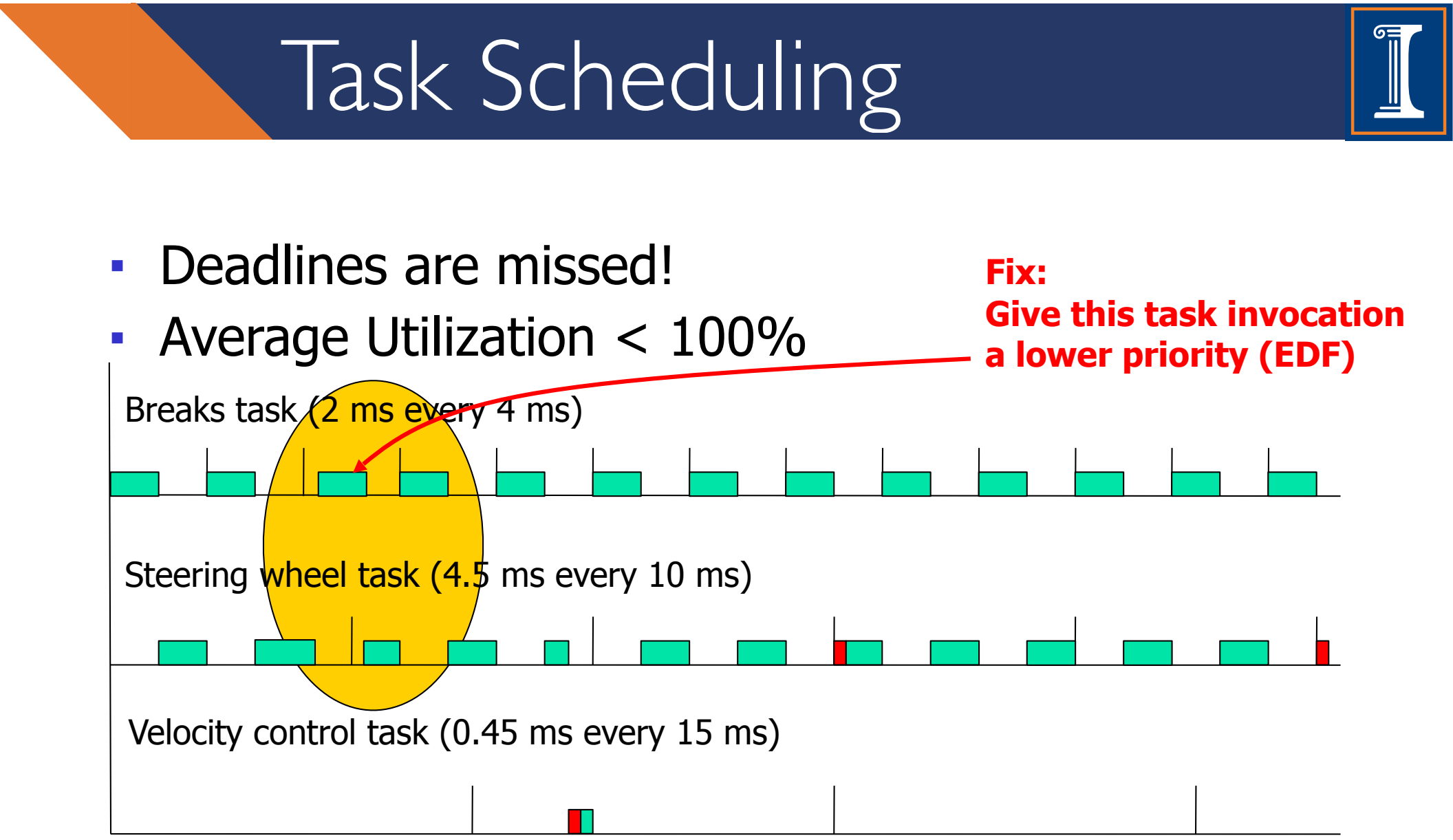
Task Scheduling

- Deadlines are missed!
- Average Utilization < 100%

Fix:
Give this task invocation a lower priority



Task Scheduling

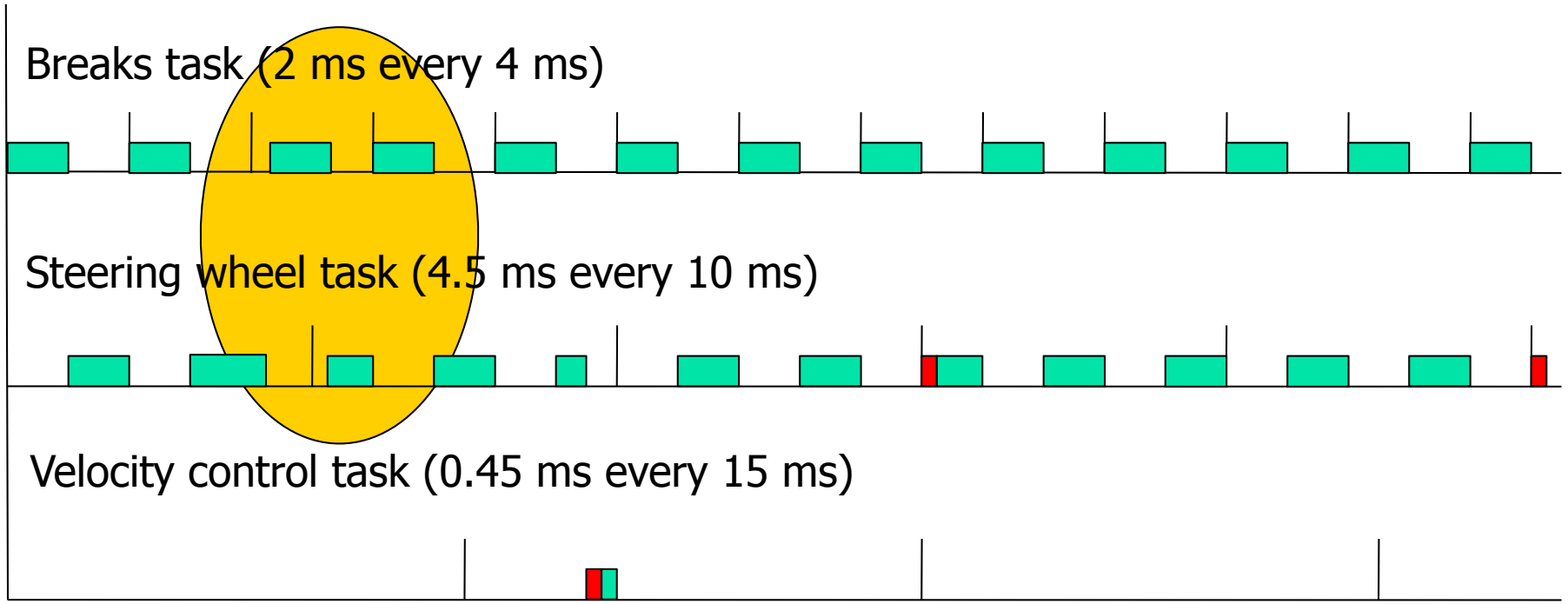


- Deadlines are missed!
- Average Utilization < 100%

Fix:
Give this task invocation a lower priority (EDF)

Task Scheduling

- Static versus Dynamic priorities?
  - Static: Instances of the same task have the same priority
  - Dynamic: Instances of same task may have different priorities



Intuition: Dynamic priorities offer the designer more flexibility and hence are more capable to meet deadlines

Task Scheduling

Re: Real Time Scheduling of Periodic Tasks…

- Result #1: Earliest Deadline First (EDF) is the optimal dynamic priority scheduling policy for independent periodic tasks (meets the most deadlines of all dynamic priority scheduling policies)
- Result #2: Rate Monotonic Scheduling (RM) is the optimal static priority scheduling policy for independent periodic tasks (meets the most deadlines of all static priority scheduling policies)

Task Scheduling

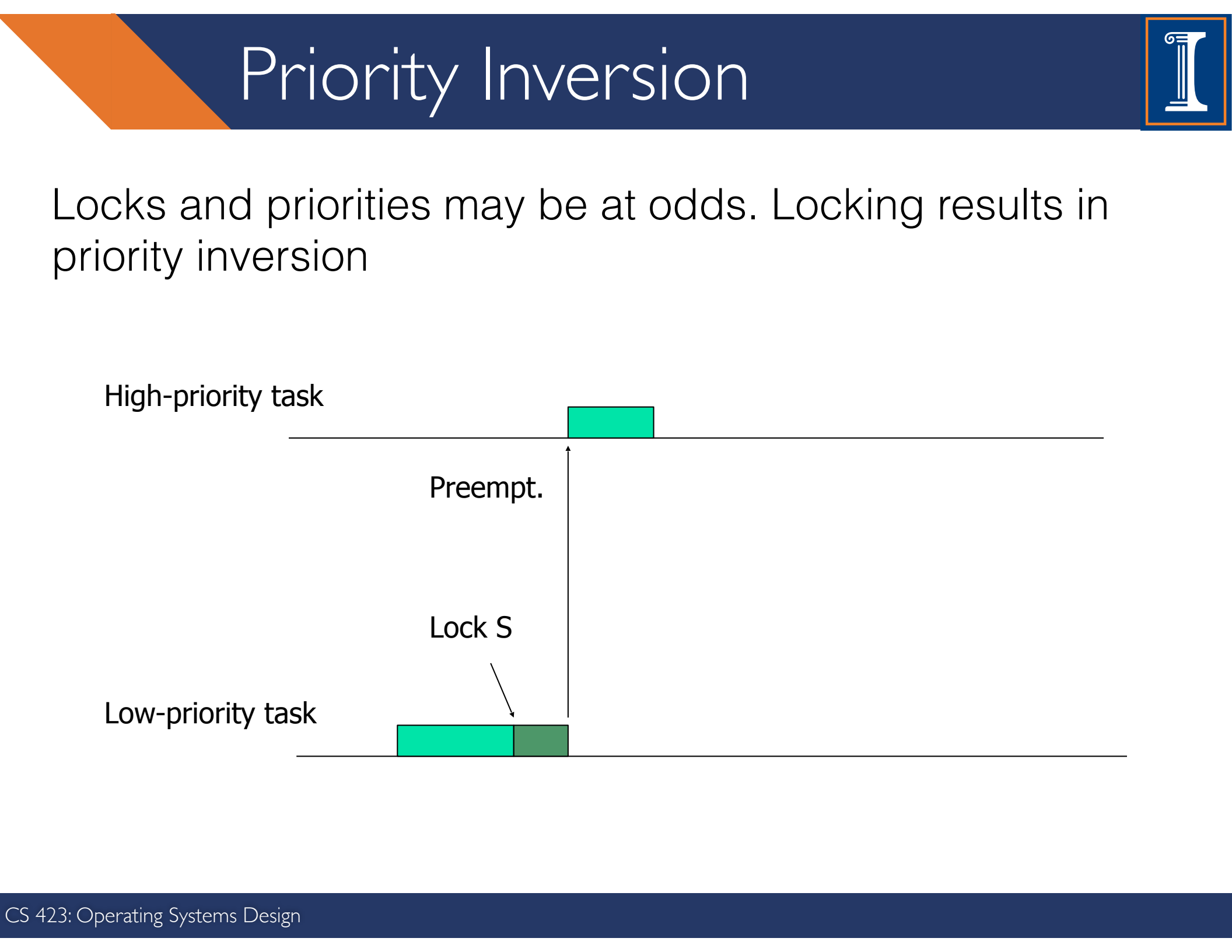
Re: Real Time Scheduling of Periodic Tasks…

- Result #1: Earliest Deadline First (EDF) is the optimal dynamic priority scheduling policy for independent periodic tasks (meets the most deadlines of all dynamic priority scheduling policies)
- Result #2: Rate Monotonic Scheduling (RM) is the optimal static priority scheduling policy for independent periodic tasks (meets the most deadlines of all static priority scheduling policies)

Locking vs. Priority

- What if a higher-priority process needs a resource locked by a lower-priority process?
  - How long will the higher priority process have to wait for lower-priority execution?

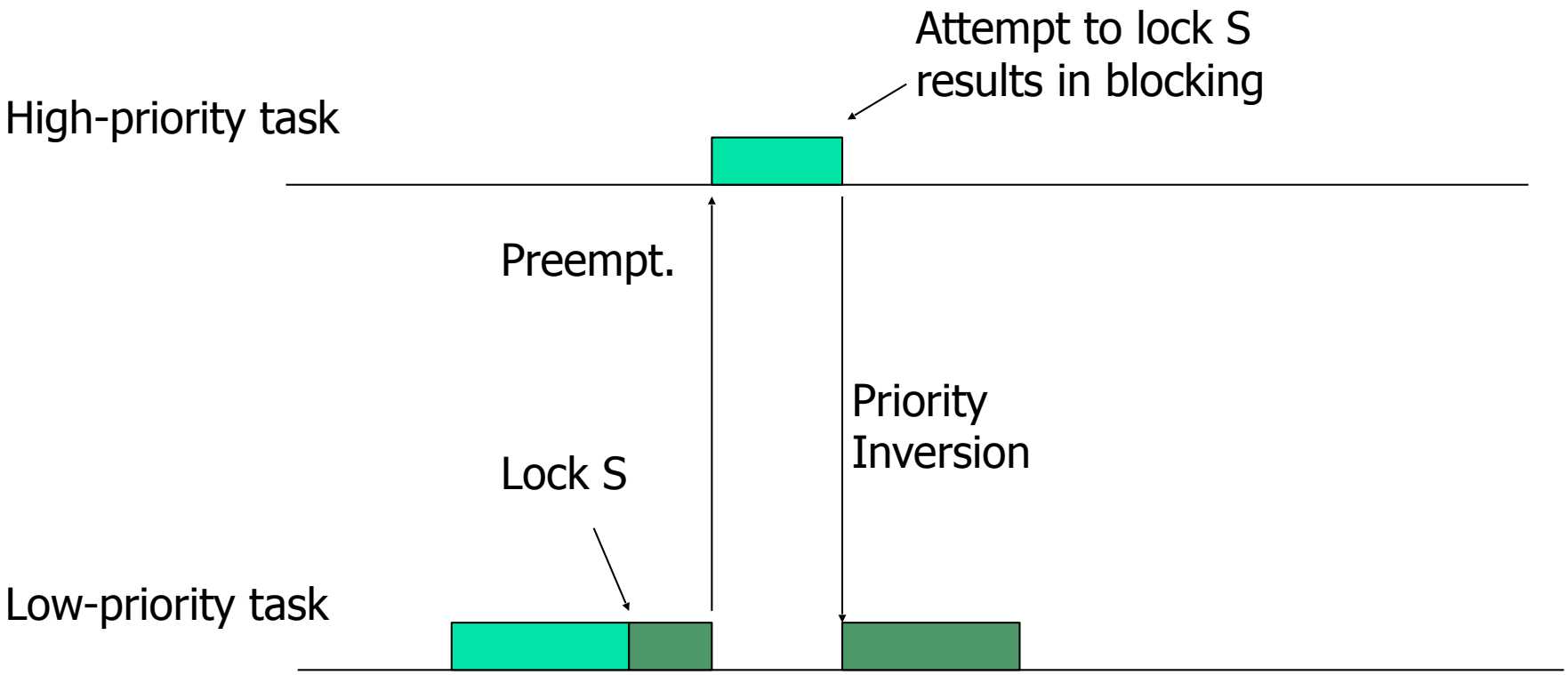
Priority Inversion



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

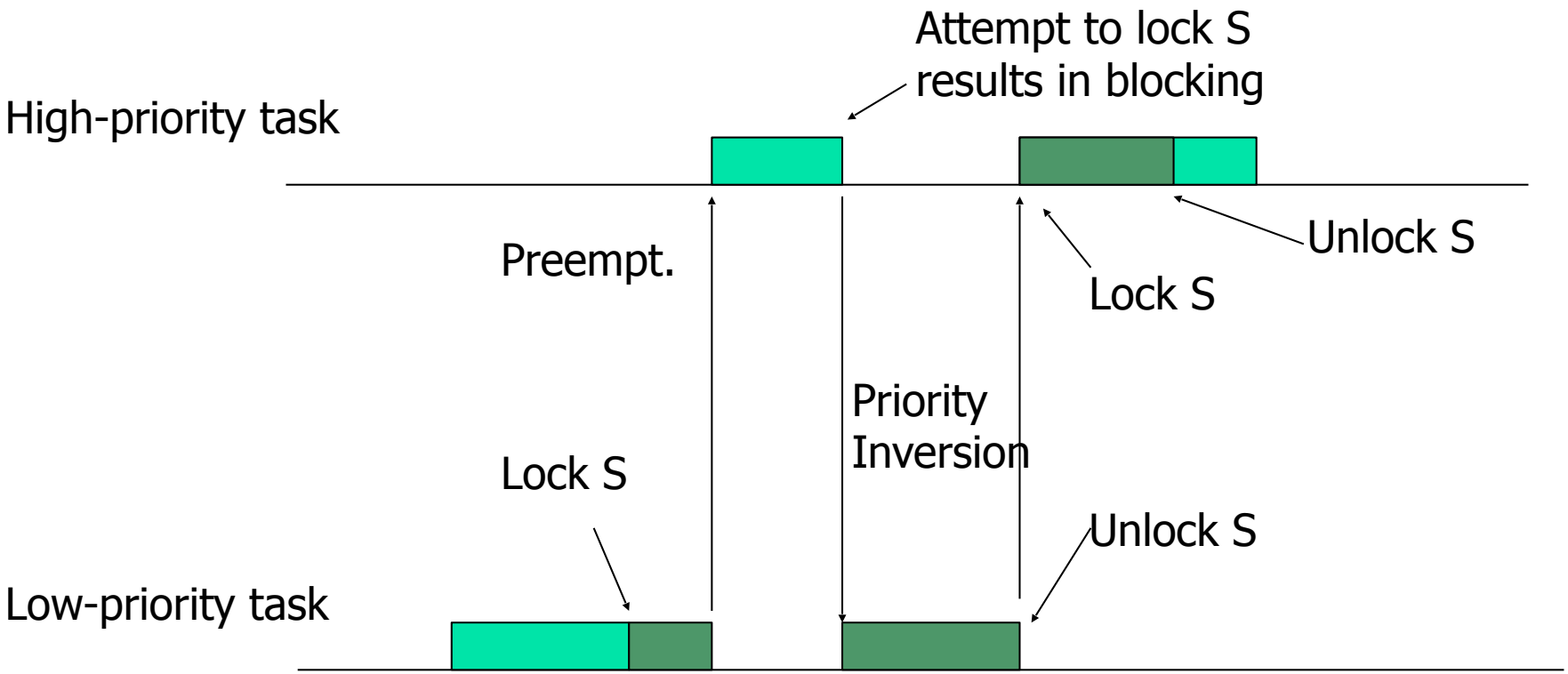
Priority Inversion

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

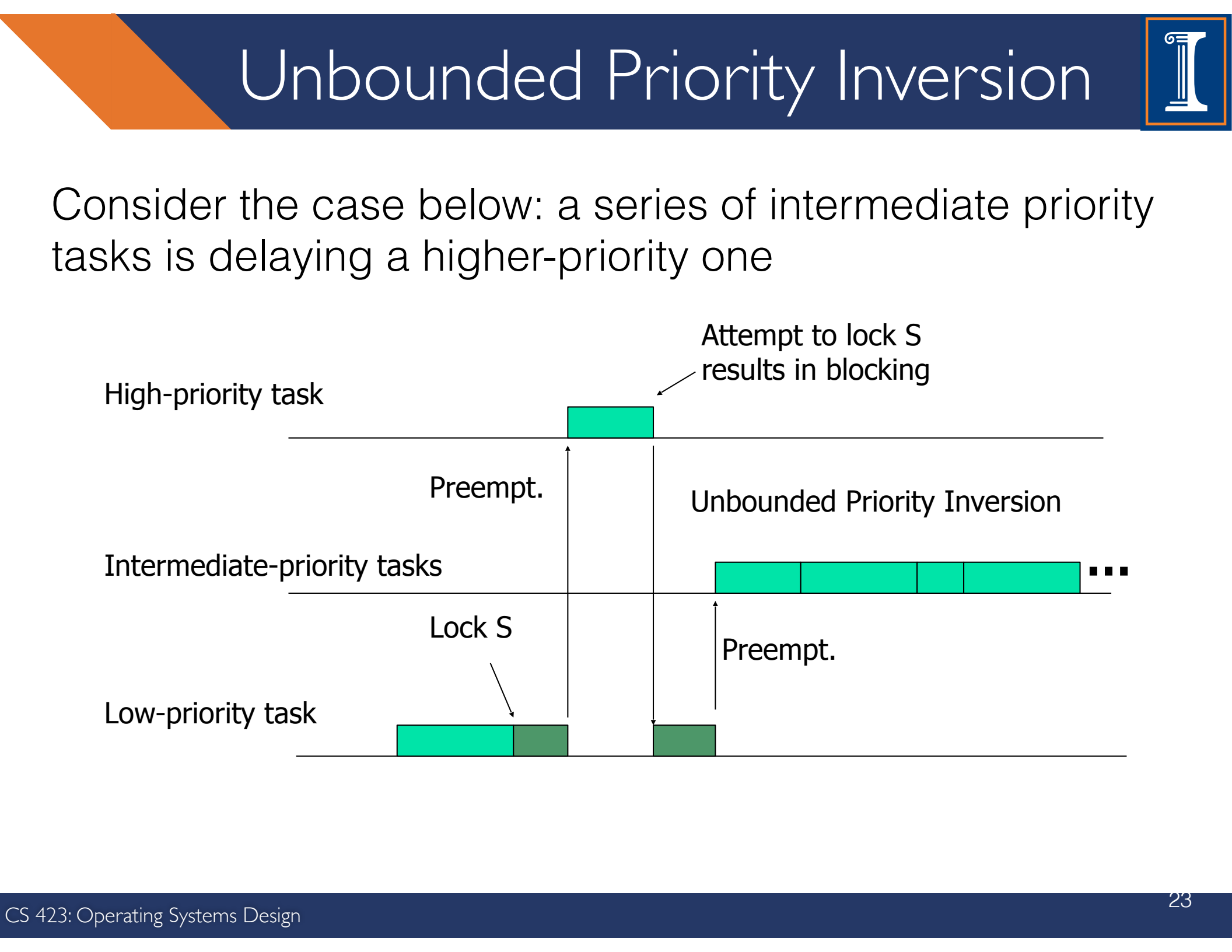


Priority Inversion

How should we account for priority inversion?



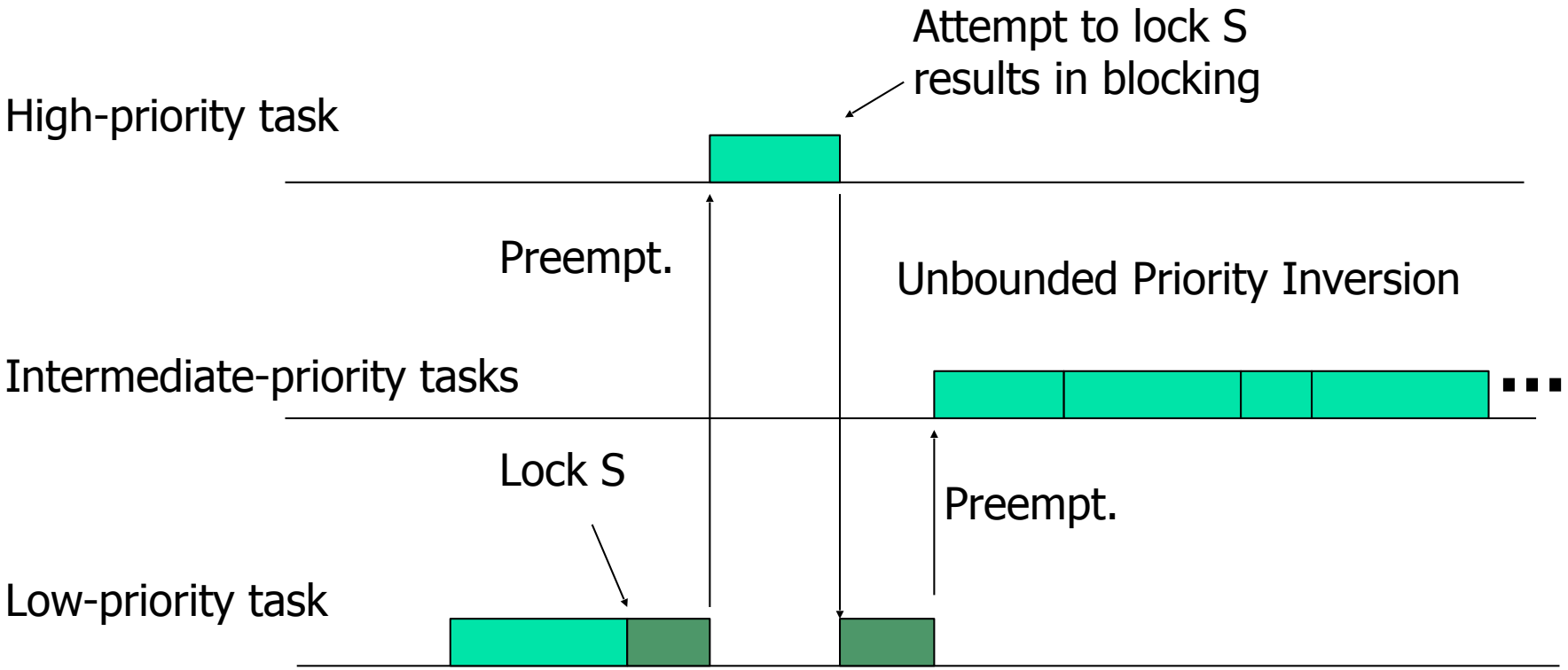
Unbounded Priority Inversion



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

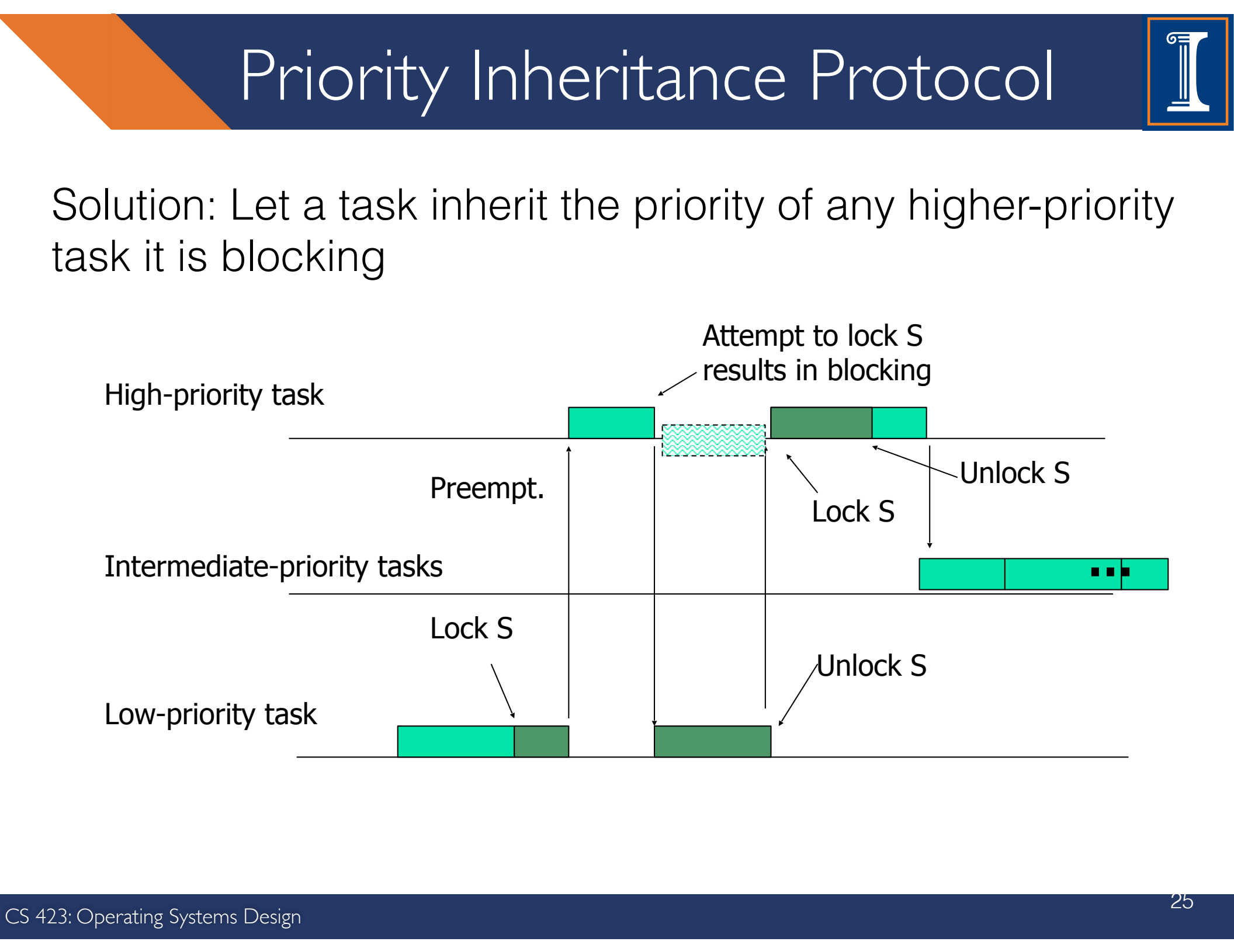
Unbounded Priority Inversion

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



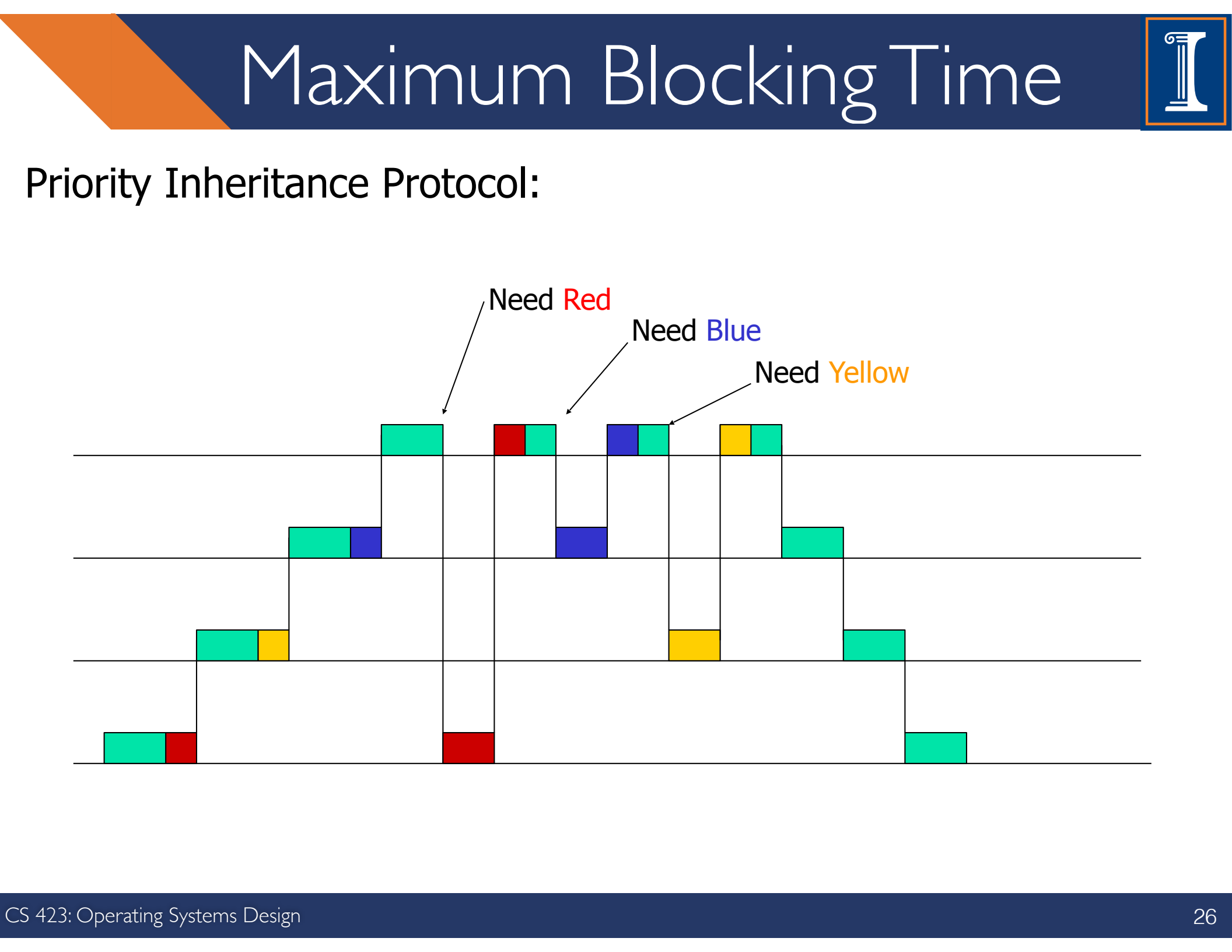
How can we prevent unbounded priority inversion?

Priority Inheritance Protocol



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

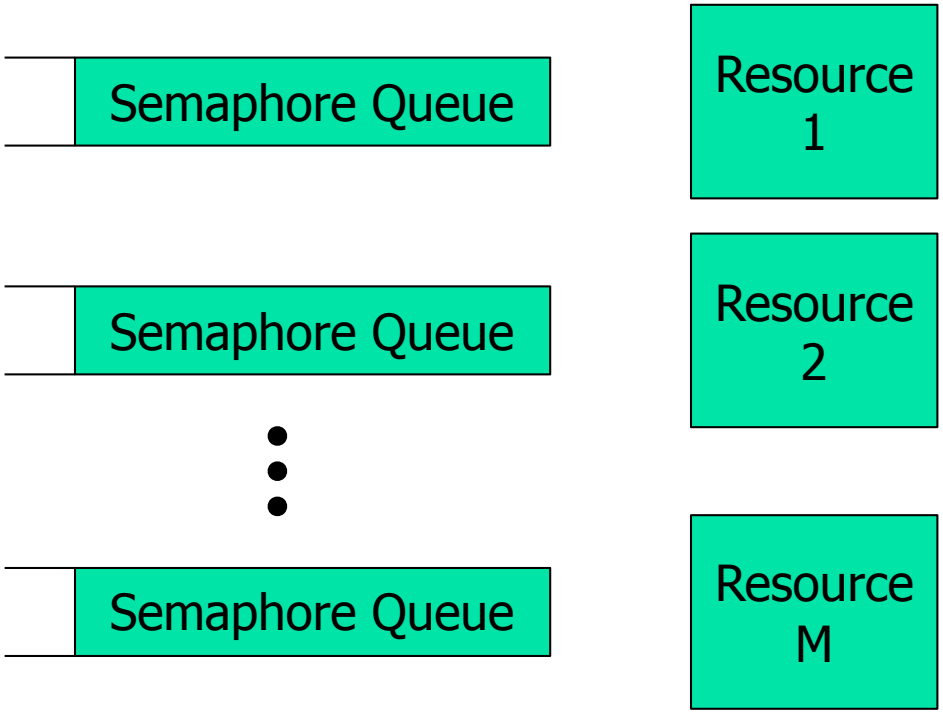
Maximum Blocking Time



Priority Inheritance Protocol:

Max Priority Inversion Time

- What is the longest period of time a high priority task will wait on a resource?



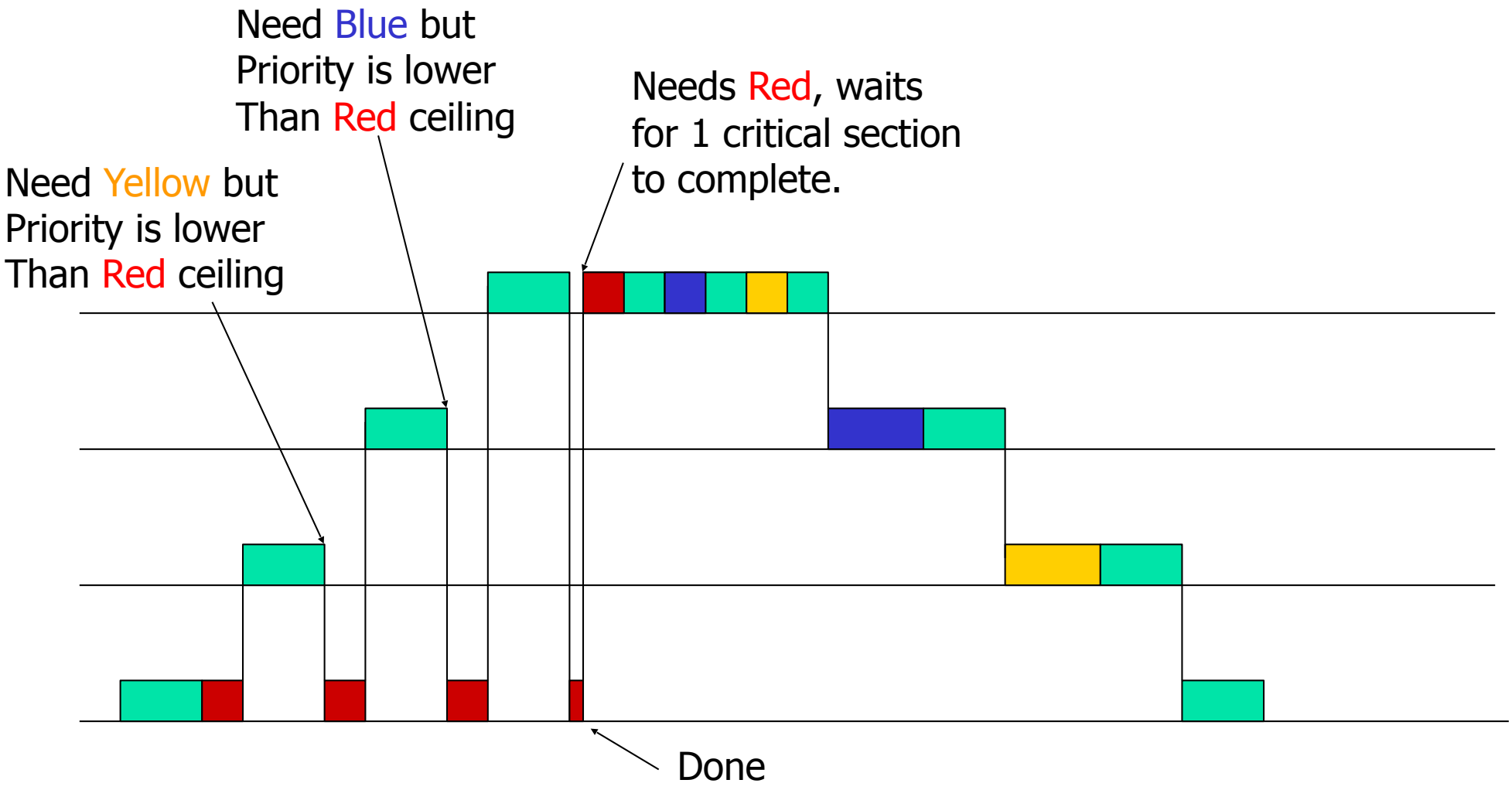
Two priority inversion scenarios to consider:
(a) Lower priority task holds a resource I need (direct blocking)
(b) Lower priority task inherits a higher priority than me because it holds a resource the higher-priority task needs (push-through blocking)

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 semaphores currently locked by other tasks (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

Maximum Blocking Time

Priority Ceiling Protocol:



Least Slack Time (LST)

- Idea: There's no point in completing a task earlier than its deadline. Other tasks many be executed first
- LST: Orders queue with nondecreasing slack times
- Pros: Can run online, might improve response times
- Cons: Needs computing times, only works for preemptive tasks, difficult implementation